Storage Devices

Wilko Bulte

wilko@FreeBSD.org

$FreeBSD: doc/en_US.ISO8859-1/articles/storage-devices/article.sgml,v 1.15 2004/11/29 21:43:34 ceri Exp $

This article talks about storage devices with FreeBSD.

1 Using ESDI hard disks

ESDI is an acronym that means Enhanced Small Device Interface. It is loosely based on the good old ST506/412 interface originally devised by Seagate Technology, the makers of the first affordable 5.25" winchester disk.

The acronym says Enhanced, and rightly so. In the first place the speed of the interface is higher, 10 or 15 Mbits/second instead of the 5 Mbits/second of ST412 interfaced drives. Secondly some higher level commands are added, making the ESDI interface somewhat “smarter” to the operating system driver writers. It is by no means as smart as SCSI by the way. ESDI is standardized by ANSI.

Capacities of the drives are boosted by putting more sectors on each track. Typical is 35 sectors per track, high capacity drives I have seen were up to 54 sectors/track.

Although ESDI has been largely obsoleted by IDE and SCSI interfaces, the availability of free or cheap surplus drives makes them ideal for low (or now) budget systems.

1.1 Concepts of ESDI

1.1.1 Physical connections

The ESDI interface uses two cables connected to each drive. One cable is a 34 pin flat cable edge connector that carries the command and status signals from the controller to the drive and vice-versa. The command cable is daisy chained between all the drives. So, it forms a bus onto which all drives are connected.

The second cable is a 20 pin flat cable edge connector that carries the data to and from the drive. This cable is radially connected, so each drive has its own direct connection to the controller.

To the best of my knowledge PC ESDI controllers are limited to using a maximum of 2 drives per controller. This is compatibility feature(?) left over from the WD1003 standard that reserves only a single bit for device addressing.

1.1.2 Device addressing

On each command cable a maximum of 7 devices and 1 controller can be present. To enable the controller to uniquely identify which drive it addresses, each ESDI device is equipped with jumpers or switches to select the devices address.

On PC type controllers the first drive is set to address 0, the second disk to address 1. *Always make sure* you set each disk to an unique address! So, on a PC with its two drives/controller maximum the first drive is drive 0, the second is drive 1.

1.1.3 Termination

The daisy chained command cable (the 34 pin cable remember?) needs to be terminated at the last drive on the chain. For this purpose ESDI drives come with a termination resistor network that can be removed or disabled by a jumper when it is not used.

So, one and *only* one drive, the one at the farthest end of the command cable has its terminator installed/enabled. The controller automatically terminates the other end of the cable. Please note that this implies that the controller must be at one end of the cable and *not* in the middle.

1.2 Using ESDI disks with FreeBSD

Why is ESDI such a pain to get working in the first place?

People who tried ESDI disks with FreeBSD are known to have developed a profound sense of frustration. A combination of factors works against you to produce effects that are hard to understand when you have never seen them before.

This has also led to the popular legend ESDI and FreeBSD is a plain NO-GO. The following sections try to list all the pitfalls and solutions.

1.2.1 ESDI speed variants

As briefly mentioned before, ESDI comes in two speed flavors. The older drives and controllers use a 10 Mbits/second data transfer rate. Newer stuff uses 15 Mbits/second.

It is not hard to imagine that 15 Mbits/second drive cause problems on controllers laid out for 10 Mbits/second. As always, consult your controller *and* drive documentation to see if things match.

1.2.2 Stay on track

Mainstream ESDI drives use 34 to 36 sectors per track. Most (older) controllers cannot handle more than this number of sectors. Newer, higher capacity, drives use higher numbers of sectors per track. For instance, I own a 670 MB drive that has 54 sectors per track.

In my case, the controller could not handle this number of sectors. It proved to work well except that it only used 35 sectors on each track. This meant losing a lot of disk space.

Once again, check the documentation of your hardware for more info. Going out-of-spec like in the example might or might not work. Give it a try or get another more capable controller.

1.2.3 Hard or soft sectoring

Most ESDI drives allow hard or soft sectoring to be selected using a jumper. Hard sectoring means that the drive will produce a sector pulse on the start of each new sector. The controller uses this pulse to tell when it should start to write or read.

Hard sectoring allows a selection of sector size (normally 256, 512 or 1024 bytes per formatted sector). FreeBSD uses 512 byte sectors. The number of sectors per track also varies while still using the same number of bytes per formatted sector. The number of *unformatted* bytes per sector varies, dependent on your controller it needs more or less overhead bytes to work correctly. Pushing more sectors on a track of course gives you more usable space, but might give problems if your controller needs more bytes than the drive offers.

In case of soft sectoring, the controller itself determines where to start/stop reading or writing. For ESDI hard sectoring is the default (at least on everything I came across). I never felt the urge to try soft sectoring.

In general, experiment with sector settings before you install FreeBSD because you need to re-run the low-level format after each change.

1.2.4 Low level formatting

ESDI drives need to be low level formatted before they are usable. A reformat is needed whenever you figgle with the number of sectors/track jumpers or the physical orientation of the drive (horizontal, vertical). So, first think, then format. The format time must not be underestimated, for big disks it can take hours.

After a low level format, a surface scan is done to find and flag bad sectors. Most disks have a manufacturer bad block list listed on a piece of paper or adhesive sticker. In addition, on most disks the list is also written onto the disk. Please use the manufacturer's list. It is much easier to remap a defect now than after FreeBSD is installed.

Stay away from low-level formatters that mark all sectors of a track as bad as soon as they find one bad sector. Not only does this waste space, it also and more importantly causes you grief with bad144 (see the section on bad144).

1.2.5 Translations

Translations, although not exclusively a ESDI-only problem, might give you real trouble. Translations come in multiple flavors. Most of them have in common that they attempt to work around the limitations posed upon disk geometries by the original IBM PC/AT design (thanks IBM!).

First of all there is the (in)famous 1024 cylinder limit. For a system to be able to boot, the stuff (whatever operating system) must be in the first 1024 cylinders of a disk. Only 10 bits are available to encode the cylinder number. For the number of sectors the limit is 64 (0-63). When you combine the 1024 cylinder limit with the 16 head limit (also a design feature) you max out at fairly limited disk sizes.

To work around this problem, the manufacturers of ESDI PC controllers added a BIOS prom extension on their boards. This BIOS extension handles disk I/O for booting (and for some operating systems *all* disk I/O) by using translation. For instance, a big drive might be presented to the system as having 32 heads and 64 sectors/track. The result is that the number of cylinders is reduced to something below 1024 and is therefore usable by the system

without problems. It is noteworthy to know that FreeBSD does not use the BIOS after its kernel has started. More on this later.

A second reason for translations is the fact that most older system BIOSes could only handle drives with 17 sectors per track (the old ST412 standard). Newer system BIOSes usually have a user-defined drive type (in most cases this is drive type 47).

> **Warning:** Whatever you do to translations after reading this document, keep in mind that if you have multiple operating systems on the same disk, all must use the same translation

While on the subject of translations, I have seen one controller type (but there are probably more like this) offer the option to logically split a drive in multiple partitions as a BIOS option. I had select 1 drive == 1 partition because this controller wrote this info onto the disk. On power-up it read the info and presented itself to the system based on the info from the disk.

1.2.6 Spare sectoring

Most ESDI controllers offer the possibility to remap bad sectors. During/after the low-level format of the disk bad sectors are marked as such, and a replacement sector is put in place (logically of course) of the bad one.

In most cases the remapping is done by using N-1 sectors on each track for actual data storage, and sector N itself is the spare sector. N is the total number of sectors physically available on the track. The idea behind this is that the operating system sees a "perfect" disk without bad sectors. In the case of FreeBSD this concept is not usable.

The problem is that the translation from *bad* to *good* is performed by the BIOS of the ESDI controller. FreeBSD, being a true 32 bit operating system, does not use the BIOS after it has been booted. Instead, it has device drivers that talk directly to the hardware.

So: do not use spare sectoring, bad block remapping or whatever it may be called by the controller manufacturer when you want to use the disk for FreeBSD.

1.2.7 Bad block handling

The preceding section leaves us with a problem. The controller's bad block handling is not usable and still FreeBSD's filesystems assume perfect media without any flaws. To solve this problem, FreeBSD use the `bad144` tool. Bad144 (named after a Digital Equipment standard for bad block handling) scans a FreeBSD slice for bad blocks. Having found these bad blocks, it writes a table with the offending block numbers to the end of the FreeBSD slice.

When the disk is in operation, the disk accesses are checked against the table read from the disk. Whenever a block number is requested that is in the `bad144` list, a replacement block (also from the end of the FreeBSD slice) is used. In this way, the `bad144` replacement scheme presents "perfect" media to the FreeBSD filesystems.

There are a number of potential pitfalls associated with the use of `bad144`. First of all, the slice cannot have more than 126 bad sectors. If your drive has a high number of bad sectors, you might need to divide it into multiple FreeBSD slices each containing less than 126 bad sectors. Stay away from low-level format programs that mark *every* sector of a track as bad when they find a flaw on the track. As you can imagine, the 126 limit is quickly reached when the low-level format is done this way.

Second, if the slice contains the root filesystem, the slice should be within the 1024 cylinder BIOS limit. During the boot process the bad144 list is read using the BIOS and this only succeeds when the list is within the 1024 cylinder limit.

> **Note:** The restriction is not that only the root *filesystem* must be within the 1024 cylinder limit, but rather the entire *slice* that contains the root filesystem.

