Matt Thompson's

SCSI SYSTEM

FAST OS-9 Level II Drivers
For use with the MC6809 or HD6309 CPU

from

HardSoft

Reference Manual

Distributed by:

Northern Exposure
7 Glenalpine Cres
Ottawa, ON
Canada
K2P 1B5
(613)730-0329

Many hours have gone into the preparation of SCSI Sys. Please help support quality software by not pirating this commercial package.

1. INTRODUCTION

Thank you for purchasing the Registered Version of the SCSI System. Many hours of programming and testing have gone into its creation. Please read the manual thoroughly before you begin any work with your system. A thorough understanding of this manual will greatly assist you in setting up your system.

Some of the features of the SCSI System include:

- Handles both 256 and 512 byte-per-block devices.

- Handles drives larger than 120 megabytes using clustering or partitioning, your choice.

- Comes with easy to use descriptor editing and drive formatting utilities. Eliminates the need for RS-DOS formatters and 'dmode' wizardry.

- Full reset support for stuck drives.

- A new SS.SCSI SetStat call allows applications to access the drive directly under OS-9.

- Designed to work with most CoCo 3 hardware, including Disto, Ken-Ton, and LR-Tech SCSI interfaces, Seagate 'N' SCSI drives, the Adaptec SCSI controller board, and Multipaks.

The registered version also comes with the Turbo Cache, accelerated I/O handshaking and optional 6309 optimizations for CoCo SCSI speed like you've never seen before. Megareads below 20 seconds are possible!

2. NOTES

It is assumed that the user of the SCSI System knows how to create new boot disks with new driver modules and descriptors. Knowledge of this process is fundamental to the use of OS-9, and is too lengthy to describe here.

In this document, 'interface' refers to the CoCo plug-in Pak, while 'controller' is either the SCSI circuitry right on the drive, or the Adaptec controller board. Also, 'sector' refers to logical OS-9 256-byte sectors, while 'block' refers to the physical data blocks on the SCSI device, be they 256, 512 or whatever bytes.

Operation of the SCSI System is not guaranteed with more than one SCSI interface in a Multipak at a time, because of the way RBF handles things.

Becuase of the hardware limitations of the interface paks, you cannot use newer SCSI-2 drives, data parity, or use the SCSI System in a multiple-initiator system.

3. TURBO CACHE

For a long time, CoCo SCSI interfaces had a reputation for being slow compared to other CoCo hard drive interfaces. The registered version of the SCSI System comes with Turbo Cache, and it erases this notion.

This cache improves disk performance by keeping the most likely to-be-used data in RAM. The cache occupies an 8K block of RAM. However, the loss of an 8K block is worth it. You'll find that you won't need a ramdisk for C compiles, and you won't need to keep as many commands in memory since they load so much faster.

The cache is organized into 3 segments of 2K each, and they are replaced using a least-recently-used (LRU) algorithm. When the driver gets a read request, it checks to see if the desired sector is stored in any of the segments. If it is (a cache hit), that sector is accessed right out of RAM, without a disk access. If it is not in any segment (a cache miss), the LRU segment is refreshed with new data by a disk read operation. It then becomes the most-recently-used segment, and the other segments are aged accordingly. All segments are aligned on 2K boundaries with respect to the disk sectors.

To ensure that the data on the disk never gets corrupted, the driver uses a write-through scheme, which means every write request

is always immediately written to disk. However, all writes are also posted to the cache, and if the sector to be written is not in any of the segments, the LRU segment is updated with a read operation. This helps to reduce the amount of read-modify-write SCSI operations needed when writing a 256-byte OS-9 sector to a 512-byte block.

Besides speeding up multitasking disk access, the segmented nature of the cache speeds up certain monotasking operations, such as deleting a fragmented file. RBF scans back and forth from the file descriptor sector to the allocation bitmap for each file segment, and a 48-segment file near the end of a slow disk can take 10 seconds to delete. The segmented cache keeps the file descriptor around all the time, so that the disk head stays on track 0 during the delete, greatly speeding it up.

The remaining 2K of the 8K block is used for up to eight LSN0 buffers. LSN0 is rarely changed, but it is accessed every time a file is opened. This includes change dir operations or when a command is executed and has to be loaded off of the disk. The LSN0 buffer keeps LSN0 in RAM all the time, so that head seeks are reduced even further, above and beyond the regular cache. The buffers are filled when the respective LSN0 is accessed for the first time. A write to LSN0 also updates the buffers. If there are more than 8 logical drives, drive numbers 8 and above are not LSN0 buffered.

In addition to the cache, the registered version has other speed-ups. There is a turbo handshake option that speeds up the flow of the data on the SCSI bus. This is opposed to the tried and true regular handshake scheme that the shareware version uses. The registered version also comes with drivers optimized for use with the 6309. They utilize special 6309 instructions and the native mode to speed things even more. See the section on 'Interleave and Speed' for more information.

4. INSTALLING THE SCSI SYSTEM

SCSI stands for Small Computer System Interface, and is pronounced 'scuzzy'. It is a parallel interface bus for computer peripherals,

and it has become extremely popular in mass storage devices such as hard drives, tape drives and CD-ROMs.

Installation of the SCSI System should be trouble free if you are patient and check through the following steps carefully. While it may involve a considerable amount of work, once it is done it'll be worth it. These steps mention the 'scsidesc' and 'scsifmt' utilities, included with the SCSI System. Their use is described in seperate sections, to keep this section simpler.

