1617: SCSI Hard Disk Supplement

Version 1.139
August, 1993

Applix 1616 microcomputer project
Applix Pty Ltd

1617 SCSI Hard Disk Supplement

Original version disk code by Andrew Morton.
All hard disk and hard disk utility code by Mark Harvey, who continues to update the software.
Original version of this manual was written by Mark Harvey and Kathy Morton.
Additional introductory and tutorial material by Eric Lindsay, who edits and fancy prints the manuals.

Comments about this manual or the software it describes should be sent to:

Applix Pty Limited
Lot 1, Kent Street,
Yerrinbool, 2575
N.S.W. Australia
(048) 839 372

ISBN 0 947341 xx x

1
Introduction

This is a supplement to the *Disk Co-processor Manual*. It is supplied with the hard drive kit, which is available from Applix for $99.

The hard drive kit includes the few components required to add a SCSI controller to your SSDCC Disk Co-processor. That is, the NCR5380 SCSI chip, the termination resistors, and the 50 way connector. Applix also have available a 50 way cable, and various brands of SCSI hard disk.

More importantly, the hard disk kit includes the EPROM containing Mark Harvey's additional SCSI commands, which are detailed in the last chapter of this manual.

Also provided is a Hard Drive Users Disk containing additional hard drive utility programs. These make it considerably easier to format and control your hard disk.

The additional SCSI commands and utilities provided in the SSDCC Version 2.n update, and connection of the SSDCC to hard disks, are covered in full in this supplement. It also covers construction of the SCSI interface, and gives background information on hard drives and SCSI.

1.1 SCSI

The SCSI (Small Computer Systems Interface) standard is a specification for communicating between a maximum of 8 computers and peripheral I/O devices, over a high speed 50 wire daisy chained bus. The SCSI bus appears quite frequently in medium performance microcomputer systems, and is now often used in personal computers, including the Apple Macintosh, Atari ST and Commodore Amiga. It evolved from the SASI (Shugart Associates Systems Interface) disk controller interface developed by Shugart Associates in the late 1970's.

The actual SCSI protocols are very, very detailed, and are described in full in the ANSI X3T9.2 SCSI specification. This should be obtainable from X3 Secretariat, Computer and Business Equipment Manufacturers Association, 311 First Street. NW, Suite 500, Washinton, D.C. 20001, USA (but we haven't tried to obtain it!)

SCSI drives are very standard in design and in their interface. All SCSI drives tested have run immediately on the Applix 1616. The only non-standard drives encountered are those designed for the Apple Macintosh, which differ from the ANSI standard by using a DB25 connector instead of the correct 50 way IDC connector, and also by not supplying power for the termination resistors at the connector. A simple adaptor cable usually fixes these drives.

The SCSI protocols are more than simply a means of running a hard disk. They support both single and multiple hosts (for example, an Applix and a Microbee and a Macintosh could all use a single SCSI bus). They support multiple peripheral devices, of different types. These include hard drives, SCSI floppy drives, tape backup units, and CD ROM players. The bus also supports high speed (up to 1.5 megabyte a second) host to host communication, so an Applix could communicate with, say, a Macintosh. In short, although we do not make full use of it as yet, it is a very versatile addition.

Hard disk drives are available which have an SCSI bus interface; host commands, data and error information are passed between the controlling computer and the disk drive electronics over this bus.

1.2 NCR5380

The NCR 5380 IC is designed for interfacing microprocessors to the SCSI bus. It incorporates line drivers and receivers and so yields a single-chip solution to the 1616's need for a hard disk interface.

The 5380 is located in the Z80 I/O space, and is controlled by reading and writing eight internal registers. The /SCSICS signal is used to select the required register. The normal Z80 /IORD and /IOWR read and write signals are used to select the data direction. Reset comes directly from the 1616 expansion connector.

The 5380 has sufficient drive output to directly run a SCSI bus, and requires only external 330Ω and 220Ω termination resistors. As always with diasy chained buses, a similar termination is required at the other end of the bus. Although DMA handshake is available on the 5380, no DMA chip is present on the controller. Pseudo DMA operation can be done using /SCSICS and A3 to generate /DACK.

