SuperDriver 2.1 User Manual

The Ultimate SCSI/IDE Solution for NitrOS-9!

Manual Revision 1.2

Cloud-9
3749 County Road 30
Delano, MN 55328
http://www.cloud9tech.com/

Introduction

Congratulations on purchasing SuperDriver, another one of the fine products offered by Cloud-9. This product was designed to be used with the Radio Shack TRS-80 Color Computer 1, 2, and the Tandy Color Computer 3.

SuperDriver gives your Color Computer access to high-performance IDE and SCSI hard drives and CD-ROM drives under NitrOS-9 when used with Cloud-9's TC^3 SCSI Controller or SuperIDE Interface (the Glenside IDE interface is also supported). Using an advanced split-level driver architecture, you can run both IDE and SCSI drives together on your system with minimal impact on system memory.

Features

- Full support for SCSI hard drives and CD-ROMs, including SCSI IDs 0-6
- Master/Slave IDE support for both ATA and ATAPI devices
- Zero configuration, intelligent drivers
- Smart caching grows the cache on an as-needed basis
- Split driver architecture saves memory and supports different controllers efficiently
- Full support for all versions of NitrOS-9
- Read and write support for 256, 512, 1024 and 2048 byte sector devices
- 24-bit sector addressing supports up to 4GB
- Partitioning support for larger drives
- Access to HDB-DOS virtual drives from NitrOS-9
- Support for raw SCSI commands through the SS.DCmd SetStat call
- Additional size and speed optimizations
- Better device timeout support for ATAPI devices
- Robust error reporting for SCSI devices
- SCSI target-initiated messages are handled, resulting in additional supported drives
- IDE and SCSI boot modules can now read fragmented boot files

New Features/Bug Fixes in 2.1

- Descriptors for the SuperBoard SCSI Controller are included
- New utilities to test aspects of the SCSI system are included
- A bug in the low level SCSI driver affecting SS.DCmd calls has been fixed
- A bug affecting access to HDB-DOS partition drives has been fixed
- A bug affecting the operation of the SCSI and IDE booters has been fixed
- RBSuper no longer crashes if the low level module in the descriptor is not in memory
- An oversight in the /S0 descriptor included in the bootfile in SuperDriver 2.0 has been addressed

System Requirements

For NitrOS-9 Level 1 systems, SuperDriver requires:

- TRS-80/Tandy Color Computer 1 or 2 with 64K of RAM.
- Television.

For NitrOS-9 Level 2 systems, SuperDriver requirements are:

- Tandy Color Computer 3.
- RGB or Monochrome Monitor.

Additional required hardware:

- Multi-Pak Interface.
- Floppy disk controller in slot 4 of the Multi-Pak connected to one or more floppy drives.
- TC^3 SCSI Controller or SuperIDE Interface in any slot of the Multi-Pak
- NitrOS-9 for your Color Computer (see http://www.nitros9.org or contact Cloud-9).

Starting From Scratch

Before SuperDriver, setting up a hard drive system on the Color Computer was a daunting prospect for OS-9 users. Many found the installation and configuration of a hard drive system perplexing and frustrating, mainly because the steps are many, must be done in the proper order, and must be performed without deviation.

A good deal of the complexity in setting up a hard drive came from the fact that a bootfile had to be built with the proper driver and descriptors. Additionally, certain commands needed to be present on the boot disk in order to format the hard drive and provide a minimal set of tools.

That work has largely been alleviated with the advent of SuperDriver. Cloud-9 has worked hard to insure that you get your system up as quickly as possible by providing you with a bootable diskette containing both SCSI and IDE drivers and descriptors as well as commonly used utilities. (Bootable on a Color Computer 3 only).

Selecting a Hard Drive

Which hard drive should you use for your system? Today, multi-megabyte and gigabyte hard drives can be had for very good prices. Often, though, the space that these hard drives provide is considerably more than a typical NitrOS-9 system will ever need. While the RBF file manager can handle a drive size of 4 gigabytes, such space on a CoCo is absurdly huge.