1.2.8 Kernel configuration

ESDI disks are handled by the same `wd`driver as IDE and ST412 MFM disks. The `wd` driver should work for all WD1003 compatible interfaces.

Most hardware is jumperable for one of two different I/O address ranges and IRQ lines. This allows you to have two wd type controllers in one system.

When your hardware allows non-standard strappings, you can use these with FreeBSD as long as you enter the correct info into the kernel config file. An example from the kernel config file (they live in `/sys/i386/conf` BTW).

```
# First WD compatible controller
controller      wdc0    at isa? port "IO_WD1" bio irq 14 vector wdintr
disk            wd0     at wdc0 drive 0
disk            wd1     at wdc0 drive 1
# Second WD compatible controller
controller      wdc1    at isa? port "IO_WD2" bio irq 15 vector wdintr
disk            wd2     at wdc1 drive 0
disk            wd3     at wdc1 drive 1
```

1.3 Particulars on ESDI hardware

1.3.1 Adaptec 2320 controllers

I successfully installed FreeBSD onto a ESDI disk controlled by a ACB-2320. No other operating system was present on the disk.

To do so I low level formatted the disk using `NEFMT.EXE` (`ftp`able from `www.adaptec.com`) and answered NO to the question whether the disk should be formatted with a spare sector on each track. The BIOS on the ACD-2320 was disabled. I used the `free configurable` option in the system BIOS to allow the BIOS to boot it.

Before using `NEFMT.EXE` I tried to format the disk using the ACB-2320 BIOS built-in formatter. This proved to be a show stopper, because it did not give me an option to disable spare sectoring. With spare sectoring enabled the FreeBSD installation process broke down on the `bad144` run.

Please check carefully which ACB-232xy variant you have. The x is either `0` or `2`, indicating a controller without or with a floppy controller on board.

The `y` is more interesting. It can either be a blank, a `A-8` or a `D`. A blank indicates a plain 10 Mbits/second controller. An `A-8` indicates a 15 Mbits/second controller capable of handling 52 sectors/track. A `D` means a 15 Mbits/second controller that can also handle drives with $>$ 36 sectors/track (also 52?).

All variations should be capable of using 1:1 interleaving. Use 1:1, FreeBSD is fast enough to handle it.

1.3.2 Western Digital WD1007 controllers

I successfully installed FreeBSD onto a ESDI disk controlled by a WD1007 controller. To be precise, it was a WD1007-WA2. Other variations of the WD1007 do exist.

To get it to work, I had to disable the sector translation and the WD1007's onboard BIOS. This implied I could not use the low-level formatter built into this BIOS. Instead, I grabbed `WDFMT.EXE` from `www.wdc.com` Running this formatted my drive just fine.

1.3.3 Ultrastor U14F controllers

According to multiple reports from the net, Ultrastor ESDI boards work OK with FreeBSD. I lack any further info on particular settings.

1.4 Further reading

If you intend to do some serious ESDI hacking, you might want to have the official standard at hand:

The latest ANSI X3T10 committee document is: Enhanced Small Device Interface (ESDI) [X3.170-1990/X3.170a-1991] [X3T10/792D Rev 11]

On Usenet the newsgroup comp.periphs (news:comp.periphs) is a noteworthy place to look for more info.

The World Wide Web (WWW) also proves to be a very handy info source: For info on Adaptec ESDI controllers see http://www.adaptec.com/. For info on Western Digital controllers see http://www.wdc.com/.

1.5 Thanks to...

Andrew Gordon for sending me an Adaptec 2320 controller and ESDI disk for testing.

2 What is SCSI?

SCSI is an acronym for Small Computer Systems Interface. It is an ANSI standard that has become one of the leading I/O buses in the computer industry. The foundation of the SCSI standard was laid by Shugart Associates (the same guys that gave the world the first mini floppy disks) when they introduced the SASI bus (Shugart Associates Standard Interface).

After some time an industry effort was started to come to a more strict standard allowing devices from different vendors to work together. This effort was recognized in the ANSI SCSI-1 standard. The SCSI-1 standard (approximately 1985) is rapidly becoming obsolete. The current standard is SCSI-2 (see Further reading), with SCSI-3 on the drawing boards.

In addition to a physical interconnection standard, SCSI defines a logical (command set) standard to which disk devices must adhere. This standard is called the Common Command Set (CCS) and was developed more or less in

parallel with ANSI SCSI-1. SCSI-2 includes the (revised) CCS as part of the standard itself. The commands are dependent on the type of device at hand. It does not make much sense of course to define a Write command for a scanner.

The SCSI bus is a parallel bus, which comes in a number of variants. The oldest and most used is an 8 bit wide bus, with single-ended signals, carried on 50 wires. (If you do not know what single-ended means, do not worry, that is what this document is all about.) Modern designs also use 16 bit wide buses, with differential signals. This allows transfer speeds of 20Mbytes/second, on cables lengths of up to 25 meters. SCSI-2 allows a maximum bus width of 32 bits, using an additional cable. Quickly emerging are Ultra SCSI (also called Fast-20) and Ultra2 (also called Fast-40). Fast-20 is 20 million transfers per second (20 Mbytes/sec on a 8 bit bus), Fast-40 is 40 million transfers per second (40 Mbytes/sec on a 8 bit bus). Most hard drives sold today are single-ended Ultra SCSI (8 or 16 bits).

Of course the SCSI bus not only has data lines, but also a number of control signals. A very elaborate protocol is part of the standard to allow multiple devices to share the bus in an efficient manner. In SCSI-2, the data is always checked using a separate parity line. In pre-SCSI-2 designs parity was optional.

In SCSI-3 even faster bus types are introduced, along with a serial SCSI busses that reduces the cabling overhead and allows a higher maximum bus length. You might see names like SSA and fibre channel in this context. None of the serial buses are currently in widespread use (especially not in the typical FreeBSD environment). For this reason the serial bus types are not discussed any further.

As you could have guessed from the description above, SCSI devices are intelligent. They have to be to adhere to the SCSI standard (which is over 2 inches thick BTW). So, for a hard disk drive for instance you do not specify a head/cylinder/sector to address a particular block, but simply the number of the block you want. Elaborate caching schemes, automatic bad block replacement etc are all made possible by this “intelligent device” approach.

On a SCSI bus, each possible pair of devices can communicate. Whether their function allows this is another matter, but the standard does not restrict it. To avoid signal contention, the 2 devices have to arbitrate for the bus before using it.

The philosophy of SCSI is to have a standard that allows older-standard devices to work with newer-standard ones. So, an old SCSI-1 device should normally work on a SCSI-2 bus. I say Normally, because it is not absolutely sure that the implementation of an old device follows the (old) standard closely enough to be acceptable on a new bus. Modern devices are usually more well-behaved, because the standardization has become more strict and is better adhered to by the device manufacturers.

Generally speaking, the chances of getting a working set of devices on a single bus is better when all the devices are SCSI-2 or newer. This implies that you do not have to dump all your old stuff when you get that shiny 80GB disk: I own a system on which a pre-SCSI-1 disk, a SCSI-2 QIC tape unit, a SCSI-1 helical scan tape unit and 2 SCSI-1 disks work together quite happily. From a performance standpoint you might want to separate your older and newer (=faster) devices however. This is especially advantageous if you have an Ultra160 host adapter where you should separate your U160 devices from the Fast and Wide SCSI-2 devices.

2.1 Components of SCSI

As said before, SCSI devices are smart. The idea is to put the knowledge about intimate hardware details onto the SCSI device itself. In this way, the host system does not have to worry about things like how many heads a hard disks has, or how many tracks there are on a specific tape device. If you are curious, the standard specifies commands with which you can query your devices on their hardware particulars. FreeBSD uses this capability during boot to check out what devices are connected and whether they need any special treatment.

The advantage of intelligent devices is obvious: the device drivers on the host can be made in a much more generic fashion, there is no longer a need to change (and qualify!) drivers for every odd new device that is introduced.

For cabling and connectors there is a golden rule: get good stuff. With bus speeds going up all the time you will save yourself a lot of grief by using good material.

So, gold plated connectors, shielded cabling, sturdy connector hoods with strain reliefs etc are the way to go. Second golden rule: do no use cables longer than necessary. I once spent 3 days hunting down a problem with a flaky machine only to discover that shortening the SCSI bus by 1 meter solved the problem. And the original bus length was well within the SCSI specification.

2.2 SCSI bus types

From an electrical point of view, there are two incompatible bus types: single-ended and differential. This means that there are two different main groups of SCSI devices and controllers, which cannot be mixed on the same bus. It is possible however to use special converter hardware to transform a single-ended bus into a differential one (and vice versa). The differences between the bus types are explained in the next sections.

In lots of SCSI related documentation there is a sort of jargon in use to abbreviate the different bus types. A small list:

- FWD: Fast Wide Differential
- FND: Fast Narrow Differential
- SE: Single Ended
- FN: Fast Narrow
- etc.

With a minor amount of imagination one can usually imagine what is meant.

Wide is a bit ambiguous, it can indicate 16 or 32 bit buses. As far as I know, the 32 bit variant is not (yet) in use, so wide normally means 16 bit.

Fast means that the timing on the bus is somewhat different, so that on a narrow (8 bit) bus 10 Mbytes/sec are possible instead of 5 Mbytes/sec for “slow” SCSI. As discussed before, bus speeds of 20 and 40 million transfers/second are also emerging (Fast-20 == Ultra SCSI and Fast-40 == Ultra2 SCSI).