The 5380 can optionally interrupt the CPU, via the SCSIRQ line, taken to a jumper block. This option is used in Disk Co-processor firmware with *odd* version numbers. That is, Versions 2.1, 2.3 etc., use interrupts, and require alterations to your Disk Co-processor card jumper block. Versions 2.0, 2.2, etc., do not use interrupts, and do not require alterations.

1.3 Commands

Your hard disk works in exactly the same manner as any other block device (such as /F0), and obeys precisely the same 1616/OS commands. The block device name for your hard drive is /H0, (and possibly also /H1, /H2, etc). These names can preceed any absolute path.

In addition, the Interprocessor Communication commands described in your *Disk Co-processor Manual* are all supported. These are the Block Read, Block Write, Display Error Code, and so on, using command bytes $00 to $0A.

A number of additional SCSI related commands are provided. These are of two types. Some provide the support required for hard disk operation, such as Format SCSI, Sense Condition (for error handling), and Start/Stop SCSI (to safely park the heads prior to moving a drive). Others were originally developed by Mark Harvey during the course of writing the SCSI software, and are made available for experimenters, or for testing your drive. These include Re-zero SCSI, and Reset SCSI Bus. The Send SCSI Command lets you send any SCSI command you are capable of writing.

1.4 Utilities

A number of utility programs are included on your Hard Drive Users Disk (provided with the hard drive update kit). These ease the task of setting up your new hard drive.

They include a hard drive configuration program, for formatting your drive more easily. The `blockdev` provided will initialise (but not format) your newly formatted hard drive. You can read the sense information provided by your hard drive after an error. You can park your hard disk heads, restart the drive after parking, experiment with various sector skew schemes, and read the error code messages provided. If all you wish to do is format and prepare a new hard disk drive, the instructions for using the utility programs to do so are in the Utilities section of this manual.

1.5 Disk sub-system

A SCSI hard disk system consists of:

- A SCSI type hard drive, or
  SCSI adaptor and ST506 (IBM) hard drive.
- A cable to the disk drive controller (a 50 way ribbon cable, with IDC connectors, is available from Applix).

- A power supply to run the drive. The Apple style power supply normally used by Applix can sometimes be used, with a 4 way cable, available from Applix.
- A drive controller chip, and address select electronics. Applix use the NCR 5380 chip on the disk controller card.
- A computer to control the drive controller chip. This is the Z80 on the disk controller card.
- Software to control the drive controller chip at a low level. This is contained in Mark Harvey's Version 2.n EPROM supplied with the hard drive upgrade kit.

1.6 How it works

68000 in Applix 1616 issues high level disk commands such as *block read*, *block write*, *format*, etc. as a single command code byte sent to $FFFFD1. All other parameters (such as block number) are sent to $FFFFC1. Data is sent to and received from $FFFFC1. This is essentially identical to floppy use, but there are extra commands.

Z80 in controller card receives command, parameters and data at I/O address $18. Software in EPROM converts block numbers, issues appropriate command to NCR5380 chip. NCR can address multiple (up to 8) SCSI devices, each of which can control 1, 2, or 4 or 8 drives, depending on model.

NCR5380 chip issues relatively high level commands to SCSI device, and controls high speed parallel bus for read and write data, passing it via the Z80 to SDATA port at I/O address $18.

Drive is connected via 50 way control and 4 way power cable to disk controller. Responds to commands on parallel SCSI bus. Reads and writes sectors upon command. Various drives may obey a variety of relatively sophisticated and complex commands.

2
Construction

Construction of the hard disk interface is very easy, for anyone who has built a complete 1616. There are four components to be soldered into place on the SSDCC disk co-processor card. The components are supplied with the hard disk upgrade kit.

These components are the 50 way dual 0.1 right angle connector. There are also two termination resistor packs. One is 220Ω, the other is 330Ω. These components **must** be inserted the correct way round, with the small dot on the resistor pack corresponding to the dot (common connection to either power or ground) on the PCB.

A forty pin IC socket, for the NCR5380 supplied with the kit. Remember that you must ensure that the socket and chip are plugged in the correct way round (the small indentation at the end corresponding with the similar mark on the disk card). Remember also to observe static precautions (keep yourself and your work area grounded) with the NCR5380, as it is liable to damage by static charges.