I recommend that you get the following utilities, available on Delphi and elsewhere, if you haven't already: the 'dmode' RBF descriptor modifier, the 'ded' disk editor, and the 'vu' text file browser. They are very handy. Also, if you are using any sort of ACIA serial interface, I recommend that you download Bruce Isted's 'sacia' ACIA driver and the patched 'clock' modules to go with it. It supports up to a 3.75K buffer and other goodies so that no character ever gets lost.

For more information on the included files and modules, see the 'files.doc' file.

If you are already using a SCSI drive, you should verify operation of the SCSI System with the drive at 256 bytes per block before reformatting the drive for 512. That way, if it doesn't work, you aren't stuck out in the cold.

If you are already using a hard drive, you should always have a "floppy only" boot disk prepared that doesn't "need" the hard disk to boot. This disk should, of course, have the necessary drivers and descriptors installed to access the hard drive once you are booted. This way, you are never locked out of the system if something goes wrong with the hard drive.

Adding Your First SCSI Hard Drive

Since everything you have ever done is from floppy, then you already have a floppy-only boot disk. Be sure that the SCSI System utilities and the driver module are handy for creating new boot disks.

Use 'scsidesc' to create new descriptors that describe the drive you are attaching. Since you have nothing to lose, you can jump straight to 512-byte blocks when editing the descriptors.

Hook up the hard drive to your interface, and select the SCSI ID of the drive with the jumpers on the controller board. Your main drive is best set for ID 0.

Create a new boot disk with the correct SCSISYS driver for your system, and the descriptors you have created. But be sure that the '/dd' descriptor is for the floppy, and not the hard drive, for this first new boot disk. Use a different floppy for every new boot disk you make, so you'll always have the previous one to fall back on if the new one doesn't work.

Boot up with the new boot disk. The hard drive shouldn't do anything. Now, use 'scsifmt' to physically format (if applicable), verify, and logically format the hard drive.

If that worked, you should now be able to access the hard drive. Change directory to the drive, do a dir, and try it out. Everything should be working fine. If something won't work, refer to the 'Problems Checklist'.

Patching Things to Work with the Hard Drive

Once you get your SCSI drives running with the floppy-only boot, you will want to patch things so that you boot directly off of the hard drive. Actually, you can't boot 100% off of the hard drive, unless you burn an EPROM with patches to RS-DOS (or other DOS). If you wish to pursue this end, I leave it up to you.

What most CoCo users with hard drives do is merely modify things so that 'Shell' and 'GrfDrv' are loaded from the hard drive during boot up, saving time. You still have to type 'DOS' and you still need a floppy with 'OS9Boot' on it. The following steps assume you have 'ded' or some other disk file editor to make the necessary changes.

Create a CMDS and SYS directory on the hard drive, and copy all the commands and files from the floppy into them. Be sure that 'Shell' and 'GrfDrv' are in the CMDS directory, with the execution bit set.

Use 'ded' to modify the 'Init' module. Change the '/D0' to '/DD'. Be sure to set bit 7 bit on the last 'D'. You may also want to change '/Term' to '/W0' or whatever, if you haven't already. This changes your boot up drive (and window). Also, patch byte offset $00C from $0F to $0C, if you haven't already. This fixes another bug that has nothing to do with the SCSI System (for details, see the book 'Inside OS-9 Level II'). Remember to verify the module before quitting 'ded'.

Use 'ded' to modify any '/D0' or '/H0' in cc3go to '/DD'. Remember to verify the module before quitting.

Make sure that '/DD' is a SCSI System descriptor for the hard drive where the CMDS and SYS directory are.

The above changes should be sufficient for boot up. Create a new boot disk with the new DD, Init and CC3go modules, and it may just work!

Many other software packages for OS-9 are also hardcoded to work with '/D0' or even '/D1', such as certain modules of the Microware C Compiler, Microware Pascal, and many of the OS-9 game packages sold by Tandy. If you have 'ded', you can scan for occurences of '/D0' in just about any program, and patch it to '/DD'. Pay attention to the line termination used (carriage return, zero, bit 7 set, or fixed length).

Adding to or Reformatting an Existing SCSI Hard Drive System
If you have a hard drive system already up and running with another SCSI driver, you should always create a "floppy-only" boot disk that does not need the hard drive at all in order to boot, for emergencies. For now, it should contain your old SCSI drivers.

Boot up with this floppy-only boot disk. Now, use 'scsidesc' to create new descriptors for your hard drives. You cannot use your old ones with the SCSI System driver.

If you are reformatting from 256 to 512 byte blocks, your descriptors should be first set for 256-byte blocks, so you can verify operation of the drivers with your system! You can then reformat and make another boot disk at 512, once you're sure

everything is working. If you are adding a new drive to the system, you can jump straight to 512-byte blocks for its descriptor.

Create another floppy-only boot disk, only this time with the appropriate SCSI System driver module and the new descriptors for your system. Reboot with this disk and verify operation of the 256-byte drives. If something won't work, refer to the 'Problems Checklist'.

Now you must decide on a backup strategy for the 256-byte drive(s). Reformatting will erase the data on them. If the drive doesn't contain anything you care about, or has never been used before, you don't need to back it up. Otherwise, something has to be done to save your data. However, the gain in storage capacity will be well worth the trouble.

If you just have a single drive system, you are simply going to have to back up the disk on floppies or some other media. If you are adding a brand-new 512-only drive to a running 256 system, you may be able to use it to backup the 256 drives temporarily, one by one. If you have more than one 256 drive, you can backup the largest one on floppies, format it for 512, then use it to backup the other ones temporarily as you reformat them, one by one. Then you restore the files back to the largest drive again. You *must* back up with file by file utilities such as the 'hdkit' utils, 'stream', 'dsave', 'dircopy', etc., as opposed to sector by sector utils such as 'hdb', 'backup' and 'dup'. No more will be said about backing up except to say that you have been warned. I cannot be responsible for lost data.