> **Note:** The data lines > 8 are only used for data transfers and device addressing. The transfers of commands and status messages etc are only performed on the lowest 8 data lines. The standard allows narrow devices to operate on a wide bus. The usable bus width is negotiated between the devices. You have to watch your device addressing closely when mixing wide and narrow.

2.2.1 Single ended buses

A single-ended SCSI bus uses signals that are either 5 Volts or 0 Volts (indeed, TTL levels) and are relative to a COMMON ground reference. A singled ended 8 bit SCSI bus has approximately 25 ground lines, who are all tied to a single “rail” on all devices. A standard single ended bus has a maximum length of 6 meters. If the same bus is used with fast-SCSI devices, the maximum length allowed drops to 3 meters. Fast-SCSI means that instead of 5Mbytes/sec the bus allows 10Mbytes/sec transfers.

Fast-20 (Ultra SCSI) and Fast-40 allow for 20 and 40 million transfers/second respectively. So, F20 is 20 Mbytes/second on a 8 bit bus, 40 Mbytes/second on a 16 bit bus etc. For F20 the max bus length is 1.5 meters, for

F40 it becomes 0.75 meters. Be aware that F20 is pushing the limits quite a bit, so you will quickly find out if your SCSI bus is electrically sound.

> **Note:** If some devices on your bus use “fast” to communicate your bus must adhere to the length restrictions for fast buses!

It is obvious that with the newer fast-SCSI devices the bus length can become a real bottleneck. This is why the differential SCSI bus was introduced in the SCSI-2 standard.

For connector pinning and connector types please refer to the SCSI-2 standard (see Further reading) itself, connectors etc are listed there in painstaking detail.

Beware of devices using non-standard cabling. For instance Apple uses a 25pin D-type connecter (like the one on serial ports and parallel printers). Considering that the official SCSI bus needs 50 pins you can imagine the use of this connector needs some “creative cabling”. The reduction of the number of ground wires they used is a bad idea, you better stick to 50 pins cabling in accordance with the SCSI standard. For Fast-20 and 40 do not even think about buses like this.

2.2.2 Differential buses

A differential SCSI bus has a maximum length of 25 meters. Quite a difference from the 3 meters for a single-ended fast-SCSI bus. The idea behind differential signals is that each bus signal has its own return wire. So, each signal is carried on a (preferably twisted) pair of wires. The voltage difference between these two wires determines whether the signal is asserted or de-asserted. To a certain extent the voltage difference between ground and the signal wire pair is not relevant (do not try 10 kVolts though).

It is beyond the scope of this document to explain why this differential idea is so much better. Just accept that electrically seen the use of differential signals gives a much better noise margin. You will normally find differential buses in use for inter-cabinet connections. Because of the lower cost single ended is mostly used for shorter buses like inside cabinets.

There is nothing that stops you from using differential stuff with FreeBSD, as long as you use a controller that has device driver support in FreeBSD. As an example, Adaptec marketed the AHA1740 as a single ended board, whereas the AHA1744 was differential. The software interface to the host is identical for both.

2.2.3 Terminators

Terminators in SCSI terminology are resistor networks that are used to get a correct impedance matching. Impedance matching is important to get clean signals on the bus, without reflections or ringing. If you once made a long distance telephone call on a bad line you probably know what reflections are. With 20Mbytes/sec traveling over your SCSI bus, you do not want signals echoing back.

Terminators come in various incarnations, with more or less sophisticated designs. Of course, there are internal and external variants. Many SCSI devices come with a number of sockets in which a number of resistor networks can (must be!) installed. If you remove terminators from a device, carefully store them. You will need them when you ever decide to reconfigure your SCSI bus. There is enough variation in even these simple tiny things to make finding the exact replacement a frustrating business. There are also SCSI devices that have a single jumper to enable or disable a built-in terminator. There are special terminators you can stick onto a flat cable bus. Others look like external connectors, or a connector hood without a cable. So, lots of choice as you can see.

There is much debate going on if and when you should switch from simple resistor (passive) terminators to active terminators. Active terminators contain slightly more elaborate circuit to give cleaner bus signals. The general consensus seems to be that the usefulness of active termination increases when you have long buses and/or fast devices. If you ever have problems with your SCSI buses you might consider trying an active terminator. Try to borrow one first, they reputedly are quite expensive.

Please keep in mind that terminators for differential and single-ended buses are not identical. You should *not mix* the two variants.

OK, and now where should you install your terminators? This is by far the most misunderstood part of SCSI. And it is by far the simplest. The rule is: *every single line on the SCSI bus has 2 (two) terminators, one at each end of the bus.* So, two and not one or three or whatever. Do yourself a favor and stick to this rule. It will save you endless grief, because wrong termination has the potential to introduce highly mysterious bugs. (Note the "potential" here; the nastiest part is that it may or may not work.)

A common pitfall is to have an internal (flat) cable in a machine and also an external cable attached to the controller. It seems almost everybody forgets to remove the terminators from the controller. The terminator must now be on the last external device, and not on the controller! In general, every reconfiguration of a SCSI bus must pay attention to this.

> **Note:** Termination is to be done on a per-line basis. This means if you have both narrow and wide buses connected to the same host adapter, you need to enable termination on the higher 8 bits of the bus on the adapter (as well as the last devices on each bus, of course).

What I did myself is remove all terminators from my SCSI devices and controllers. I own a couple of external terminators, for both the Centronics-type external cabling and for the internal flat cable connectors. This makes reconfiguration much easier.

On modern devices, sometimes integrated terminators are used. These things are special purpose integrated circuits that can be enabled or disabled with a control pin. It is not necessary to physically remove them from a device. You may find them on newer host adapters, sometimes they are software configurable, using some sort of setup tool. Some will even auto-detect the cables attached to the connectors and automatically set up the termination as necessary. At any rate, consult your documentation!

2.2.4 Terminator power

The terminators discussed in the previous chapter need power to operate properly. On the SCSI bus, a line is dedicated to this purpose. So, simple huh?

Not so. Each device can provide its own terminator power to the terminator sockets it has on-device. But if you have external terminators, or when the device supplying the terminator power to the SCSI bus line is switched off you are in trouble.

The idea is that initiators (these are devices that initiate actions on the bus, a discussion follows) must supply terminator power. All SCSI devices are allowed (but not required) to supply terminator power.

To allow for un-powered devices on a bus, the terminator power must be supplied to the bus via a diode. This prevents the backflow of current to un-powered devices.

To prevent all kinds of nastiness, the terminator power is usually fused. As you can imagine, fuses might blow. This can, but does not have to, lead to a non functional bus. If multiple devices supply terminator power, a single blown

fuse will not put you out of business. A single supplier with a blown fuse certainly will. Clever external terminators sometimes have a LED indication that shows whether terminator power is present.

In newer designs auto-restoring fuses that “reset” themselves after some time are sometimes used.

2.2.5 Device addressing

Because the SCSI bus is, ehh, a bus there must be a way to distinguish or address the different devices connected to it.

This is done by means of the SCSI or target ID. Each device has a unique target ID. You can select the ID to which a device must respond using a set of jumpers, or a dip switch, or something similar. Some SCSI host adapters let you change the target ID from the boot menu. (Yet some others will not let you change the ID from 7.) Consult the documentation of your device for more information.

Beware of multiple devices configured to use the same ID. Chaos normally reigns in this case. A pitfall is that one of the devices sharing the same ID sometimes even manages to answer to I/O requests!

For an 8 bit bus, a maximum of 8 targets is possible. The maximum is 8 because the selection is done bitwise using the 8 data lines on the bus. For wide buses this increases to the number of data lines (usually 16).

> **Note:** A narrow SCSI device can not communicate with a SCSI device with a target ID larger than 7. This means it is generally not a good idea to move your SCSI host adapter’s target ID to something higher than 7 (or your CDROM will stop working).

The higher the SCSI target ID, the higher the priority the devices has. When it comes to arbitration between devices that want to use the bus at the same time, the device that has the highest SCSI ID will win. This also means that the SCSI host adapter usually uses target ID 7. Note however that the lower 8 IDs have higher priorities than the higher 8 IDs on a wide-SCSI bus. Thus, the order of target IDs is: [7 6 .. 1 0 15 14 .. 9 8] on a wide-SCSI system. (If you are wondering why the lower 8 have higher priority, read the previous paragraph for a hint.)

For a further subdivision, the standard allows for Logical Units or LUNs for short. A single target ID may have multiple LUNs. For example, a tape device including a tape changer may have LUN 0 for the tape device itself, and LUN 1 for the tape changer. In this way, the host system can address each of the functional units of the tape changer as desired.

2.2.6 Bus layout

SCSI buses are linear. So, not shaped like Y-junctions, star topologies, rings, cobwebs or whatever else people might want to invent. One of the most common mistakes is for people with wide-SCSI host adapters to connect devices on all three connecters (external connector, internal wide connector, internal narrow connector). Do not do that. It may appear to work if you are really lucky, but I can almost guarantee that your system will stop functioning at the most unfortunate moment (this is also known as “Murphy’s law”).

You might notice that the terminator issue discussed earlier becomes rather hairy if your bus is not linear. Also, if you have more connectors than devices on your internal SCSI cable, make sure you attach devices on connectors on both ends instead of using the connectors in the middle and let one or both ends dangle. This will screw up the termination of the bus.

The electrical characteristics, its noise margins and ultimately the reliability of it all are tightly related to linear bus rule.

Stick to the linear bus rule!