2.1 Eprom upgrade

The eprom on your SSDCC disk co-processor card must be upgraded from Version 1.4 to Version 2 or greater. This eprom is also supplied with the upgrade kit.

Version 2.0, 2.2 and other *even* version numbers do not require any other alterations to the disk card.

Note. Considerable changes were made to the SCSI code between versions 2.0 and 2.2. You *must* have an appropriate version of the hard disk utilities disk. That is, early version utilities may not work correctly with late version eproms. If you update one, update the other at the same time.

Versions 2.1, 2.3, and other *odd* version numbers use interrupts, and are about 5% faster in operation. If using these versions, link pins 3 and 5 of the interrupt pin connectors on the disk card.

2.2 Cabling

You require a 50 way ribbon cable between the disk card, and the SCSI drive. These are best made using insulation displacement (25 x 2 IDC) connectors. Do not make the cable too long (a few feet is generally adequate, although the standard allows much more). Applix sell a suitable, tested cable for about $30.

Note Pin 25 (termination voltage) on the connector may have to be disconnected from the +5 volt line by cutting between two pads ('A' and 'B') on the disk card. See the disk card construction manual for details, if this is not already done. Note also that although this power connection may damage some drives, others will not work without it being present (nice trick that!)

2.3 Power supply

The biggest problem with SCSI drives is the power supply. The standard Apple power supply used by Applix is rated at between 35 and 50 watts. Like most el-cheapo switch mode power supplies, it seems to depend somewhat on your luck (and possibly the phase of the moon) as to how much power you can draw from them. Incidently, I'm not being nasty in calling the power supply an 'el-cheapo'. A suitable industrial quality power supply from an Australian company will cost about $400, which tends to explain why most builders use the Apple clone supplies.

If you buy a 3.5 inch SCSI drive, it will draw between 6 and 12 watts, and will run fine from the Apple supply (albeit leaving the supply rather warm).

If you buy a 5.25 inch SCSI drive, it will draw between 10 and 20 watts. Some of the larger drives will run on some of the Apple supplies. Others simply can not do so. The major problem is usually the large 12 volt motor starting current. In most cases, even if the 5.25 inch drive will start, it tends to draw so much power that the power supply runs overly hot after several hours.

If you use a SCSI adaptor, and an IBM style ST506 hard disk, there is no way in creation you can run it from the standard power supply. That combination invariably draws several amperes too much for the Apple clone supplies.

There are several styles of power supply that can be used as replacements for the normal one. Statronics and other Australian companies have suitable industrial quality switch mode supplies, however they are expensive. Applix can supply several different model power supplies with higher than normal ratings.

Finally, you can use an IBM clone power supply. These are generally relatively cheap (round the $150 area), and have rated outputs of about 150 watts (this rating is usually fairly fanciful, but you really probably only need 75 watts, so it doesn't matter). My impression is that they tend to be badly made, and tend to use capacitors that are marginally rated for the voltage involved. However you normally get several years life from them. Two disadvantages are they don't fit in an Applix case, and the fans in them are noisy.

2.4 Connection

The SCSI drive requires a power connection (standard four pin floppy connector, or mini four pin for 3.5 inch drives). You must ensure the polarity of the connection is correct, or you may destroy your drive..

Connect the 50 way cable, again ensuring the correct polarity.

2.5 Starting your hard disk

Power up your 1616. If the SSDCC disk card hangs (that is, if there is no version number reported on screen, or you get a time out error) then recheck your SCSI cable, and check that the 'A' 'B' board cut has been made (if required) on your drive card.

- If your hard disk came from Applix, you can ask to have it formatted and the software installed, ready to go.

  In this case, your 1616 should now boot automatically from the hard disk, and is ready to use. You need not do anything else.

- If you have an Adaptec or Xebec SCSI adaptor, driving an IBM style ST506 drive, then you must set up your drive manually. First run the `adaptec.xrel` or `xebec.xrel` program from the utility disk. This is required to set up your ST506 hard disk as if it were a SCSI drive. Then follow the instructions below for normal SCSI drives.

- If your SCSI drive did not come ready to use from Applix, you must run `hdconfig.xrel`. Select Option 0 to format your drive.