As you reformat drives, you can save yourself the trouble of rebooting by just modifying the descriptor in memory with 'scsidesc', and only 'cobbler'ing each change as you go along. Remember to change the descriptor *before* logically formatting!

Once you get all drives physically and logically formatted up to 512-byte blocks, any backed-up data restored, and a final boot disk made, your system is ready for use! Keep this latest floppy-only boot disk handy. If you want to go back to booting off of the hard

drive, you may now make another boot disk with '/dd', 'cc3go' and 'init' modified as before.

Alternatively, you can just leave the drive at 256 bytes per block. You may want to do this if you have RGB-DOS set up on it. At this time, I don't have a patch to allow RGB-DOS to access 512-byte blocks. But if I figure out something, I'll post the patch on various forums and echoes. Nonetheless, the Turbo Cache and other speed-up options of the registered version will boost the speed performance (under OS-9) of 256-byte block drives the same as for 512-byte drives. Thus you don't absolutely have to backup or reformat to benefit from the registered version.

5. THE SCSIDESC COMMAND

The 'scsidesc' command is a custom utility for the SCSI system designed to take the guesswork out of creating and modifying descriptors. It is much more user friendly than 'dmode'. The 'dmode' command can still be used to modify SCSI system descriptors, however. See the 'Descriptor Format' section for more information on the various bit fields.

NOTE: You *cannot* use your old descriptors from other SCSI drivers with the SCSI System driver. You *must* create new ones.

When you enter 'scsidesc', you will see a menu listing each field in the descriptor. To create a custom descriptor, you simply change any of the fields by pressing the letter associated with it. Then you will be prompted to enter a new value for the field. An invalid entry will be ignored, but you must enter something. You can also select the Transfer option, which pops you into a submenu where you can load and save descriptors in and out of the edit buffer. The Transfer menu is discussed later. First is a breakdown of what each descriptor field represents, and what you should enter for it:

Port Address: This is the hardware address of the SCSI interface. You do not have to remember the address for your interface. The editor will let you choose based on the interface type! If you have a Ken-Ton or LR-Tech interface, it is possible to change the

default port address ($FF74) to an alternate address ($FF70) via a jumper on the board, hence the two choices of addresses. See the docs for those interfaces for more info.

Drive Number: This is the *logical* drive number of the drive. Ideally, the main drive is drive 0, but this is just a convention. The logical drive number is independent of the SCSI ID and LUN of the drive, which can be set seperately. The maximum number of logical drives that can be handled is fixed when SCSISYS is compiled. The default number is 8. NOTE: If any two descriptors refer to the same logical drive (eg 'DD' and 'H0'), they *must* have the same drive number! NOTE: If you are using partitioning, each parition is a seperate logical drive, and *must* be assigned a seperate Drive Number!

512 Flag: Set this field to 1 if the drive is 512 bytes per block, or to 0 if it is 256 bytes per block.

SCSI ID: The SCSI ID is the SCSI controller IDentification number assigned to each SCSI device. For embedded SCSI drives, each drive has a seperate SCSI ID number, which is set by jumpers on the drive's circuit board. On the other hand, if you have more than one drive hooked up to an Adaptec controller, they will each have the same SCSI ID, but different LUNs.

SCSI LUN: This is the SCSI Logical Unit Number. Virtually all embedded SCSI drives have a LUN of 0. If you are using an Adaptec board with two drives, the 1st drive's LUN is 0, and the 2nd drive's LUN is 1.

Error Type: Set this to 1 to have the raw SCSI errors (sense codes) returned when an error occurs. This is useful for debugging purposes. Otherwise, leave this field at 0 to have the errors converted to their nearest OS-9 equivalent. See the 'Errors' section for more information.

Cluster Size: Although the cluster size is defined in LSN 0 of a disk, it is not specified in ordinary RBF device descriptors, so the SCSI System has added it to its descriptor definition.

Cylinders: Set this to the number of cylinders on the drive.

Heads: Set this to the number of read/write heads on the drive.

Sectors/Track: Set this to the number of logical sectors per track on the drive. This value is used for both the regular (IT.SCT) and the track zero (IT.TOS) values in the descriptor.

If you are unsure of any of the above three parameters, here is a chart of the most common Seagate 'N' series SCSI drives:

Model #:	225	251	277	296	125	138	157	177	1096
Meg (@512):	21	43	65	85	21	32	48	61	84
Cylinders:	615	818	818	818	407	615	615	921	906
Heads:	4	4	6	6	4	4	6	5	7
*Blks/Trk@512:	17	26	26	34	26	26	26	26	26
Blks/Trk@256:	32	44	44	64	44	44	44	44	44

If you are using the Adaptec controller, the number of blocks per track is independent of the brand of drive that is connected. The only variables are the number of cylinders and heads on the drive.

	Adaptec 4000A (MFM)	Adaptec 4070 (RLL)
*Blks/Trk@512	18 (17 if ilv 1:1)	26 (25 if ilv 1:1)
Blks/Trk@256	33 (32 if ilv 1:1)	47 (46 if ilv 1:1)

*: Note that the number of blocks/track '@512' in the tables is the number of 512-byte physical blocks. When determining the OS-9 sectors/track value for the descriptor, the number of physical blocks/track of any 512-byte disk must be multiplied by 2. Therefore, the 1096N has 26 x 2 = 52 logical OS-9 sectors per track at 512 bytes per block.