2.3 Using SCSI with FreeBSD

2.3.1 About translations, BIOSes and magic...

As stated before, you should first make sure that you have a electrically sound bus.

When you want to use a SCSI disk on your PC as boot disk, you must aware of some quirks related to PC BIOSes. The PC BIOS in its first incarnation used a low level physical interface to the hard disk. So, you had to tell the BIOS (using a setup tool or a BIOS built-in setup) how your disk physically looked like. This involved stating number of heads, number of cylinders, number of sectors per track, obscure things like precompensation and reduced write current cylinder etc.

One might be inclined to think that since SCSI disks are smart you can forget about this. Alas, the arcane setup issue is still present today. The system BIOS needs to know how to access your SCSI disk with the head/cyl/sector method in order to load the FreeBSD kernel during boot.

The SCSI host adapter or SCSI controller you have put in your AT/EISA/PCI/whatever bus to connect your disk therefore has its own on-board BIOS. During system startup, the SCSI BIOS takes over the hard disk interface routines from the system BIOS. To fool the system BIOS, the system setup is normally set to No hard disk present. Obvious, is it not?

The SCSI BIOS itself presents to the system a so called *translated* drive. This means that a fake drive table is constructed that allows the PC to boot the drive. This translation is often (but not always) done using a pseudo drive with 64 heads and 32 sectors per track. By varying the number of cylinders, the SCSI BIOS adapts to the actual drive size. It is useful to note that 32 * 64 / 2 = the size of your drive in megabytes. The division by 2 is to get from disk blocks that are normally 512 bytes in size to Kbytes.

Right. All is well now?! No, it is not. The system BIOS has another quirk you might run into. The number of cylinders of a bootable hard disk cannot be greater than 1024. Using the translation above, this is a show-stopper for disks greater than 1 GB. With disk capacities going up all the time this is causing problems.

Fortunately, the solution is simple: just use another translation, e.g. with 128 heads instead of 32. In most cases new SCSI BIOS versions are available to upgrade older SCSI host adapters. Some newer adapters have an option, in the form of a jumper or software setup selection, to switch the translation the SCSI BIOS uses.

It is very important that *all* operating systems on the disk use the *same translation* to get the right idea about where to find the relevant partitions. So, when installing FreeBSD you must answer any questions about heads/cylinders etc using the translated values your host adapter uses.

Failing to observe the translation issue might lead to un-bootable systems or operating systems overwriting each others partitions. Using fdisk you should be able to see all partitions.

You might have heard some talk of “lying” devices? Older FreeBSD kernels used to report the geometry of SCSI disks when booting. An example from one of my systems:

```
aha0 targ 0 lun 0: <MICROP 1588-15MB1057404HSP4>
da0: 636MB (1303250 total sec), 1632 cyl, 15 head, 53 sec, bytes/sec 512
```

Newer kernels usually do not report this information. e.g.

```
(bt0:0:0): "SEAGATE ST41651 7574" type 0 fixed SCSI 2
```

```
da0(bt0:0:0): Direct-Access 1350MB (2766300 512 byte sectors)
```

Why has this changed?

This info is retrieved from the SCSI disk itself. Newer disks often use a technique called zone bit recording. The idea is that on the outer cylinders of the drive there is more space so more sectors per track can be put on them. This results in disks that have more tracks on outer cylinders than on the inner cylinders and, last but not least, have more capacity. You can imagine that the value reported by the drive when inquiring about the geometry now becomes suspect at best, and nearly always misleading. When asked for a geometry, it is nearly always better to supply the geometry used by the BIOS, or *if the BIOS is never going to know about this disk*, (e.g. it is not a booting disk) to supply a fictitious geometry that is convenient.

2.3.2 SCSI subsystem design

FreeBSD uses a layered SCSI subsystem. For each different controller card a device driver is written. This driver knows all the intimate details about the hardware it controls. The driver has a interface to the upper layers of the SCSI subsystem through which it receives its commands and reports back any status.

On top of the card drivers there are a number of more generic drivers for a class of devices. More specific: a driver for tape devices (abbreviation: sa, for serial access), magnetic disks (da, for direct access), CDROMs (cd) etc. In case you are wondering where you can find this stuff, it all lives in `/sys/cam/scsi`. See the man pages in section 4 for more details.

The multi level design allows a decoupling of low-level bit banging and more high level stuff. Adding support for another piece of hardware is a much more manageable problem.

2.3.3 Kernel configuration

Dependent on your hardware, the kernel configuration file must contain one or more lines describing your host adapter(s). This includes I/O addresses, interrupts etc. Consult the manual page for your adapter driver to get more info. Apart from that, check out `/sys/i386/conf/LINT` for an overview of a kernel config file. `LINT` contains every possible option you can dream of. It does *not* imply `LINT` will actually get you to a working kernel at all.

Although it is probably stating the obvious: the kernel config file should reflect your actual hardware setup. So, interrupts, I/O addresses etc must match the kernel config file. During system boot messages will be displayed to indicate whether the configured hardware was actually found.

> **Note:** Note that most of the EISA/PCI drivers (namely `ahb`, `ahc`, `ncr` and `amd` will automatically obtain the correct parameters from the host adapters themselves at boot time; thus, you just need to write, for instance, `controller ahc0`.

An example loosely based on the FreeBSD 2.2.5-Release kernel config file `LINT` with some added comments (between []):

```
# SCSI host adapters: 'aha', 'ahb', 'aic', 'bt', 'nca'
#
# aha: Adaptec 154x
# ahb: Adaptec 174x
# ahc: Adaptec 274x/284x/294x
# aic: Adaptec 152x and sound cards using the Adaptec AIC-6360 (slow!)
```

```
# amd: AMD 53c974 based SCSI cards (e.g., Tekram DC-390 and 390T)
# bt: Most Buslogic controllers
# nca: ProAudioSpectrum cards using the NCR 5380 or Trantor T130
# ncr: NCR/Symbios 53c810/815/825/875 etc based SCSI cards
# uha: UltraStore 14F and 34F
# sea: Seagate ST01/02 8 bit controller (slow!)
# wds: Western Digital WD7000 controller (no scatter/gather!).
#

[For an Adaptec AHA274x/284x/294x/394x etc controller]
controller ahc0

[For an NCR/Symbios 53c875 based controller]
controller ncr0

[For an Ultrastor adapter]
controller uha0 at isa? port "IO_UHA0" bio irq ? drq 5 vector uhaintr

# Map SCSI buses to specific SCSI adapters
controller scbus0 at ahc0
controller scbus2 at ncr0
controller scbus1  at uha0

# The actual SCSI devices
disk da0 at scbus0 target 0 unit 0 [SCSI disk 0 is at scbus 0, LUN 0]
disk da1 at scbus0 target 1             [implicit LUN 0 if omitted]
disk da2 at scbus1 target 3             [SCSI disk on the uha0]
disk da3 at scbus2 target 4             [SCSI disk on the ncr0]
tape sa1 at scbus0 target 6             [SCSI tape at target 6]
device cd0 at scbus?                    [the first ever CDROM found, no wiring]
```

The example above tells the kernel to look for a ahc (Adaptec 274x) controller, then for an NCR/Symbios board, and so on. The lines following the controller specifications tell the kernel to configure specific devices but *only* attach them when they match the target ID and LUN specified on the corresponding bus.

Wired down devices get “first shot” at the unit numbers so the first non “wired down” device, is allocated the unit number one greater than the highest “wired down” unit number for that kind of device. So, if you had a SCSI tape at target ID 2 it would be configured as sa2, as the tape at target ID 6 is wired down to unit number 1.

> **Note:** Wired down devices need not be found to get their unit number. The unit number for a wired down device is reserved for that device, even if it is turned off at boot time. This allows the device to be turned on and brought on-line at a later time, without rebooting. Notice that a device’s unit number has *no* relationship with its target ID on the SCSI bus.

Below is another example of a kernel config file as used by FreeBSD version $< 2.0.5$. The difference with the first example is that devices are not “wired down”. “Wired down” means that you specify which SCSI target belongs to which device.

A kernel built to the config file below will attach the first SCSI disk it finds to da0, the second disk to da1 etc. If you ever removed or added a disk, all other devices of the same type (disk in this case) would “move around”. This implies you have to change `/etc/fstab` each time.

Although the old style still works, you are *strongly* recommended to use this new feature. It will save you a lot of grief whenever you shift your hardware around on the SCSI buses. So, when you re-use your old trusty config file after upgrading from a pre-FreeBSD2.0.5.R system check this out.

```
[driver for Adaptec 174x]
controller      ahb0 at isa? bio irq 11 vector ahbintr

[for Adaptec 154x]
controller      aha0    at isa? port "IO_AHA0" bio irq 11 drq 5 vector ahaintr

[for Seagate ST01/02]
controller      sea0    at isa? bio irq 5 iomem 0xc8000 iosiz 0x2000 vector seaintr

controller      scbus0

device          da0     [support for 4 SCSI harddisks, da0 up da3]
device          sa0 [support for 2 SCSI tapes]

[for the CDROM]
device          cd0     #Only need one of these, the code dynamically grows
```

Both examples support SCSI disks. If during boot more devices of a specific type (e.g. da disks) are found than are configured in the booting kernel, the system will simply allocate more devices, incrementing the unit number starting at the last number “wired down”. If there are no “wired down” devices then counting starts at unit 0.

Use `man 4 scsi` to check for the latest info on the SCSI subsystem. For more detailed info on host adapter drivers use e.g., `man 4 ahc` for info on the Adaptec 294x driver.