In fact, many NitrOS-9 users will never need more than 20 megabytes. If you don't already have a hard drive, consider selecting one with a size range between 40 – 200 megabytes. Drives such as the ZIP 250 make a good choice for a CoCo's hard drive system. If you are considering CompactFlash as a storage medium, 128MB is a recommended size.

Setting Up The Hardware

Once you've obtained one or more hard drives, you will need to set them up. Here are a few pointers that you should follow for your devices:

1. For SCSI drives, each hard drive or CD-ROM should be configured to have a unique SCSI ID between 0 and 6. No two drives should have the same SCSI ID. For IDE drives, be sure that the master and slave settings are set properly.

2. If you are using SCSI, pay attention to cabling and termination. This is VERY IMPORTANT! Be sure that all cables are secure, and that the SCSI device on the very end of the cable is properly terminated. No other drives between the last SCSI device and the controller should have their termination jumpers set. Consult your drive's manual on specific steps to insure proper termination.

Now write down the SCSI or IDE device ID for each drive. No two drives can share the same ID, so make sure each one is unique.

Once your hard drives have been connected and properly terminated, you need to connect your TC^3 SCSI Controller or SuperIDE Interface to the Color Computer. Both can be placed in any slot of the Multi-Pak, though you should reserve slot 4 for any floppy disk controller you may have.

Powering Up The System

Once everything has been properly connected, power up your Color Computer, Multi-Pak and drives AT THE SAME TIME. A power multi-strip is extremely handy for this and is highly recommended. Depending upon the setup, turning on your computer and drives at separate times can cause startup problems. Save yourself the hassle and start all components up together.

Setting Up Your System

The traditional approach to configuring and using hard drive system under OS-9 for the Color Computer has been the following:

1. Create device descriptors with the appropriate cylinder, side and sector values for each of the hard disks on your system.

2. Build a floppy boot disk that contained a hard disk driver and descriptors for each hard drive.

3. Boot from that floppy then format the hard drive(s).

4. Copy the contents of the System Master (commands, support files, etc.) onto the hard drive.

5. Build a second floppy boot disk whose DD device was a copy of the main hard drive's descriptor. Booting from this floppy caused OS-9 to use the hard drive once the bootfile was loaded from floppy.

Each of these steps contained additional sub-steps, making the process cumbersome and overwhelming for the average OS-9 user.

The design of SuperDriver has made the old "from scratch" configuration of a hard drive system obsolete by providing a *bootable* distribution disk containing descriptors for both the TC^3 SCSI Controller and SuperIDE Interface. One descriptor exists for each possible device: `s0-s6` for 7 SCSI devices, and `i0-i1` for master/slave IDE devices. In addition, `sh` and `ih` descriptors are included to access an HDB-DOS partition under SCSI or IDE, respectively.

Please Backup First!

Don't make the mistake of using your SuperDriver distribution disk. Please backup your disk and keep the master in a safe place. ONLY WORK FROM A COPY OF YOUR DISK!

Booting from the Distribution Disk

A Color Computer 3 is required to boot from the SuperDriver distribution disk. To begin, simply insert a copy of the disk into drive 0. From Disk BASIC, type DOS and press ENTER. NitrOS-9 will boot up and present you with a shell. You will now want to refer to the notes that you took earlier, describing each SCSI or IDE drive that you have connected.

Formatting the SCSI Hard Drives

It is time to format all of the SCSI hard drives that are connected, presuming that they do not have important data that you wish to keep. For each device in your list, you can type:

```
format /sx
```

Where *x* is the SCSI ID of the hard drive you wish to format. Of course, CD-ROMs cannot be formatted. Simply ignore them at this point.

Format will automatically query the drive for its size and format the drive to its maximum capacity. Additionally, the version of the format utility that comes with SuperDriver will automatically determine the cluster size, so you do not have to pre-compute this value.