If you own a drive other than one of those listed in the tables, and you don't know the parameters for it, plug in values for the following formula to make it work. The number of blocks available might be printed on the drive somewhere. These are generally 512-byte blocks, so remember to convert them to twice as many 256-byte sectors.

Cylinders * Heads * Sectors/Track * 256 = bytes available

OR

Cylinders * Heads * Sectors/Track = logical sectors available

Some drives, such as Quantum, have variable numbers of blocks per track, so you'd have to use a fudged value anyway. Assuming you know how many 'meg' your drive has, you can simply pick reasonable numbers that satisfy this equation. Note that the advertised 'meg' of a hard drive is in terms of 10^6 (1,000,000-byte) 'megs', and not 2^20 (1,048,576-byte) 'megs'. The same reasoning follows for 'gig' hard drives.

If you are using partitioning, all of the fixed partitions are 262,144 sectors, so just use 1024 cylinders, 32 sectors/track and 8 heads and that will do the trick. For the last partition on the disk, you take the total number of sectors on the disk and subtract 262,144 for every partition that comes before it. This gives the number of sectors left over for the final partition. This *must* be less than 262,144 sectors, or else you'll need another partition.

The purpose of the Cylinders, Heads and Sectors/Track fields is for RBF's internal calculations. The driver deals in terms of logical sector numbers, and doesn't care what values are used for these fields. However, to make sure RBF lets you use the whole drive, the above values should work out to a shade *greater* than the actual capacity of the drive. The logical formatter will map out the extra sectors in the allocation bitmap.

Segment Size: This value determines how many sectors are allocated at a time when a file is written. Setting it too low can lead to frequent file fragmentation errors, and setting it too high can slow down free space searches when writing files. It defaults to 8. This field is used by RBF, but not by device drivers.

Device Name: A descriptor needs a device name, so enter it here. You are limited to two characters, no more, no less. Names like 'DD', 'H0', 'z3' or whatever you want are acceptable. You can use upper or lower case. The device name should begin with an alphabetic character (A-Z,a-z).

Interleave: You may set this field to reflect the Interleave value of your hard drive. This is purely for your reference. Neither the drivers nor the physical formatter look at this value.

Multipak Slot: For Disto users who have their controller in a Multipak, they will need to set the slot number of the controller. Luckily, Ken-Ton and LR-Tech interfaces are mapped into an address range exempt from Multipak switching, so they don't need any Multipak handling. Non-Multipak versions of the drivers ignore this field. Note that the Disto + Multipak driver will work without the Multipak plugged in, although you'd best use the non-Multipak Disto driver if the arrangement is to be permanent.

No-Cache Flag: With the registered version, you can set this flag to disable the caching for a particular drive if it already has a built-in cache. This may or may not improve performance, it depends on the drive. LSN0 is still buffered, however, since it is used so often.

No-Turbo Flag: With the registered version, you can set this flag to disable the accelerated handshake mode, if your drive can't handle it or you are having reliability problems. The shareware doesn't have accelerated handshaking, so this field is ignored.

Partition: This is the partition number of the logical drive. Set this to zero if partitioning is not used. Otherwise 1 is the partition 0, 2 is partition 1, etc.

6309 Stack: Set this flag if you are using NitrOS9 with your 6309, otherwise leave it at zero for Turbo Boost 6309 or regular 6809. This flag is valid even in the shareware version. It also may be removed in the future if I add auto-detect capability to the drivers.

6. Moving Descriptors Around

When using 'scsidesc', there are three places a descriptor can be. They can be stored on 'disk', or they can be in 'memory' as part of the system module directory, or one can be stored in an 'internal' descriptor buffer in 'scsidesc'. The Descriptor Editor affects the

values stored in the internal buffer. However, in order to create or edit descriptors, they must be loaded into or copied out of the internal buffer. This is where the Transfer menu comes in.

You will have the choice of either 'getting' a descriptor into the buffer, or 'putting' it from the buffer. You can then select the source or destination, namely 'disk' or 'memory'. When doing disk descriptors, you will be asked for a filename which can be any OS-9 pathname. When getting memory descriptors, the module name of the descriptor is used. When putting memory descriptors, the Device Name field in the Descriptor Editor determines the module name to be used.

NOTE: When putting a descriptor to memory, it must already exist there! That is, the amount of trickery needed to fool OS-9 into creating a new memory module on the spot is too great to implement. So you can only overwrite existing descriptors in memory. Brand new descriptors can only be saved to disk. All descriptors must be SCSI System descriptors. No other descriptors can be edited by 'scsidesc' effectively.

If you must create a new descriptor in memory, you can load one off the disk. The third option, Load Descriptor, reads a descriptor from the pathname you specify and creates a brand new memory module, bypassing the internal buffer, by forking the OS-9 'load' command. If the 'load' command fails, you will not be notified. You can save memory descriptors either with the OS-9 'save' command, or by using 'scsidesc' as an intermediary.

So just fill in values for your drives with the Descriptor Editor, then save the descriptors to disk with the Transfer menu. Then you can create a new boot disk with these descriptors. If they need any changes, you can change the memory and/or disk descriptor with 'scsidesc' as need be. Just be sure to create a new boot disk every time you change an important parameter, or else you might lock yourself out of your system.

7. THE SCSIFMT COMMAND

The 'scsifmt' command gives you an easy, user-friendly way to

format and verify your hard drives. You can reformat the hard drive, set up the OS-9 file system on it, and repair failing blocks on the drive.