- After formatting the drive, select option 2, to configure the partition sizes for your drive. 1616/OS Version 3 limits hard disk sizes to 16 megabytes, in two partitions. We suggest upgrading to Version 4. 1616/OS Version 4 allows up to 40 megabytes for the first (/H0) partition, and 8 megabytes for the second (/H1) partition. If you need larger and/or extra partitions, you must use an extra block driver in your `mrdrivers` file. More details of setting up partitions are given in the next chapter, which describes each of the utility programs in detail.

Note that the partition table is stored on SCSI drive 0, logical unit number (LUN) 0. Therefore, this **must** be formatted first, and **must** be always available on the SCSI bus, otherwise you won't find your other partitions. Take a note of how many blocks you provide in each partition, as you need it when using `blockdev.xrel`.

- The next step is to set up boot blocks, directory structure and so on, using `blockdev.xrel` as normal. You do need to remember the number of blocks each partition contains.
- Since the bit map will not be re-read when you exit `blockdev.xrel`, you must do a **reset**, so that 1616/OS can read the bit map (for obvious reasons, bitmaps on non-removable disks are only read on reset).
- Now you can copy over your files from diskette, set up directories, and so on, as with a normal removable disk. See the next chapter for further details of each utility mentioned above.

3
Utility Programs for SCSI Drives

The hard drive users disk includes a number of handy utility programs to assist you in understanding, formatting, and using the SCSI hard disk on your Applix 1616. The use of these programs is briefly described in this chapter. Understanding precisely how they operate, and how to modify them, is best achieved by using them, reading the source code provided, and reading the detailed list of commands to which the SSDCC EPROM responds.

Since some of the programs operate on the hard disk while it is active, I suggest that you first copy those you wish to use onto your /RD ram disk, and operate them from there. You are particularly silly to operate a program like `shutdown` (which parks the hard disk), only to find that the corresponding `startup` program is only on the now parked drive!

If formatting a new drive using an Adaptec adaptor, you will use `adaptec`, `hdconfig`, and `blockdev`, in that order.

If formatting a new genuine SCSI drive, only `hdconfig` and `blockdev` are required.

You will need to know the parameters of the drive, so make sure the manual is handy. The first drive partition (/H0) can not exceed 40 megabyte, and the second (/H1) not exceed 8 megabyte (unless you provide an appropriate MRD program). If you wish to use more than two drive partitions, or if your drive exceeds 48 megabyte, you will need to understand how to produce (or possible modify) an MRD program (if that doesn't make any sense to you, you need to ask for help from Applix in preparing your drive).

The usual Mark Harvey warranty applies on all these programs: if you break them, you own both halves (that is why the source code is available ... just joking ... you can also contact Mark via Applix if changes are required). More seriously, we would like to hear from users. If you have a problem, please obtain as much information about your problem as you can (operating system version, drive details, what you were trying to do, etc.,) and contact Applix. If you don't tell us it is broken, we can't fix it!

3.1 Adaptec

The program `adaptec.c` is provided for those using the Adaptec ACB4000A ST506 to SCSI adaptor. This adaptor enables you to convert two IBM style ST506 hard disk drives into SCSI hard drives. The code allows you to setup and format these drives, and is run prior to using `hdconfig`. The same code will also work with Adaptec 4070 Run Length Limited (RLL) controllers, and the four drive Adaptec 5500 adaptor.

It will ask you about the following parameters, so you will need to find out the answers prior to using it. There is a lengthy list of typical drive parameters later in this manual, but these may not be totally accurate or complete.

SCSI I/D (usually 0).
Logical Unit Number (usually 0).
Number of Cylinders (read the drive manual).
Number of Heads (read the drive manual).
Cylinder for Reduced Head Current (default is 0 on early versions, which is a bit misleading, use Number of Clyinders for most drives).
Cylinder for Write Precompensation (use 1/2 the number of cylinders if unknown).
Landing Zone, tracks beyond last data cylinder (0 to 4 beyond the last data cylinder are usually safe).
Step Rate (use 0 for most recent drives).