2.3.4 Tuning your SCSI kernel setup

Experience has shown that some devices are slow to respond to INQUIRY commands after a SCSI bus reset (which happens at boot time). An INQUIRY command is sent by the kernel on boot to see what kind of device (disk, tape, CDROM etc.) is connected to a specific target ID. This process is called device probing by the way.

To work around the “slow response” problem, FreeBSD allows a tunable delay time before the SCSI devices are probed following a SCSI bus reset. You can set this delay time in your kernel configuration file using a line like:

```
options         SCSI_DELAY=15         #Be pessimistic about Joe SCSI device
```

This line sets the delay time to 15 seconds. On my own system I had to use 3 seconds minimum to get my trusty old CDROM drive to be recognized. Start with a high value (say 30 seconds or so) when you have problems with device recognition. If this helps, tune it back until it just stays working.

2.3.5 Rogue SCSI devices

Although the SCSI standard tries to be complete and concise, it is a complex standard and implementing things correctly is no easy task. Some vendors do a better job then others.

This is exactly where the “rogue” devices come into view. Rogues are devices that are recognized by the FreeBSD kernel as behaving slightly (...) non-standard. Rogue devices are reported by the kernel when booting. An example for two of my cartridge tape units:

```
Feb 25 21:03:34 yedi /kernel: ahb0 targ 5 lun 0: <TANDBERG TDC 3600       -06:>
Feb 25 21:03:34 yedi /kernel: sa0: Tandberg tdc3600 is a known rogue

Mar 29 21:16:37 yedi /kernel: aha0 targ 5 lun 0: <ARCHIVE VIPER 150  21247-005>
Mar 29 21:16:37 yedi /kernel: sa1: Archive  Viper 150 is a known rogue
```

For instance, there are devices that respond to all LUNs on a certain target ID, even if they are actually only one device. It is easy to see that the kernel might be fooled into believing that there are 8 LUNs at that particular target ID. The confusion this causes is left as an exercise to the reader.

The SCSI subsystem of FreeBSD recognizes devices with bad habits by looking at the INQUIRY response they send when probed. Because the INQUIRY response also includes the version number of the device firmware, it is even possible that for different firmware versions different workarounds are used. See e.g. `/sys/cam/scsi/scsi_sa.c` and `/sys/cam/scsi/scsi_all.c` for more info on how this is done.

This scheme works fine, but keep in mind that it of course only works for devices that are known to be weird. If you are the first to connect your bogus Mumbletech SCSI CDROM you might be the one that has to define which workaround is needed.

After you got your Mumbletech working, please send the required workaround to the FreeBSD development team for inclusion in the next release of FreeBSD. Other Mumbletech owners will be grateful to you.

2.3.6 Multiple LUN devices

In some cases you come across devices that use multiple logical units (LUNs) on a single SCSI ID. In most cases FreeBSD only probes devices for LUN 0. An example are so called bridge boards that connect 2 non-SCSI hard disks to a SCSI bus (e.g. an Emulex MD21 found in old Sun systems).

This means that any devices with LUNs != 0 are not normally found during device probe on system boot. To work around this problem you must add an appropriate entry in /sys/cam/scsi and rebuild your kernel.

Look for a struct that is initialized like below: (FIXME: which file? Do these entries still exist in this form now that we use CAM?)

```
{
        T_DIRECT, T_FIXED, "MAXTOR", "XT-4170S", "B5A",
        "mx1", SC_ONE_LU
}
```

For your Mumbletech BRIDGE2000 that has more than one LUN, acts as a SCSI disk and has firmware revision 123 you would add something like:

```
{
        T_DIRECT, T_FIXED, "MUMBLETECH", "BRIDGE2000", "123",
        "da", SC_MORE_LUS
}
```

The kernel on boot scans the inquiry data it receives against the table and acts accordingly. See the source for more info.

2.3.7 Tagged command queuing

Modern SCSI devices, particularly magnetic disks, support what is called tagged command queuing (TCQ).

In a nutshell, TCQ allows the device to have multiple I/O requests outstanding at the same time. Because the device is intelligent, it can optimize its operations (like head positioning) based on its own request queue. On SCSI devices like RAID (Redundant Array of Independent Disks) arrays the TCQ function is indispensable to take advantage of the device's inherent parallelism.

Each I/O request is uniquely identified by a "tag" (hence the name tagged command queuing) and this tag is used by FreeBSD to see which I/O in the device drivers queue is reported as complete by the device.

It should be noted however that TCQ requires device driver support and that some devices implemented it "not quite right" in their firmware. This problem bit me once, and it leads to highly mysterious problems. In such cases, try to disable TCQ.

2.3.8 Bus-master host adapters

Most, but not all, SCSI host adapters are bus mastering controllers. This means that they can do I/O on their own without putting load onto the host CPU for data movement.

This is of course an advantage for a multitasking operating system like FreeBSD. It must be noted however that there might be some rough edges.

For instance an Adaptec 1542 controller can be set to use different transfer speeds on the host bus (ISA or AT in this case). The controller is settable to different rates because not all motherboards can handle the higher speeds. Problems like hang-ups, bad data etc might be the result of using a higher data transfer rate then your motherboard can stomach.

The solution is of course obvious: switch to a lower data transfer rate and try if that works better.

In the case of a Adaptec 1542, there is an option that can be put into the kernel config file to allow dynamic determination of the right, read: fastest feasible, transfer rate. This option is disabled by default:

```
options         "TUNE_1542"             #dynamic tune of bus DMA speed
```

Check the manual pages for the host adapter that you use. Or better still, use the ultimate documentation (read: driver source).

2.4 Tracking down problems

The following list is an attempt to give a guideline for the most common SCSI problems and their solutions. It is by no means complete.

- Check for loose connectors and cables.
- Check and double check the location and number of your terminators.
- Check if your bus has at least one supplier of terminator power (especially with external terminators.
- Check if no double target IDs are used.
- Check if all devices to be used are powered up.

- Make a minimal bus config with as little devices as possible.
- If possible, configure your host adapter to use slow bus speeds.
- Disable tagged command queuing to make things as simple as possible (for a NCR host adapter based system see man ncrcontrol)
- If you can compile a kernel, make one with the `SCSIDEBUG` option, and try accessing the device with debugging turned on for that device. If your device does not even probe at startup, you may have to define the address of the device that is failing, and the desired debug level in `/sys/cam/cam_debug.h`. If it probes but just does not work, you can use the camcontrol(8) command to dynamically set a debug level to it in a running kernel (if `CAMDEBUG` is defined). This will give you *copious* debugging output with which to confuse the gurus. See `man camcontrol` for more exact information. Also look at `man 4 pass`.

2.5 Further reading

If you intend to do some serious SCSI hacking, you might want to have the official standard at hand:

Approved American National Standards can be purchased from ANSI at

13th Floor
11 West 42nd Street
New York
NY 10036
Sales Dept: (212) 642-4900

You can also buy many ANSI standards and most committee draft documents from Global Engineering Documents,

15 Inverness Way East
Englewood
CO, 80112-5704
Phone: (800) 854-7179
Outside USA and Canada: (303) 792-2181
Fax: (303) 792- 2192

Many X3T10 draft documents are available electronically on the SCSI BBS (719-574-0424) and on the `ncrinfo.ncr.com` anonymous FTP site.

Latest X3T10 committee documents are:

- AT Attachment (ATA or IDE) [X3.221-1994] (*Approved*)
- ATA Extensions (ATA-2) [X3T10/948D Rev 2i]
- Enhanced Small Device Interface (ESDI) [X3.170-1990/X3.170a-1991] (*Approved*)
- Small Computer System Interface — 2 (SCSI-2) [X3.131-1994] (*Approved*)
- SCSI-2 Common Access Method Transport and SCSI Interface Module (CAM) [X3T10/792D Rev 11]

Other publications that might provide you with additional information are:

- “SCSI: Understanding the Small Computer System Interface”, written by NCR Corporation. Available from: Prentice Hall, Englewood Cliffs, NJ, 07632 Phone: (201) 767-5937 ISBN 0-13-796855-8
- “Basics of SCSI”, a SCSI tutorial written by Ancot Corporation Contact Ancot for availability information at: Phone: (415) 322-5322 Fax: (415) 322-0455
- “SCSI Interconnection Guide Book”, an AMP publication (dated 4/93, Catalog 65237) that lists the various SCSI connectors and suggests cabling schemes. Available from AMP at (800) 522-6752 or (717) 564-0100
- “Fast Track to SCSI”, A Product Guide written by Fujitsu. Available from: Prentice Hall, Englewood Cliffs, NJ, 07632 Phone: (201) 767-5937 ISBN 0-13-307000-X
- “The SCSI Bench Reference”, “The SCSI Encyclopedia”, and the “SCSI Tutor”, ENDL Publications, 14426 Black Walnut Court, Saratoga CA, 95070 Phone: (408) 867-6642
- “Zadian SCSI Navigator” (quick ref. book) and “Discover the Power of SCSI” (First book along with a one-hour video and tutorial book), Zadian Software, Suite 214, 1210 S. Bascom Ave., San Jose, CA 92128, (408) 293-0800

On Usenet the newsgroups comp.periphs.scsi (news:comp.periphs.scsi) and comp.periphs (news:comp.periphs) are noteworthy places to look for more info. You can also find the SCSI-FAQ (http://scsifaq.org:9080/scsi_faq/scsifaq.html) there, which is posted periodically.