While 'scsifmt' is executing, DON'T DO ANYTHING IN ANY OTHER WINDOW, in paticular access the hard drive, or you could permanently foul up your hard drive! This is true of any SCSI formatter, not just the SCSI system. If the power goes off while you are formatting, this could also permanently ruin the drive! So don't trip on the cord or do the format during a thunderstorm! Only use 'scsifmt' for your SCSI drives. It is not designed for non-SCSI hard drives or floppies.

NOTE: The physical format and verify passes do not need the SCSISYS driver and descriptors to be in memory (ie as part of 'OS9Boot'), although they can be loaded if you want. 'Scsifmt' has a "built-in" SCSI port driver. Because of this, it will ask you for the SCSI ID of the drive, as well as the interface type, in order to send commands to the drive. If you are using a Disto you will also need to give a Multipak slot number. If you are not using a Multipak, enter anything since it doesn't matter.

Physical Format

The first thing you must do is decide if the drive should be physically formatted. Reformatting a drive from 256 to 512 byte blocks generally frees up a significant amout of space (ie 15 megabytes extra on the 1096N). A physical format will completely reinitialize the disk drive surface. If you are converting a 256 drive up to 512 for the extra space, then you must physically reformat. If you are hooking up a brand new SCSI drive to the system, most of them are shipped from the factory with an interleave of 1:1 and a block size of 512, which doesn't really need to be changed, and you can probably get away with only a verify pass on it. Check the info sheet that should have come with the drive. However, you must format any new ST412 drive you are hooking up to an Adaptec board.

You can do a physical format at either 256 or 512 bytes per block, so choose which at the prompt.

Next, you must specify either a Seagate or Adaptec format. These

are supported because they are the most common 256-byte systems used by CoCo owners. No attempt is made to support the plethora of other drives out there as they are usually shipped from the factory preformatted for 512 with 1:1 interleave. Usually the Seagate format will work with such drives, but if not, they are proabably usable as is. If you choose the Seagate format, you will have the choice of keeping the mapped out errors intact or not. You will not be able to keep your mapped out errors intact if you are changing the block size of the disk, say from 256 to 512 or vice versa. If you are just changing the interleave or refreshing the drive, you can keep the mapped out errors, so you won't need to do a verify pass every time.

When you format, you will be able to specify the interleave. This is both a science and an art, and is discussed fully in the 'Interleave and Speed' section.

If you picked Adaptec there will be a series of questions you must answer about the drive configuration. See the file, 'adaptec.doc' for more information.

One rule when formatting the drive is to let it 'warm up' for at least half an hour before beginning, to make sure that all of the components of the drive have thermally expanded to their normal operating alignment. This reduces the likelyhood of errors developing after the format.

Once all the necessary parameters are entered, you'll be asked to watch the drive light of the drive you hope you've selected to see if it flashes. If everything is OK, the format will proceed, and will take a few minutes. If the physical format worked, you should then do a verify pass and then a logical format.

Interleave and Speed

For the SCSI System, I recommend an interleave of 1:1 for embedded SCSI drives. Don't be misled by this, it does not mean that the CoCo 3 can sustain the 1:1 thruput of 500K or more per second. It's just that, in the registered version, the cache reads data in blobs of 2K at a time. If the sectors are one right after the other, the drive can buffer them as soon as possible. Even though older SCSI drives don't have a built-in cache, they do have a

buffer which can hold several sectors. On the commonly used Seagate SCSI drives, the buffer is generally 2K long.

Also, with the cache *disabled*, and with just a regular 6809, the registered version can still megaread in 35 seconds if the drive is formatted 1:1. This means that subsequent blocks are processed no later than one complete revolution of the disk (at 3600 rpm). This may sound slow, but compared to older SCSI drivers which took one and a third revolutions to process a single 256-byte block, this is over twice as fast! And that's *without* the turbo cache or the 6309! This demonstrates the huge speed improvement using the accelerated handshake mode of the registered version, over and above the cache and 6309 enhancements.

The shareware version uses the standard handshake mode, which takes a little more than a revolution to process a 512-byte block. This means that a 1:1 interleave drags this out to 2 revolutions per block. But at least the timing is consistent. You can tweak this if you wish, but you'll have to reformat again if you get the registered version.

The exception to the 1:1 rule is the Adaptec SCSI controller. When formatted with an interleave greater than 1, you gain an extra sector per track. Also, the Adaptec buffer is only 1K, so it can't buffer a whole 2K blob at a time. Therefore, the interleave on the Adaptec (or any drive with a buffer known to be less than 2K) should be several sectors. You may have to play with the interleave to find the optimum value.

Also note that the overhead required for the Multipak exacts a small but noticeable speed penalty from the SCSI system, if you must use your Disto controller in one. *Generally* you should get megareads of around 20 seconds with the 6309, and 30 seconds with the 6809, with a Seagate SCSI drive, if you have the registered version. The Adaptec board will probably be slower, while drives with built-in caches (such as Quantum) will usually be faster. At any rate, your mileage may vary.

Verify Pass

The verify pass will find and repair failing blocks on the SCSI drive. It uses the special SCSI 'Verify' instruction to locate bad

blocks much faster than any other OS-9 verifiers, such as 'format' or 'fcheck'. And there is one other bonus. While previous OS-9 verifiers mapped out failing blocks by setting their bit in OS-9 allocation bitmap, 'scsifmt' instead issues a SCSI 'Reassign' instruction to map the failing block to a spare good block on the hard drive. Thus the drive will be absolutely perfect to OS-9, and no bits will need to be mapped out in the bitmap.