Incidently, you can work out the number of blocks you have (as is needed for using `blockdev`) by multiplying the number of heads, by the number of cylinders, by the number of 512 byte sectors (17 works for most drives, 25 for RLL controllers like the 4070), and divide the whole by 2 (a block is two sectors).

3.2 Blockdev

This is the usual `blockdev` program, as modified for hard drives. It is used to initialise a hard disk partition that has already been prepared (formatted) and partitioned using `hdconfig`. If your drive has not already been formatted and partitioned, you will not get anywhere using `blockdev`, so check out `hdconfig` first. Also, check that your Hard Driver User floppy disk really includes a file `/f0/sys/bootv3`, because you need to know where that file is later.

Invoke it with `blockdev /H0` or `blockdev /H1`. You will note there is no longer a format option, as `blockdev` is unable to format a hard drive. To prepare a new hard drive, select option '1', to reinitialise disk.

You will be asked the number of blocks on the drive. This is the same value as given in `hdconfig`, so make sure you noted it down somewhere.

You are also asked if the device is removable. Unless you have very expensive tastes in removable drives (or own a SCSI tape backup), the answer is 'n'.

The remainding questions are identical to preparing a floppy disk. Since the storage capacity of a hard disk is considerable, you should seriously consider increasing the number of entries in your root directory from the standard 64 to 128 or so (you can't change this later without losing your data). As always, you can make the volume name of your drive anything you like. Being lazy, I stick to unimaginative names like /hard0.

If you want to make the hard drive the boot device, you must provide an appropriate boot program. A simple, typical boot code example is contained on your Hard Drive User Disk, usually in the `/sys` sub directory. It is usually called `bootv3`, or in older disks, `bootv3.exec`. Since `blockdev` insists on you providing full path details (usually `/f0/sys/bootv3`), you must check where this code is before invoking `blockdev`.

You **must** warm boot or reset after using `blockdev`, as the bitmaps for non-removable drives are not updated until then. If you do not, you will find that a directory listing will show vast numbers of blocks occupied.

Remember that, if you have partitioned your new hard disk into two partitions, you must also `blockdev /h1`. Also, if you get an error message, just run `blockdev` again, and it will usually work the second time (the error may even be spurious).

3.3 Findbad

A recent, non-destructive test for bad blocks. On SCSI drives, it reads the boot block, finds the partition size, and then reads the blocks looking for errors. Displays the block numbers of blocks that produce errors, but does not perform a sense command on the error.

It provides the logical unit number and block number of the hard disk, however the block number is the absolute number, not relative to the current partition, so be careful about interpretting that.

Works on floppy drives also, so `findbad /h1` is a typical correct usage, but `findbad /f0` will also work.

3.4 Hdconfig

This program configures a new SCSI hard drive. Your first action with a new, unformatted SCSI drive is to format it (unless you are using an ST506 style with an Adaptec or similar

controller, in which case run `adaptec` first). The format using `hdconfig` totally destroys all existing data, so don't do it if Applix or someone else has already set the drive up for you. Formatting a drive is started by selecting option 0. Mark's code does an enquiry on the drive, to determine how many blocks will fit on it (except on Omti 3100 SCSI adaptors, which can't answer that enquiry).

The interleave factor is next. At the moment, 3 is probably the best choice for interleave.

The list of format parameters is an unknown factor, since it depends entirely upon the drive. This tells the drive which sectors or tracks to format, and which to mark as bad. For Miniscribe 3425S types, answer 24 (that uses the manufacturer's inbuilt list of bad blocks). For other drives, see the manufacturer's drive manual (on some drives, the figure you give will be taken as the track number from which to start formatting the drive). If all else fails, use 0 as your answer.

The last prompt is the Logical Unit Number. For your first drive, this will be 0.

Incidently, `hdconfig` gave me strange responses, or appeared to have stalled for a few seconds. Just ignore such behaviour, since it is usually just waiting for a response from the drive. In some cases, the response arrives well after the drive card has timed out. Another minor problem is that a 15 MHz 1616 may timeout before the SSDCC when using this command. Doubtless it will eventually be altered to respond to real time instead. However, please note all the details of any problems you encounter, and report them to Applix, so that any problems can be fixed.