Formatting the IDE Hard Drives

Much like the SCSI drives above, only format the IDE drives that do not already contain important data. For each device in your list, you can type:

```
format /ix
```

Where *x* is the IDE ID of the hard drive you wish to format For the master drive, use 0. For the slave drive, use 1. Again, format will automatically query the drive for its size and format the drive to its maximum capacity.

Copying Files

Now that the hard drives are formatted, you are ready to copy files to it. Select one of the drives to be your main hard drive (usually /s0 or /i0). If you have a NitrOS-9 System Master disk, insert it in the floppy drive and copy it to one of your hard drives:

```
chd /d0; chx /d0/cmds
dsave /s0 ! shell -p
```

This will copy all files from your floppy to the selected hard drive.

Advanced Customization

Choosing the Right Driver

SuperDriver provides three different sets of drivers: one set for NitrOS-9/6809 Level 1, another set for NitrOS-9/6809 Level 2 and another set for NitrOS-9/6309 Level 2. Be sure to use the appropriate set of modules for the version of NItrOS-9 that you will be using.

"Floppy-Less" Booting with HDB-DOS

SuperDriver co-exists nicely with another one of Cloud-9's products: HDB-DOS. With HDB-DOS it is possible to boot into NitrOS-9 completely from the hard drive, without the need of a floppy. This requires that HDB-DOS and NitrOS-9 reside on the same hard drive with a process called *partitioning*.

The first step is to run WIZARD.BAS, a program that comes on the HDB-DOS disk. It will help you configure HDB-DOS to co-exist with NitrOS-9 on the same hard drive, and provide you with information helpful in customizing the NitrOS-9 device descriptor for your hard drive.

Next, you will need to build a 35 track single-sided boot disk with either *boot_tc3* or *boot_ide* located on track 34, depending on whether you are using the TC^3 SCSI Controller or the SuperIDE Interface. The *mb* script found in NITROS9/6?09L?/SCRIPTS of the NitrOS-9 Modules disk can be modified to use the appropriate boot module.

Once you have built the boot disk with the alternate boot module, then:

1. Reset into BASIC and backup the newly created floppy to a virtual hard drive of your choice.

2. Run the LINK.BAS program that comes with HDB-DOS and type in the virtual hard drive number that you backed up your boot floppy to.

3. From that point on you can type `DOS x` (where x is the virtual drive number of the boot floppy, which now resides on the hard drive) and boot directly into NitrOS-9 without the need for a boot floppy.

Note: If you wish to change the SCSI/IDE ID that the boot module uses to find the OS9Boot file, modify the offset at the end of the boot module minus 4 bytes (0-6 for boot_tc3 or 0-1 for boot_ide). Also, you can change the base address of the controller, located at the end of the boot module minus 6 bytes. For your convenience, you can use the `ded` utility that comes with SuperDriver to change these bytes.

RBSuper Features

RBSuper is the intelligent component of the driver interface, and is a relatively complex piece of software. This section describes the features of this powerful driver.

Split Driver Architecture

One of the most advanced features of RBSuper is its delineation between 'high level' driver tasks (file manager interaction, cache management, etc.) and 'low level' driver tasks (talking to a specific controller or interface to read/write sectors). This split in the driver is reflected in the high-level driver *rbsuper* and the low-level drivers (*lltc3* or *llide*) included in the bootfile.

This architecture has several advantages:

- **Simplified driver development.** The complicated tasks of cache management and sector deblocking are handled by the high level driver. Low-level drivers only need to concern themselves with reading and writing physical sectors on the device it supports.

- **Conservation of system memory.** System RAM is conserved when using two or more low level drivers, since the high level driver contains a great deal of common code that would normally be duplicated in drivers for different controllers.

Sector Deblocking

Although RBF and many disk applications were written for 256 byte sector devices, virtually all SCSI and IDE drives available today have 512 byte sectors. In order to utilize the full capacity of these drives and still give RBF the impression that sectors are 256 bytes in size, RBSuper uses a method called deblocking. Deblocking involves RBSuper reading sectors in the drive's native size and returning the appropriate 256 byte sector to RBF. While deblocking can be slower on writes due to the extra reading that is necessary, the speed penalty is less severe due to caching.