The verify pass is also good if you can't or don't want to reformat your new 512-only drive, but you do want to make sure there are no errors on it.

NOTE: The verify pass may be issued seperately at any time after the drive is set up and has files on it. Most drives develop new errors after a while, so you can use 'scsifmt' anytime to keep the drive "perfect". Alternatively, you can just list the failing blocks without doing a reassign, so you can double check which files belong to the failing sectors before mapping the bad blocks out. Either way, it's better and faster than any other disk checker you may have.

NOTE: The verify pass works on SCSI blocks, not OS-9 sectors. So if it's a 512-byte drive, the blocks printed out will have to be multiplied by 2 to equal the OS-9 sector number. Both that sector and the one following it (they both shared the same 512-byte block) will be mapped out. All block numbers are printed in hexadecimal.

Logical Format

After you do a physical format and a verify, you can do a logical format. The logical format sets up the drive with the OS-9 file system and root directory. Unlike the physical format and verify options, the driver and descriptors *must* be in 'OS9Boot' for the logical format to work. This is the case with the OS-9 'format' command as well. The 'scsifmt' Logical Formatter has a number of improvements over the stock OS-9 version. With it, you get additional support for cluster sizes greater than 1. It will also query the drive over the SCSI bus and *ask* it how big it is, so you don't have to fudge the DD.TOT value in LSN0.

You will be prompted for the device name to format, and it will retrieve the format information from the descriptor in memory into

its internal buffer. You will get to see the descriptor parameters before beginning the format. If any of these are wrong, you should fix it with 'scsidesc'. Save the change to the memory descriptor, and make sure that change gets 'cobbler'ed into a a new boot disk. Otherwise your change will be lost when you invoke the Logical Format again, because it was just reload the values from the memory descriptor.

Once all the necessary parameters are entered, you'll be asked to watch the drive light of the drive you hope you've selected to see if it flashes. This way you'll verify that you are actually formatting the correct drive!

Before formatting begins, you will be asked if you want to obtain the exact number of sectors from drive itself (recommended). Either that, or the heads, cylinders and sectors/track values are multiplied together to obtain the number of sectors. The value being used is displayed, and it is rounded off to a cluster boundary to avoid end-of-disk problems. The logical formatter will handle partitions just fine, and you should be able to use clustering in any partition, too.

You will then be asked to continue, and the Logical Format shouldn't take very long, as there is no physical verify, which is unnecessary if you've already done a fast 'scsifmt' verify pass. Once it is finished, the hard drive is ready for use!

8. CLUSTERING AND PARTITIONING

Using a cluster size of 1 sector per cluster, RBF can only handle drives up to about 120 megabytes. This is due to a limit on the size of the allocation bitmap. Other than that, RBF can potentially handle drives up to 4 gigabytes in size.

The SCSI System allows you to get around this limit in one of two ways. You can use a cluster size greater than one, or (new to version 2.2) you can partition the drive into a number of smaller logical drives.

Clustering

Clustering only affects how many sectors are represented by each bit in the allocation bitmap. It does not scale LSNs or DD.TOT in any way. One other difference is that a file must begin on a cluster boundary (that is, the file descriptor starts on a cluster boundary, and the first sector of the file will immediately follow it, if the cluster size is greater than 1). The cluster size has to be a power of 2. Also, the Segment Allocation Size is still in terms of LSNs, but it gets rounded up to a multiple of the cluster size. For example, if the cluster size is 8, and SAS is 11, the the file will actually be allocated in segments of 16 sectors. Clustering may also increase the amount of wasted space if you have a lot of small files. This information was verified by Brian White who experimented with clusters to get to the bottom of things. From his tests, and my dissassembly of RBF, I can say that clustering is handled robustly by the stock RBF, so you don't have to worry about the file system becoming corrupt. Also, the stock OS-9 commands 'dcheck' and 'free' handle clustering OK.

The SCSI System lets you set the cluster size up to 128 sectors. If you are using a disk drive with more than about 120 meg, the cluster size should be increased to 2. At 240 meg, go with 4, at 480, go with 8, etc. Note that 'scsifmt' may only handle up to 2 gigabytes because of the lack of an unsigned long in Microware C. In case you are wondering, the SCSI System driver will use 10-byte extended SCSI read and write commands when the block number won't fit into 21 bits. I was thinking ahead!

UPDATE: It was recently discovered that RBFMan v30 (the one with the 'undel' patches) does not free up all of the appropriate sectors in the bitmap when deleting files at a cluster size greater than 1. This finding, along with people clamoring to use their favorite disk utils that only work with a cluster size of 1, is what prompted me to add disk partitioning to the SCSI System. But if you want to use clustering, use the old RBF until I or someone else figures out how to fix RBFMan v30. The old version appears not to suffer this problem.

Partitioning

Clustering speeds up free space searches on a large drive, and it allows you to use a large drive as one single drive. But many

utilities do not handle cluster sizes greater than 1, such as repack, ba, bd, or fcheck. Also, the latest patched version of ded that prints out the sectors represented by bits in the allocation map does not take into account clustering.

To avoid these problems, disk partitioning has been added to the SCSI System. To keep driver complexity down and to save on system map space, the parititioning is restricted to a fixed size of 262,144 (2^18) sectors per partition, except for the last partition which fills out to the end of the disk. This size represents a 32K allocation bitmap at 1 sector per cluster. Each partition has its own descriptor. Using partitions increases the amount of system map memory used by the RBF drive tables as well as the extra descriptors. The SCSI System is normally compiled to support 8 logical drives so you can have up to 8 partitions, or 536 megabytes.