Most major SCSI device and host adapter suppliers operate FTP sites and/or BBS systems. They may be valuable sources of information about the devices you own.

3 * Disk/tape controllers

3.1 * SCSI

3.2 * IDE

3.3 * Floppy

4 Hard drives

4.1 SCSI hard drives

Contributed by Satoshi Asami <asami@FreeBSD.org>. 17 February 1998.

As mentioned in the SCSI section, virtually all SCSI hard drives sold today are SCSI-2 compliant and thus will work fine as long as you connect them to a supported SCSI host adapter. Most problems people encounter are either due to badly designed cabling (cable too long, star topology, etc.), insufficient termination, or defective parts. Please refer to

the SCSI section first if your SCSI hard drive is not working. However, there are a couple of things you may want to take into account before you purchase SCSI hard drives for your system.

4.1.1 Rotational speed

Rotational speeds of SCSI drives sold today range from around 4,500RPM to 15,000RPM. Most of them are either 7,200RPM or 10,000RPM, with 15,000RPM becoming affordable (June 2002). Even though the 10,000RPM drives can generally transfer data faster, they run considerably hotter than their 7,200RPM counterparts. A large fraction of today's disk drive malfunctions are heat-related. If you do not have very good cooling in your PC case, you may want to stick with 7,200RPM or slower drives.

Note that newer drives, with higher areal recording densities, can deliver much more bits per rotation than older ones. Today's top-of-line 7,200RPM drives can sustain a throughput comparable to 10,000RPM drives of one or two model generations ago. The number to find on the spec sheet for bandwidth is "internal data (or transfer) rate". It is usually in megabits/sec so divide it by 8 and you will get the rough approximation of how much megabytes/sec you can get out of the drive.

(If you are a speed maniac and want a 15,000RPM drive for your cute little PC, be my guest; however, those drives become extremely hot. Do not even think about it if you do not have a fan blowing air *directly at* the drive or a properly ventilated disk enclosure.)

Obviously, the latest 15,000RPM drives and 10,000RPM drives can deliver more data than the latest 7,200RPM drives, so if absolute bandwidth is the necessity for your applications, you have little choice but to get the faster drives. Also, if you need low latency, faster drives are better; not only do they usually have lower average seek times, but also the rotational delay is one place where slow-spinning drives can never beat a faster one. (The average rotational latency is half the time it takes to rotate the drive once; thus, it is 2 milliseconds for 15,000RPM, 3ms for 10,000RPM drives, 4.2ms for 7,200RPM drives and 5.6ms for 5,400RPM drives.) Latency is seek time plus rotational delay. Make sure you understand whether you need low latency or more accesses per second, though; in the latter case (e.g., news servers), it may not be optimal to purchase one big fast drive. You can achieve similar or even better results by using the ccd (concatenated disk) driver to create a striped disk array out of multiple slower drives for comparable overall cost.

Make sure you have adequate air flow around the drive, especially if you are going to use a fast-spinning drive. You generally need at least 1/2" (1.25cm) of spacing above and below a drive. Understand how the air flows through your PC case. Most cases have the power supply suck the air out of the back. See where the air flows in, and put the drive where it will have the largest volume of cool air flowing around it. You may need to seal some unwanted holes or add a new fan for effective cooling.

Another consideration is noise. Many 10,000 or faster drives generate a high-pitched whine which is quite unpleasant to most people. That, plus the extra fans often required for cooling, may make 10,000 or faster drives unsuitable for some office and home environments.

4.1.2 Form factor

Most SCSI drives sold today are of 3.5" form factor. They come in two different heights; 1.6" ("half-height") or 1" ("low-profile"). The half-height drive is the same height as a CDROM drive. However, do not forget the spacing rule mentioned in the previous section. If you have three standard 3.5" drive bays, you will not be able to put three half-height drives in there (without frying them, that is).

4.1.3 Interface

The majority of SCSI hard drives sold today are Ultra, Ultra-wide, or Ultra160 SCSI. As of this writing (June 2002), the first Ultra320 host adapters and devices become available. The maximum bandwidth of Ultra SCSI is 20MB/sec, and Ultra-wide SCSI is 40MB/sec. Ultra160 can transfer 160MB/sec and Ultra320 can transfer 320MB/sec. There is no difference in max cable length between Ultra and Ultra-wide; however, the more devices you have on the same bus, the sooner you will start having bus integrity problems. Unless you have a well-designed disk enclosure, it is not easy to make more than 5 or 6 Ultra SCSI drives work on a single bus.

On the other hand, if you need to connect many drives, going for Fast-wide SCSI may not be a bad idea. That will have the same max bandwidth as Ultra (narrow) SCSI, while electronically it is much easier to get it "right". My advice would be: if you want to connect many disks, get wide or Ultra160 SCSI drives; they usually cost a little more but it may save you down the road. (Besides, if you can not afford the cost difference, you should not be building a disk array.)

There are two variant of wide SCSI drives; 68-pin and 80-pin SCA (Single Connector Attach). The SCA drives do not have a separate 4-pin power connector, and also read the SCSI ID settings through the 80-pin connector. If you are really serious about building a large storage system, get SCA drives and a good SCA enclosure (dual power supply with at least one extra fan). They are more electronically sound than 68-pin counterparts because there is no "stub" of the SCSI bus inside the disk canister as in arrays built from 68-pin drives. They are easier to install too (you just need to screw the drive in the canister, instead of trying to squeeze in your fingers in a tight place to hook up all the little cables (like the SCSI ID and disk activity LED lines).

4.2 * IDE hard drives

5 Tape drives

Contributed by Jonathan M. Bresler <`jmb@FreeBSD.org`>. 2 July 1996.

5.1 General tape access commands

mt(1) provides generic access to the tape drives. Some of the more common commands are `rewind`, `erase`, and `status`. See the mt(1) manual page for a detailed description.

5.2 Controller Interfaces

There are several different interfaces that support tape drives. The interfaces are SCSI, IDE, Floppy and Parallel Port. A wide variety of tape drives are available for these interfaces. Controllers are discussed in Disk/tape controllers.

5.3 SCSI drives

The st(4) driver provides support for 8mm (Exabyte), 4mm (DAT: Digital Audio Tape), QIC (Quarter-Inch Cartridge), DLT (Digital Linear Tape), QIC Mini cartridge and 9-track (remember the big reels that you see spinning in Hollywood computer rooms) tape drives. See the st(4) manual page for a detailed description.

The drives listed below are currently being used by members of the FreeBSD community. They are not the only drives that will work with FreeBSD. They just happen to be the ones that we use.

5.3.1 4mm (DAT: Digital Audio Tape)

Archive Python 28454

Archive Python 04687

HP C1533A

HP C1534A

HP 35450A

HP 35470A

HP 35480A

SDT-5000

Wangtek 6200

5.3.2 8mm (Exabyte)

EXB-8200

EXB-8500

EXB-8505

5.3.3 QIC (Quarter-Inch Cartridge)

Archive Anaconda 2750

Archive Viper 60

Archive Viper 150

Archive Viper 2525

Tandberg TDC 3600

Tandberg TDC 3620

Tandberg TDC 3800

Tandberg TDC 4222

Wangtek 5525ES

5.3.4 DLT (Digital Linear Tape)

Digital TZ87

5.3.5 Mini-Cartridge

Conner CTMS 3200

Exabyte 2501

5.3.6 Autoloaders/Changers

Hewlett-Packard HP C1553A Autoloading DDS2

5.4 * IDE drives

5.5 Floppy drives

Conner 420R

5.6 * Parallel port drives

5.7 Detailed Information

5.7.1 Archive Anaconda 2750

The boot message identifier for this drive is `ARCHIVE ANCDA 2750 28077 -003 type 1 removable SCSI 2`

This is a QIC tape drive.

Native capacity is 1.35GB when using QIC-1350 tapes. This drive will read and write QIC-150 (DC6150), QIC-250 (DC6250), and QIC-525 (DC6525) tapes as well.

Data transfer rate is 350kB/s using dump(8). Rates of 530kB/s have been reported when using Amanda

Production of this drive has been discontinued.

The SCSI bus connector on this tape drive is reversed from that on most other SCSI devices. Make sure that you have enough SCSI cable to twist the cable one-half turn before and after the Archive Anaconda tape drive, or turn your other SCSI devices upside-down.

Two kernel code changes are required to use this drive. This drive will not work as delivered.

If you have a SCSI-2 controller, short jumper 6. Otherwise, the drive behaves are a SCSI-1 device. When operating as a SCSI-1 device, this drive, “locks” the SCSI bus during some tape operations, including: fsf, rewind, and rewoffl.

If you are using the NCR SCSI controllers, patch the file `/usr/src/sys/pci/ncr.c` (as shown below). Build and install a new kernel.

```
*** 4831,4835 ****
                };

!               if (np->latetime>4) {
```

```
                        /*
                        **      Although we tried to wake it up,
--- 4831,4836 ----
                };

!               if (np->latetime>1200) {
                        /*
                        **      Although we tried to wake it up,
```

Reported by: Jonathan M. Bresler <jmb@FreeBSD.org>

5.7.2 Archive Python 28454

The boot message identifier for this drive is `ARCHIVE Python 28454-XXX4ASB type 1 removable SCSI 2 density code 0x8c, 512-byte blocks`

This is a DDS-1 tape drive.

Native capacity is 2.5GB on 90m tapes.

Data transfer rate is XXX.

This drive was repackaged by Sun Microsystems as model 595-3067.