Smart Caching

In order to improve disk performance and handle drives with larger sector sizes, RBSuper employs smart caching. Through smart caching, the cache can grow up to the maximum 2048 bytes to accommodate CD-ROM drives. However, in order to save system memory, RBSuper will only allocate the amount of cache needed for the largest sector sized device that comes online.

For example, assume that you have two SCSI devices on your system: a hard drive that has 512 byte sectors, and a CD-ROM that has 2048 byte sectors. As long as you use the hard drive, your cache size will only be 512 bytes, matching the sector size of the hard drive and

saving system memory. As soon as the CD-ROM is accessed, the cache size increases to 2048 bytes to accommodate the larger sector size of the CD-ROM. Although more system memory is used, the 2048 byte cache is now utilized by the 512 byte sector device to load in more sectors at read time, thereby increasing performance.

Note that the cache is allocated on a *per controller* basis and shared among devices on that controller. Therefore, if you have both an IDE and a SCSI controller on the same system, then SCSI devices will share one cache, while IDE devices share another cache.

Full IDE Support

In the IDE world, there are two main types of drives: ATA and ATAPI. In the past, IDE support for OS-9 has consisted of ATA only drivers, leaving out a large number of ATAPI devices such as CD-ROMs, ZIP drives and LS-120's. Now, SuperDriver supports both ATA and ATAPI devices automatically – no special device descriptor configuration is required. Additionally, ATA devices are automatically detected as either CHS or LBA devices, further improving performance wherever possible.

Turbo Mode (SCSI only)

If you are using very fast SCSI drives, turbo mode can significantly increase the speed of data transfers by eliminating certain handshaking steps during the SCSI transfer. Only faster hard drives can handle this feature; slower hard drives may exhibit data corruption problems or not work at all. We recommend carefully testing each drive for its ability to perform in turbo mode.

To turn on turbo mode for a SCSI descriptor, bit #4 of the IT.DNS byte must be set. For example, if the IT.DNS value of /s0 is $00, then you should set the new value to $10 to turn on turbo mode:

```
dmode /s0 dns=10
```

Automatic Drive Size Query

NitrOS-9 Version 03.02.02 and later supports automatic drive size querying through the SS.DSize GetStat call. This feature allows drivers to return the size of a drive without having to modify the device descriptor. Both low level SCSI and IDE drivers support the SS.DSize GetStat, and can return drive size information.

The format program will attempt to call the SS.DSize call, but will only do so if a certain bit in the IT.TYP field of the descriptor is set. If the bit is not set, then format will use the IT.CYL, IT.SID and IT.SCT values in the device descriptor to determine the drive's size.

By default, the device descriptors that come with SuperDriver have the drive size query bit (bit #4) in IT.TYP set to 1. You can use dmode to turn the bit on or off.

Write Verification

If the IT.VFY field in the device descriptor is set to 0, then write verification is enabled. With this feature, RBSuper reads back every sector that it writes to the hardware and compares it to the originally written sector. To turn on write verification, use the following command (using /s0 as an example):

```
dmode /s0 vfy=0
```

Clustering

Clustering is RBF's method of dealing with larger (greater than 128MB) hard drives due to limits imposed on the bitmap size.

Format will automatically determine the best cluster size for your drive; however, you can manually specify a cluster size by enclosing the value in slashes (e.g. /2/ for a cluster size of two). The following table shows recommended cluster sizes for various size hard drives:

Hard Drive Size (in bytes)	Cluster Size
0-134,215,679	1
134,215,680-268,431,359	2
268,431,360-536,862,719	4
536,826,720-1,073,725,439	8
1,073,725,439-2,147,450,879	16
2,147,450,880-4,294,901,759	32

Partitioning

While clustering is an acceptable and sometimes even desirable way of utilizing large hard drives, there are disk utilities from 3rd parties that do not properly handle clustering, and can actually compromise your hard drive's integrity if they are run.