OS-9 68000 RBFMan
Unfortunately, 512-byte block drives logically formatted under OS-9 6809 are incompatible with the variable block size RBF under OS-9 68000, which does things in a totally different way. Therefore you cannot simply plug an OS-9 68000 512-byte drive into the CoCo and be able to use the files on it normally, or vice versa. A utility could be written to read and write to an OS-9 68000 disk, I suppose.

9. SCSI SYSTEM DEVICE DESCRIPTOR FORMAT

Because of all the features that the SCSI System supports, many of the unused fields of the RBF device descriptors are now used, and some of the old definitions for the fields have been changed.

IT.DRV (offset $13)

This contains the logical drive number of the drive. The logical drive number is independent of the SCSI ID and LUN of the drive. The actual SCSI ID and LUN of the drives are controlled by bits in IT.DNS. This way you can configure the SCSI IDs and LUNs any way you like without having to use a cryptic IT.DRV to access them.

The catch is that you are limited to IDs 0-3 and LUNs 0-3, instead of the usual 0-7 for SCSI. But this shouldn't be a problem. Note that not all CoCo interfaces may be able to drive several SCSI devices, so you may be limited by the hardware to 2 drives. Each partition of a large hard drive must have its own seperate IT.DRV number for correct operation.

IT.STP (offset $14)

Bit	Meaning
7-2	Partition: 0 = no partition, 1-63 partition number
1-0	Slot: Multipak slot number (0-3 => 1-4)

IT.TYP (offset $15)

Bit	Meaning
7	Hard Disk: 1 = hard disk, 0 = floppy disk (set to 1)
6	Fudge LSN0: 0 = OS-9 disk, 1 = non-OS-9 disk #
3-5	Clusters: Cluster Size (0-7 => 1,2,4,8,16,32,64,128)
2	6309 Stack: 0 = None / Power Boost, 1 = NitrOS9
0-1	Sect Sizes: 0 = 256/512, 1 = 1024/2048, 3 = 4096/8192, 3=2336 #

IT.DNS (offset $16)

Bit	Meaning
7	512 Flag: 0 = 256 byte blocks, 1 = 512 byte blocks
5-6	LUN: SCSI Logical Unit Number of drive (0-3)
4	Error Type: 0 = OS9 error, 1 = SCSI extended sense code
3	No Cache: 0 = cache this device, 1 = do not cache *
2	No Turbo: 0 = use accelerated handshaking, 1 = standard *
0-1	SCSI ID: SCSI ID of the drive or controller (0-3)

*: valid for the registered version only
#: reserved for possible non-OS-9 device support (eg CD-ROMs)

10. GETSTAT / SETSTAT CALLS

The SCSI System adds a new SetStat call, SS.SCSI (code 222). This allows you to send any command to the SCSI device and send or

receive any data associated with it, all within the boundaries of OS-9. The format of the call is thus:

Entry: A = path number
B = function code (222)
X = pointer to SCSI command / data packet

Exit: X = number of bytes returned
B = carry set, contains an OS-9 error code
carry not set, contains SCSI status byte returned

The format of the command packet is simply a series of six, ten, or twelve bytes (depends on the command) set up with the desired command bytes. There is no need to specify the command size as it is inherent in the command opcode itself. Imediately following the command bytes is any data bytes that are to be sent to the drive. Again, the number of bytes to be sent is inherent in the format of the data packet, and need not be specified otherwise. If the command does not involve sending any data, don't worry about the bytes after the command packet.

If the command is one which returns data, the data is placed back into the packet pointed to by X at the entry into the call. The command bytes get overwritten. However, X now equals the number of bytes returned, or 0 if there isn't any. Just be sure your buffer can hold all of the data that might be returned.

Commands are not retried, and the sense information is not automatically obtained, by the SS.SCSI call. You will have to look at the SCSI status byte, which is returned in register B, and do your own Request Sense command or retry, as you see fit. If the drive cannot be accessed, an E$Unit error is returned.

The LUN (from the descriptor) of the drive is always masked into the top three bits of the second command byte. Thus you don't have to figure the LUN out for yourself. The bottom 5 bits are passed through unchanged.

Any call to SS.SCSI resets the cache, as it can't know if the command changes the disk drive or not.

To get a path number to use the SS.SCSI call, simply open the desired drive with an '@' appended to the name, such as '/dd@'. Opening in the '@' mode does not initially access the drive.

The speed of data transfers using this call is not paticularly fast, and it would be impractical to use it to perform reads and writes of stored data on a regular basis.

A simple 'scsicmd' util is included which uses this system call to allow you to access the drive and send your own commands to it. This can be used to manually change Mode Select parameters or read them back in, or to manually initiate diagnostic tests. For more information on using 'scsicmd', and a quick reference of SCSI commands that you can use, see the file 'scsicmd.doc'. For complete information, however, you will need the SCSI programming manual for your paticular drive. You must contact the manufacturer and have them send it to you. There may or may not be a charge for it.

The driver also supports the SS.SQD 'park' system call. A 'park' command is included with the SCSI System for you to use, if you need it. Most embedded SCSI drives, such as the Seagate 'N' series, use the spinning discs as a generator when the power fails in order to perform shutdown functions, such as parking, automatically. Thus you can just shut off the power without worrying about parking the drive. Generally, any drive hooked up to the Adaptec board will need to be parked, as they are older ST412 drives.

If the SS.Wtrk (format) call or the SS.Reset (track 0) call is received by SCSISYS, they are ignored. All other SetStat (and GetStat) codes will return with an Unknown Service Request error.