Reported by: Bob Bishop <rb@gid.co.uk>

Throughput is in the 1.5 MByte/sec range, however this will drop if the disks and tape drive are on the same SCSI controller.

Reported by: Robert E. Seastrom <rs@seastrom.com>

5.7.3 Archive Python 04687

The boot message identifier for this drive is `ARCHIVE Python 04687-XXX 6580 Removable Sequential Access SCSI-2 device`

This is a DAT-DDS-2 drive.

Native capacity is 4GB when using 120m tapes.

This drive supports hardware data compression. Switch 4 controls MRS (Media Recognition System). MRS tapes have stripes on the transparent leader. Switch 4 *off* enables MRS, *on* disables MRS.

Parity is controlled by switch 5. Switch 5 *on* to enable parity control. Compression is enabled with Switch 6 *off*. It is possible to override compression with the `SCSI MODE SELECT` command (see mt(1)).

Data transfer rate is 800kB/s.

5.7.4 Archive Viper 60

The boot message identifier for this drive is `ARCHIVE VIPER 60 21116 -007 type 1 removable SCSI 1`

This is a QIC tape drive.

Native capacity is 60MB.

Data transfer rate is XXX.

Production of this drive has been discontinued.

Reported by: Philippe Regnauld <regnauld@hsc.fr>

5.7.5 Archive Viper 150

The boot message identifier for this drive is `ARCHIVE VIPER 150 21531 -004` `Archive Viper 150 is a known rogue` `type 1 removable SCSI 1`. A multitude of firmware revisions exist for this drive. Your drive may report different numbers (e.g `21247 -005`.

This is a QIC tape drive.

Native capacity is 150/250MB. Both 150MB (DC6150) and 250MB (DC6250) tapes have the recording format. The 250MB tapes are approximately 67% longer than the 150MB tapes. This drive can read 120MB tapes as well. It can not write 120MB tapes.

Data transfer rate is 100kB/s

This drive reads and writes DC6150 (150MB) and DC6250 (250MB) tapes.

This drives quirks are known and pre-compiled into the SCSI tape device driver (st(4)).

Under FreeBSD 2.2-CURRENT, use `mt blocksize 512` to set the blocksize. (The particular drive had firmware revision 21247 -005. Other firmware revisions may behave differently) Previous versions of FreeBSD did not have this problem.

Production of this drive has been discontinued.

Reported by: Pedro A M Vazquez <vazquez@IQM.Unicamp.BR>

Michael Smith <msmith@FreeBSD.org>

5.7.6 Archive Viper 2525

The boot message identifier for this drive is `ARCHIVE VIPER 2525 25462 -011` `type 1 removable SCSI 1`

This is a QIC tape drive.

Native capacity is 525MB.

Data transfer rate is 180kB/s at 90 inches/sec.

The drive reads QIC-525, QIC-150, QIC-120 and QIC-24 tapes. Writes QIC-525, QIC-150, and QIC-120.

Firmware revisions prior to `25462 -011` are bug ridden and will not function properly.

Production of this drive has been discontinued.

5.7.7 Conner 420R

The boot message identifier for this drive is `Conner tape`.

This is a floppy controller, mini cartridge tape drive.

Native capacity is XXXX

Data transfer rate is XXX

The drive uses QIC-80 tape cartridges.

Reported by: Mark Hannon <mark@seeware.DIALix.oz.au>

5.7.8 Conner CTMS 3200

The boot message identifier for this drive is `CONNER CTMS 3200 7.00 type 1 removable SCSI 2`.

This is a mini cartridge tape drive.

Native capacity is XXXX

Data transfer rate is XXX

The drive uses QIC-3080 tape cartridges.

Reported by: Thomas S. Traylor <tst@titan.cs.mci.com>

5.7.9 DEC TZ87 (http://www.digital.com/info/Customer-Update/931206004.txt.html)

The boot message identifier for this drive is `DEC TZ87 (C) DEC 9206 type 1 removable SCSI 2 density code 0x19`

This is a DLT tape drive.

Native capacity is 10GB.

This drive supports hardware data compression.

Data transfer rate is 1.2MB/s.

This drive is identical to the Quantum DLT2000. The drive firmware can be set to emulate several well-known drives, including an Exabyte 8mm drive.

Reported by: Wilko Bulte <wilko@FreeBSD.org>

5.7.10 Exabyte EXB-2501 (http://www.Exabyte.COM:80/Products/Minicartridge/2501/Rfeatures.html)

The boot message identifier for this drive is `EXABYTE EXB-2501`

This is a mini-cartridge tape drive.

Native capacity is 1GB when using MC3000XL mini cartridges.

Data transfer rate is XXX

This drive can read and write DC2300 (550MB), DC2750 (750MB), MC3000 (750MB), and MC3000XL (1GB) mini cartridges.

WARNING: This drive does not meet the SCSI-2 specifications. The drive locks up completely in response to a SCSI MODE_SELECT command unless there is a formatted tape in the drive. Before using this drive, set the tape blocksize with

```
# mt -f /dev/st0ctl.0 blocksize 1024
```

Before using a mini cartridge for the first time, the mini cartridge must be formated. FreeBSD 2.1.0-RELEASE and earlier:

```
# /sbin/scsi -f /dev/rst0.ctl -s 600 -c "4 0 0 0 0 0"
```

(Alternatively, fetch a copy of the `scsiformat` shell script from FreeBSD 2.1.5/2.2.) FreeBSD 2.1.5 and later:

```
# /sbin/scsiformat -q -w /dev/rst0.ctl
```

Right now, this drive cannot really be recommended for FreeBSD.

Reported by: Bob Beaulieu <ez@eztravel.com>

5.7.11 Exabyte EXB-8200

The boot message identifier for this drive is `EXABYTE EXB-8200 252X type 1 removable SCSI 1`

This is an 8mm tape drive.

Native capacity is 2.3GB.

Data transfer rate is 270kB/s.

This drive is fairly slow in responding to the SCSI bus during boot. A custom kernel may be required (set SCSI_DELAY to 10 seconds).

There are a large number of firmware configurations for this drive, some have been customized to a particular vendor's hardware. The firmware can be changed via EPROM replacement.

Production of this drive has been discontinued.

Reported by: Michael Smith <msmith@FreeBSD.org>

5.7.12 Exabyte EXB-8500

The boot message identifier for this drive is `EXABYTE EXB-8500-85Qanx0 0415 type 1 removable SCSI 2`

This is an 8mm tape drive.

Native capacity is 5GB.

Data transfer rate is 300kB/s.

Reported by: Greg Lehey <grog@lemis.de>

5.7.13 Exabyte EXB-8505 (http://www.Exabyte.COM:80/Products/8mm/8505XL/Rfeatures.html)

The boot message identifier for this drive is `EXABYTE EXB-85058SQANXR1 05B0 type 1 removable SCSI 2`

This is an 8mm tape drive which supports compression, and is upward compatible with the EXB-5200 and EXB-8500.

Native capacity is 5GB.

The drive supports hardware data compression.

Data transfer rate is 300kB/s.

Reported by: Glen Foster <gfoster@gfoster.com>

5.7.14 Hewlett-Packard HP C1533A

The boot message identifier for this drive is `HP C1533A 9503 type 1 removable SCSI 2`.

This is a DDS-2 tape drive. DDS-2 means hardware data compression and narrower tracks for increased data capacity.

Native capacity is 4GB when using 120m tapes. This drive supports hardware data compression.

Data transfer rate is 510kB/s.

This drive is used in Hewlett-Packard's SureStore 6000eU and 6000i tape drives and C1533A DDS-2 DAT drive.

The drive has a block of 8 dip switches. The proper settings for FreeBSD are: 1 ON; 2 ON; 3 OFF; 4 ON; 5 ON; 6 ON; 7 ON; 8 ON.

switch 1	switch 2	Result
On	On	Compression enabled at power-on, with host control
On	Off	Compression enabled at power-on, no host control
Off	On	Compression disabled at power-on, with host control
Off	Off	Compression disabled at power-on, no host control

Switch 3 controls MRS (Media Recognition System). MRS tapes have stripes on the transparent leader. These identify the tape as DDS (Digital Data Storage) grade media. Tapes that do not have the stripes will be treated as write-protected. Switch 3 OFF enables MRS. Switch 3 ON disables MRS.

See HP SureStore Tape Products (http://www.hp.com/tape/c_intro.html) and Hewlett-Packard Disk and Tape Technical Information (http://www.impediment.com/hp/hp_technical.html) for more information on configuring this drive.

Warning: Quality control on these drives varies greatly. One FreeBSD core-team member has returned 2 of these drives. Neither lasted more than 5 months.

Reported by: Stefan Esser <`se@FreeBSD.org`>

5.7.15 Hewlett-Packard HP 1534A

The boot message identifier for this drive is `HP HP35470A T503 type 1 removable SCSI 2 Sequential-Access density code 0x13, variable blocks.`

This is a DDS-1 tape drive. DDS-1 is the original DAT tape format.

Native capacity is 2GB when using 90m tapes.

Data transfer rate is 183kB/s.

The same mechanism is used in Hewlett-Packard's SureStore 2000i (http://www.dmo.hp.com/tape/sst2000.htm) tape drive, C35470A DDS format DAT drive, C1534A DDS format DAT drive and HP C1536A DDS format DAT drive.