Partitioning is a useful means of managing a large amount of hard drive space without resorting to clustering. It works by breaking up a hard drive in multiple logical devices while avoiding the problems associated with clustering and some 3rd party utilities. A 24-bit **LSN offset** can be specified in the device descriptor, which is then added to any access to that device. For example, /S0 may have a SCSI ID of 0 and an LSN offset of zero, while /S1, also with a SCSI ID of 0, may have an LSN offset of $018A00. Although both descriptors have different names, they point to the same physical hard drive.

In the case of /S1, the value $018A00 is added to any physical sector that is accessed. This means that a request for physical sector 0 on /S1 will return the actual sector $018A00. Requesting physical sector 1 on /S1 will return actual sector $018A01, and so on.

Presuming that /S0 is formatted for $018A00 sectors, both partitions can co-exist on the same hard drive without interfering with each other.

When setting up different device descriptors for multiple partitions, each device descriptor should have its own unique logical drive number. Do NOT allow two device descriptors share the same logical drive number if they have different LSN offsets.

The following command will set the LSN offset of descriptor /S1 to $018A00:

```
dmode /s1 wpc=01 ofs=8a00
```

Auto drive size querying should be turned OFF for descriptors that are used for partitioning. If not, the format program will format the entire drive and not just the partition.

HDB-DOS Virtual Disk Support

If you are using HDB-DOS on your system and have dedicated a portion of your hard drive to setting up virtual disks, you can now access those virtual disks from NitrOS-9 with a special HDB-DOS device descriptor.

The HDB-DOS descriptor flags the RBSuper driver to ignore data beyond the first 256 bytes of a sector, since HDB-DOS doesn't fully support sector sizes greater than 256 bytes. This feature is turned on by setting bit #3 of the IT.DNS field in the descriptor. In addition to this bit being set, the IT.STP field is set to the value of the HDB-DOS drive to be accessed (0-255). Finally, the LSN offset in the device descriptor should match the "HDB-DOS Offset" in the HDB-DOS ROM, and the logical drive number should be a unique value, as noted above.

The `sh` and `ih` device descriptors in your bootfile are already set up for SCSI and IDE HDB-DOS partition access. Both access virtual drive 255 with an LSN offset of $018A00. Feel free to use them as a starting point for your own HDB-DOS device descriptors.

Writing a Low-Level Driver

RBSuper's driver architecture represents a new way of writing RBF device drivers, especially for hard drives that have sector sizes that are larger than 256 bytes.

By writing your RBF driver to RBSuper's rules, you get the immediate benefits of caching, sector size handling and write verification support. The complexity of writing your driver is significantly reduced, and you can focus on the details of communicating with your specific controller or interface.

The distribution disk contains three files in the SRC directory that will get you started writing your own super driver:

- llex.asm
- rbsuper.d
- superdesc.asm

llex.asm is a low level skeleton driver. It doesn't do anything useful, but it can be used as a starting point for your own driver development.

rbsuper.d contains static memory definitions used by RBSuper and low level drivers, and is included by the llex.asm source file.

superdesc.asm is the source file for an RBSuper descriptor.

Entry Points

Like the traditional RBF driver, the RBSuper driver also has 6 defined entry points:

ll_init

This is the initialization entry point into the low level driver. Your code should do whatever is necessary to prepare the hardware for use.

Upon entry, the Y and U register points to the device descriptor and driver static memory, respectively.

It is important to note that this entry point is called once PER controller, NOT per device. This means if you have 5 devices attached to your controller, ll_init will only get called the first time one of the devices is initialized. As the other devices come on line, they will NOT cause ll_init to be called.

ll_term

This is the termination entry point into the low level driver. Your code should do whatever is necessary to prepare the hardware for shutdown.

Like ll_init, this entry point is called once PER controller, NOT per device. This means that ll_term will only get called on the last device to shut down.