11. PROBLEM CHECKLIST

A controller, interface or cable isn't plugged in correctly, or is loose, or is corroded. Or, the connectors on the cable aren't crimped on correctly or reliably.

The SCSI ID and/or LUN in the descriptor does not match the drive. Or other fields in the descriptor are incorrect, such as the 256/512 flag, Multipak slot, or whatever. Perhaps the combination of jumpers on the controller or drive is the problem.

If you have more than one SCSI drive hooked up, only one of them (no more, no less) should have the SCSI termination resistors installed. Remove the resistor strips from all other devices... and don't lose them! Technically, the resistors should be installed in the device that is the furthest away from the interface. They cut down on ringing, bounce and reflection on the wires at high frequencies.

You do not have 'Shell' and/or 'GrfDrv' in the CMDS directory, or they do not have the correct attributes set. Or you have the wrong '/dd' descriptor, or 'init' or 'cc3go' module in the bootfile, for the setup you have.

Some Seagate drives have a bug whereby they forget that they were reformatted with a new block size. The only solution is new EPROMs from Seagate. Also, the 225N is one of the oldest SCSI drives around, and has a lot of flukes.

Perhaps your drives are running too hot, and need ventilation. Or maybe your power supply can't handle the drives. Or despite the SCSI System's ability to support many drives, your controller may not have enough strength to drive more than 2.

Some people have reported bugs that turned out to be the BLOB (boot list order bug). So you may have to shuffle boot modules around if you are prone to the BLOB.

If you are using a drive other than a Seagate drive, then you may also have problems. I cannot guarantee your non-Seagate drive will work right off the bat, but I can assure you that the SCSI System uses only the standard SCSI command set. In paticular, a number of people have reported that they are having no problems using Quantum and Maxtor drives with the SCSI System.

Also, you may need to configure the jumpers on your drive to get it to work properly, and I really can't tell you about the jumpers on

your paticular drive. However, if you don't have the manual or information sheet for your drive, a rule of of thumb is to pull off all of the jumpers, and that usually sets the drive for SCSI ID 0 with the parity disabled.

12. ERRORS

NOTE: See the Error Codes on pages 26-27.

Most SCSI devices support the extended sense code which reveals the exact nature of the error that occured. While these error codes are useful to figure out what the problem is, they have no correlation to OS-9 error codes. The SCSI system, via the Error Type bit, supports conversion of the SCSI error into a more meaningful OS-9 error.

These error codes are fairly standard across a broad range of drives. The codes here are from the Seagate SCSI manual, except for those marked with a '*', which are from some other manuals I have. Those marked with a '#' are those where the drive is at fault, and not the SCSI System or the user (although this is not carved in stone). If the OS-9 Equivalent in the list is blank, then SCSISYS converts it to E$Unit by default.

13. LAST RESORT

The SCSI System has been designed to work with a wide variety of hardware and CoCo setups, but it is bound to hit a stumbling block here or there because I can't possibly test all known hard drives, computers, interfaces and controllers! So if you find a bug or problem, you may contact me, the author, Matthew Thompson, in any of the following ways, and I'll see if I can help you.

TECHNICAL AND ORDER ASSISTANCE

Fidonet: Colin McKay @ 1:163/306

Telephone: (613)736-0329
Tuesdays, 8-11PM EST are best.

Postal: Northern Xposure
7 Greenboro Cres
OTTAWA ON K1T 1W6
Canada

SCSI	OS-9	Specific Error	Equivalent	Problem
$00				nothing
$01	246	E$NotRdy	#	no index pulse
$02	247	E$Seek	#	no seek complete
$03	245	E$Write	#	write error
$04	246	E$NotRdy	#	drive not ready
$05				*drive not selected
$06	247	E$Seek	#	no TK00 found
$07				*multiple drives selected
$08				#*logical unit comm error
$09				#*track following error
$10	243	E$CRC	#	id CRC error
$11	244	E$Read	#	unrecovered read error of data
$12	244	E$Read	#	address mark not found in ID field
$13	244	E$Read	#	address mark not found in data field
$14	247	E$Seek	#	record not found
$15	247	E$Seek	#	seek error
$16				#*data sync mark error
$17	243	E$CRC	#	recovered using retries
$18	243	E$CRC	#	recovered using ECC
$19	187	E$IllArg		defect list error
$1A	187	E$IllArg		parameter overrun
$1C	244	E$Read	#	primary defect list not found
$1E	243	E$CRC	#	recovered ID with ECC
$20	192	E$IllCmd		illegal command opcode
$21	241	E$Sect		illegal block address
$22				*illegal function for device type
$23				*volume overflow
$24	241	E$Sect		illegal arg in CDB
$25	187	E$IllArg		invalid LUN
$26	187	E$IllArg		invalid field in param list
$29	246	E$NotRdy		power on or media change occured (ignoreable)
$2A	246	E$NotRdy		mode select parameters changed (ignorable)

SCSI	OS-9	Specific Error	Equivalent	Problem
$31				*format failed (could be your or drive's fault)
$32			#	no more room for reassigned blocks
$40			#	RAM failure
$41			#*	ECC diagnostic error
$42			#	power-on diagnostic failure
$43				*message reject
$44			#	internal controller error
$45				select/reselect failure
$46			#	unsuccessful soft reset
$47				parity error (parity jumper should be removed)
$48				initiator detected error
$49				*illegal message
$80-$FF			#	power-up diagnostic error
other	240	E$Unit		some other error

This page intentionally left blank.