3.5 Hddvr.c

This program is the C code for an MRD driver for /H2 and /H3, if you want more than the default two drive partitions. The default size of each is 8 megabyte. Look up MRDs in the *Technical Reference Manual* before playing with it, and read the code first for notes on how to alter the default partition sizes. This really isn't appropriate for beginners, so if you are a beginner, and have a drive larger than 48 megabyte, it is best to consult Applix..

3.6 Readerror

No source visible for this one, but if you pass it an error number (e.g. `readerror 10`) it will provide a string describing that error. From context, the error strings appear to be from the SSDCC EPROM, version 2.2. These errors are listed elsewhere. These are all new error messages, and don't really correspond with the error messages from Version 1.4. Extra error messages will be added as required. These is a list in this manual, at the end of the last section.

3.7 Sdc_id

Merely displays error code 9, date of the assembly of code, so you can find out whether you have a sufficiently recent set of EPROMS.

3.8 Sense

This reads sense information from the Z80 SCSI command. In essence, it lets you find out what messages are coming back to the Z80 from the SCSI drive. How you interpret the sense command depends upon which SCSI drive you happen to own. Make sure you copy it to ram drive /RD *before* you run into an error, because it won't be real helpful otherwise! I'll try to get some typical meanings listed elsewhere in this supplement.

3.9 Shutdown

This command should park the heads on your hard disk, thus making it safe to move. You should always park the heads on your drives, as a precaution against accidental damage, prior to switching the system off. You will get very strange results if you run this command from the hard disk, so copy it to your ram drive /RD before you use it.

The program will prompt for the SCSI identification and Logical Unit Number. For a single drive, these are usually both 0. You can also supply both parameters from the command line, as `shutdown 0 0`.

Some drives, such as the Miniscribe 3425S, not only park the heads, but also switch off their drive motor entirely (very handy if you want some peace and quiet for a while). Use `startup` (from the ram drive) to start the motor.

3.10 Startup

The obvious complement to `shutdown`. The command `startup 0 0` will re-start drive 0 after it has been shutdown. The program will also prompt for SCSI identification and Logical Unit Number (both usually 0 for a first drive) if you do not provide them from the command line.

If using this command, remember to have it available on your ram drive or floppy disk, since it won't be much use if it is only on the hard disk!

3.11 Skew

One of Mark Harvey's experiments. Lets you experiment with a wide variety of skew factors on a drive, to see which is fastest. The parameters are the skew factor (generally 1 to 16), and the device number. Valid device numbers are 0 = /RD, 1 = /F0, 2 = /F1, 3 = /H0, 4 = /H1, etc.

3.12 Ver

Tells you the version number of both the 1616/OS, and of the Disk Coprocessor EPROM. Handy when you can't remember which EPROMs you have, and where, and don't want to check the hardware!

4
Hard Drives and Controllers

We will describe hard drives, and then outline how the disk controller chip works. This chapter can be considered as background information only. You need not read it if you simply want to use a hard disk.

4.1 Types of hard disk drives

Hard disk drives have long been used in mainframe computers, however the traditional type were large, fragile, and very expensive. This changed with the introduction of Winchester technology (the name reputedly comes from the IBM 3030 code number for this design). These drives used a permanently sealed chamber in which the magnetic coated aluminium disks were fixed. The sealed chamber eliminates the problem of contamination of the disk surface by dust and smoke particles, any of which could destroy a disk surface on older drives (a disasterous event known as a *head crash*).

Physical size

Drive sizes have been decreasing over the years. Where they formerly used 8 inch, or even 14 inch disks, 5.25 inches in now the usual size. The more recent 3.5 inch drive is increasingly common. The 3.5 inch drive has the advantage of consuming less power (which means you are more likely to be able to run it from your existing Applix power supply), but tends to be more expensive. There is some suggestion that decreasing sizes reduce reliability, however this factor is probably not important.

Most drives are now ‘half height’, or about 3 centimeter thick. Drives that are ‘full height’ are generally low capacity, and very old (old in computer terms tends to mean last years model!) Unless you get excellent prices, avoid full height drives. The exception to this are some very high capacity drives, where including a large number of heads (over 6) has resulted in a thicker package.