Upon entry, the Y and U register points to the device descriptor and driver static memory, respectively.

ll_read

This is the read entry point into the low level driver. Like an RBF driver, registers Y and U point to the path descriptor and driver static memory. Unlike RBF's read entry point, however, ll_read pulls LSN and other information from static variables that have been defined by RBSuper.

- V.PhysSect: the 24 bit value which represents the PHYSICAL sector to be read.
- V.SectSize: the physical size of the sector (0 = 256 bytes/sector, 1 = 512, 2 = 1024, 3 = 2048)
- V.SectCnt: the number of physical sectors to read
- V.CchPSpot: the position in the cache to move the read sector data into.

Note that this routine deals with PHYSICAL sectors. That is, the sector size that is native to the device. Usually the physical sector size is 512, but can be as large as 2048, in the case of CD-ROMs. The V.SectSize value indicates the physical sector size.

All registers may be modified, but static variables must remain unchanged.

ll_write

This is the write entry point into the low level driver. Like ll_read, Y and U point to the path descriptor and driver static memory. It uses the same static variables, except the sector data at V.CchPSpot is written to the drive.

ll_getstat/ll_setstat

These are the GetStat and SetStat entry points into the low-level driver. The stat code is passed in the caller's B register, and the U register points to the driver's static memory.

Device Descriptor

RBSuper's device descriptor is virtually the same as existing RBF descriptors. However, an additional two byte field is present. This field is an offset from the start of the descriptor to a string, which holds the name of the low level driver.

The superdesc.asm source file is set up to reference llex, the example low level driver.

Direct Hardware Access with SS.DCmd

SuperDriver allows direct access to SCSI devices through the SS.DCmd (Direct Command) NitrOS-9 SetStat system call. This call allows you to send SCSI commands to any SCSI device by just opening a path to that device and setting CPU registers to point to an appropriate SCSI packet and a transfer buffer.

The following 6809 assembly language code segment demonstrates how to use the SS.DCmd call to read the first sector of the device /S0:

```
txbuf     rmb  2048

Device    fcs  "/S0@"
ReadLSN0  fcb  $08,$00,$00,$00,$01,$00

          leax Device,pcr
          lda  #READ.
          os9  I$Open

* Register B = SS.DCmd
* Register X = Transfer buffer
* Register Y = SCSI command packet
          ldb  #SS.DCmd
          leax txbuff,u
          leay ReadLSN0,pcr
          os9  I$SetStt
```

The SCSI device descriptor must have @ appended, indicating to NitrOS-9 that the device is to be opened in raw mode. Once the path to the raw device has been obtained, the X register is set to point to the transfer buffer, the Y register is set to point to the SCSI packet that will be sent to the controller, and the B register will contain the value SS.DCmd.

Upon completion of the DCmd call, any data returned will be placed in the memory location pointed to by the X register. Also, register A will hold the SCSI status byte that the controller returned in response to the command.

Note: Only the super user (user 0) may use the SS.DCmd call. Any other user will receive an E$FNA (error 214) upon making the call.

Certain SCSI commands like START UNIT and STOP UNIT do not return data, so the X register can be ignored.

Included Utilities

SuperDriver now includes several utilities in the CMDS directory that demonstrate the SCSI driver's SS.DCmd SetStat call:

- scsitest – Sends a UNIT START, READ, UNIT STOP and UNIT START command to a SCSI device.

- scsiquery – Sends INQUIRY and READ CAPACITY commands to a SCSI device and reports back information about the device including vendor information, production identification, block count and block size.

Each utility requires the name of the device that will be accessed (i.e. scsitest /s0)
Source code for these utilities is included in the SRC directory. You may wish to study them for writing your own commands utilizing the SS.DCmd call.

Technical Support

Cloud-9 is committed to creating quality hardware and software products to the Color Computer market. If you have access to the Internet, you can usually find answers to common questions by checking the support section of our website. If you encounter issues with your product, please send an email to us and we will answer it as quickly as possible.

E-mail: support@cloud9tech.com

Web: http://www.cloud9tech.com/

Snail Mail:

Cloud-9
3749 County Road 30
Delano, MN 55328