The HP C1534A DDS format DAT drive has two indicator lights, one green and one amber. The green one indicates tape action: slow flash during load, steady when loaded, fast flash during read/write operations. The amber one

indicates warnings: slow flash when cleaning is required or tape is nearing the end of its useful life, steady indicates an hard fault. (factory service required?)

Reported by Gary Crutcher <gcrutchr@nightflight.com>

5.7.16 Hewlett-Packard HP C1553A Autoloading DDS2

The boot message identifier for this drive is "".

This is a DDS-2 tape drive with a tape changer. DDS-2 means hardware data compression and narrower tracks for increased data capacity.

Native capacity is 24GB when using 120m tapes. This drive supports hardware data compression.

Data transfer rate is 510kB/s (native).

This drive is used in Hewlett-Packard's SureStore 12000e (http://www.dmo.hp.com/tape/sst12000.htm) tape drive.

The drive has two selectors on the rear panel. The selector closer to the fan is SCSI id. The other selector should be set to 7.

There are four internal switches. These should be set: 1 ON; 2 ON; 3 ON; 4 OFF.

At present the kernel drivers do not automatically change tapes at the end of a volume. This shell script can be used to change tapes:

```
#!/bin/sh
PATH="/sbin:/usr/sbin:/bin:/usr/bin"; export PATH

usage()
{
        echo "Usage: dds_changer [123456ne] raw-device-name
        echo "1..6 = Select cartridge"
        echo "next cartridge"
        echo "eject magazine"
        exit 2
}

if [ $# -ne 2 ] ; then
        usage
fi

cdb3=0
cdb4=0
cdb5=0

case $1 in
        [123456])
                cdb3=$1
                cdb4=1
                ;;
        n)
                ;;
        e)
                cdb5=0x80
                ;;
```

```
        ?)
                usage
                ;;
esac

scsi -f $2 -s 100 -c "1b 0 0 $cdb3 $cdb4 $cdb5"
```

5.7.17 Hewlett-Packard HP 35450A

The boot message identifier for this drive is `HP HP35450A -A C620 type 1 removable SCSI 2 Sequential-Access density code 0x13`

This is a DDS-1 tape drive. DDS-1 is the original DAT tape format.

Native capacity is 1.2GB.

Data transfer rate is 160kB/s.

Reported by: Mark Thompson <`mark.a.thompson@pobox.com`>

5.7.18 Hewlett-Packard HP 35470A

The boot message identifier for this drive is `HP HP35470A 9 09 type 1 removable SCSI 2`

This is a DDS-1 tape drive. DDS-1 is the original DAT tape format.

Native capacity is 2GB when using 90m tapes.

Data transfer rate is 183kB/s.

The same mechanism is used in Hewlett-Packard's SureStore 2000i (http://www.dmo.hp.com/tape/sst2000.htm) tape drive, C35470A DDS format DAT drive, C1534A DDS format DAT drive, and HP C1536A DDS format DAT drive.

Warning: Quality control on these drives varies greatly. One FreeBSD core-team member has returned 5 of these drives. None lasted more than 9 months.

Reported by: David Dawes <`dawes@rf900.physics.usyd.edu.au`> (9 09)

5.7.19 Hewlett-Packard HP 35480A

The boot message identifier for this drive is `HP HP35480A 1009 type 1 removable SCSI 2 Sequential-Access density code 0x13`.

This is a DDS-DC tape drive. DDS-DC is DDS-1 with hardware data compression. DDS-1 is the original DAT tape format.

Native capacity is 2GB when using 90m tapes. It cannot handle 120m tapes. This drive supports hardware data compression. Please refer to the section on HP C1533A for the proper switch settings.

Data transfer rate is 183kB/s.

This drive is used in Hewlett-Packard's SureStore 5000eU (http://www.dmo.hp.com/tape/sst5000.htm) and 5000i (http://www.dmo.hp.com/tape/sst5000.htm) tape drives and C35480A DDS format DAT drive..

This drive will occasionally hang during a tape eject operation (`mt offline`). Pressing the front panel button will eject the tape and bring the tape drive back to life.

WARNING: HP 35480-03110 only. On at least two occasions this tape drive when used with FreeBSD 2.1.0, an IBM Server 320 and an 2940W SCSI controller resulted in all SCSI disk partitions being lost. The problem has not be analyzed or resolved at this time.

5.7.20 Sony SDT-5000 (http://www.sel.sony.com/SEL/ccpg/storage/tape/t5000.html)

There are at least two significantly different models: one is a DDS-1 and the other DDS-2. The DDS-1 version is `SDT-5000 3.02`. The DDS-2 version is `SONY SDT-5000 327M`. The DDS-2 version has a 1MB cache. This cache is able to keep the tape streaming in almost any circumstances.

The boot message identifier for this drive is `SONY SDT-5000 3.02 type 1 removable SCSI 2 Sequential-Access density code 0x13`

Native capacity is 4GB when using 120m tapes. This drive supports hardware data compression.

Data transfer rate is depends upon the model or the drive. The rate is 630kB/s for the `SONY SDT-5000 327M` while compressing the data. For the `SONY SDT-5000 3.02`, the data transfer rate is 225kB/s.

In order to get this drive to stream, set the blocksize to 512 bytes (`mt blocksize 512`) reported by Kenneth Merry <`ken@ulc199.residence.gatech.edu`>.

`SONY SDT-5000 327M` information reported by Charles Henrich <`henrich@msu.edu`>.

Reported by: Jean-Marc Zucconi <`jmz@FreeBSD.org`>

5.7.21 Tandberg TDC 3600

The boot message identifier for this drive is `TANDBERG TDC 3600 =08: type 1 removable SCSI 2`

This is a QIC tape drive.

Native capacity is 150/250MB.

This drive has quirks which are known and work around code is present in the SCSI tape device driver (st(4)). Upgrading the firmware to XXX version will fix the quirks and provide SCSI 2 capabilities.

Data transfer rate is 80kB/s.

IBM and Emerald units will not work. Replacing the firmware EPROM of these units will solve the problem.

Reported by: Michael Smith <`msmith@FreeBSD.org`>

5.7.22 Tandberg TDC 3620

This is very similar to the Tandberg TDC 3600 drive.

Reported by: Jörg Wunsch <`joerg@FreeBSD.org`>

5.7.23 Tandberg TDC 3800

The boot message identifier for this drive is `TANDBERG TDC 3800 =04Y Removable Sequential Access SCSI-2 device`

This is a QIC tape drive.

Native capacity is 525MB.

Reported by: Julian Stacey <jhs@FreeBSD.org>

5.7.24 Tandberg TDC 4222

The boot message identifier for this drive is `TANDBERG TDC 4222 =07` `type 1 removable SCSI 2`

This is a QIC tape drive.

Native capacity is 2.5GB. The drive will read all cartridges from the 60 MB (DC600A) upwards, and write 150 MB (DC6150) upwards. Hardware compression is optionally supported for the 2.5 GB cartridges.

This drives quirks are known and pre-compiled into the SCSI tape device driver (st(4)) beginning with FreeBSD 2.2-CURRENT. For previous versions of FreeBSD, use `mt` to read one block from the tape, rewind the tape, and then execute the backup program (`mt fsr 1; mt rewind; dump ...`)

Data transfer rate is 600kB/s (vendor claim with compression), 350 KB/s can even be reached in start/stop mode. The rate decreases for smaller cartridges.

Reported by: Jörg Wunsch <joerg@FreeBSD.org>

5.7.25 Wangtek 5525ES

The boot message identifier for this drive is `WANGTEK 5525ES SCSI REV7 3R1` `type 1 removable SCSI 1` `density code 0x11, 1024-byte blocks`

This is a QIC tape drive.

Native capacity is 525MB.

Data transfer rate is 180kB/s.

The drive reads 60, 120, 150, and 525MB tapes. The drive will not write 60MB (DC600 cartridge) tapes. In order to overwrite 120 and 150 tapes reliably, first erase (`mt erase`) the tape. 120 and 150 tapes used a wider track (fewer tracks per tape) than 525MB tapes. The “extra” width of the previous tracks is not overwritten, as a result the new data lies in a band surrounded on both sides by the previous data unless the tape have been erased.

This drives quirks are known and pre-compiled into the SCSI tape device driver (st(4)).

Other firmware revisions that are known to work are: M75D

Reported by: Marc van Kempen <marc@bowtie.nl> `REV73R1` Andrew Gordon <Andrew.Gordon@net-tel.co.uk> `M75D`

5.7.26 Wangtek 6200

The boot message identifier for this drive is `WANGTEK 6200-HS 4B18` `type 1 removable SCSI 2` `Sequential-Access density code 0x13`

This is a DDS-1 tape drive.

Native capacity is 2GB using 90m tapes.

Data transfer rate is 150kB/s.

Reported by: Tony Kimball <alk@Think.COM>

5.8 * Problem drives

6 CDROM drives

Contributed by David O'Brien <obrien@FreeBSD.org>. 23 November 1997.

Generally speaking those in *The FreeBSD Project* prefer SCSI CDROM drives over IDE CDROM drives. However not all SCSI CDROM drives are equal. Some feel the quality of some SCSI CDROM drives have been deteriorating to that of IDE CDROM drives. Toshiba used to be the favored stand-by, but many on the SCSI mailing list have found displeasure with the 12x speed XM-5701TA as its volume (when playing audio CDROMs) is not controllable by the various audio player software.

Another area where SCSI CDROM manufacturers are cutting corners is adherence to the SCSI specification. Many SCSI CDROMs will respond to multiple LUNs for its target address. Known violators include the 6x Teac CD-56S 1.0D.