Access speed

Hard disk drives can also be divided into two speed ranges, fast and slow. This indicates how the read/write head is moved across the magnetic drive surface.

The fast drives, with average access times of 20 milliseconds (mS) to 40 mS, use a voice coil mechanism. This is something like a powerful loudspeaker mechanism, and it can move the read/write head very rapidly. In general, voice coil (fast) drives cost twice as much as stepper motor (slow) drives.

The slower types use a stepper motor, just like most of the floppy disk drives, and have an average access time of perhaps 60 mS to 100 mS. Whether this is important tends to depend upon whether your operating system can make use of the extra speed (at the moment, the Applix 1616 does not), and also whether you are doing a lot of random access to the disk. Random access mainly occurs in data base work, so you might as well stick with the cheaper, slower, stepper motor drives on the Applix.

Disk capacity

Drive capacities range from as little as 5 megabyte to a gigabyte. The common, and cheap, sizes at the moment are 20 megabyte and 40 megabyte. Capacity is generally determined by the number of disk surfaces (and thus the number of heads) in a drive, and the number of tracks. You would not see drives of more than 16 heads and 2048 tracks. Typical drives have 4 or 6 heads, and 600 to 1000 tracks.

Increasing either number of heads or number of tracks increases the costs and the delicacy of a drive. Despite this, users are generally advised to obtain the largest drive they can readily afford. In particular, do not buy less than a 20 megabyte drive, as you will soon regret it. At the moment, the Applix by default can use up to 48 megabytes, or with an unaltered MRD, 64 megabyte. It probably isn't worthwhile attempting to obtain drives substantially over this capacity, although you can alter the system to use them, if they are available to you.

4.2 Drive mechanics

The drive is rotated by a high speed DC motor, while all the heads are simultaneously moved over the disk surface by either a stepper motor (slow) or voice coil (fast and expensive). The drive electronics can read from only one head at a time, but there is a read/write head above and below each disk surface. On some high capacity (very expensive) drives, one disk surface is reserved for accurate positioning of the heads, but this will not be noticeable to you (except that the drive specifications will list an odd number of heads).

Unlike floppy disk drives, the read/write head **should never** touch the surface of the disk while it is rotating. At 3600 rpm, the head would scratch into the surface and possibly destroy the drive. The head is shaped so it 'rides' very close above the disk surface due to the aerodynamic effect of the rotation of the disk. When power is removed, either the head must be retracted out of the way, or it must be 'parked' on a safe area of the drive surface (usually on one of the most inner tracks).

Since the actual size of the disk drive approximates that of a floppy disk drive, but the number of tracks is much greater (up to 1000 typically, vs 80), it is obvious that hard disk tracks are very much narrower. Likewise, the mechanism for positioning the head must be far more precise, or have some self correcting mechanism. Luckily, since disks are not exchanged, drives don't have to be identical in their track to track spacing. Also, the drive spindle is fixed to the actual disk, which greatly assists accurate positioning.

Rotational speed

All drives now rotate at about the same speed, typically 3600 rpm, 60 times a second, or once every 16.67 mS. Hard disks usually provide sector sizes of 128 bytes, 256 bytes, 512 bytes, and 1024 bytes. As some brands of drive can not manage 1024 byte sectors, Applix uses 512 byte sectors internally. Software arranges that these are grouped in 1k blocks, as required by the operating system.

4.3 Drive electronics

In general, a SCSI drive can be treated as a 'black box'. You supply it with about 2 amps at +5 volts for the electronics, and about 2 amps at +12 volts for the drive motor (smaller drives may draw substantially less), via the standard size 4 way power connector. Motor speed control is invariably electronic, and not adjustable. Some drives are able to enter a low power mode, by shutting down the drive motor under software control.

The 50 way SCSI connector includes an 8 bit parallel bi-directional data bus, using even pins 2 to 16. The optional parity line is invariably ignored. The odd pins are ground lines, except that pin 25 is sometimes used for termination. The drive is controlled by pins 32 (attention), 38 (acknowledge), 40 (reset), and 44 (select). It responds using pins 36 (busy), 42 (message), 46 (control/data), 48 (request) and 50 (input/output). The reason this scheme appears 'reversed' is that SCSI drive are 'intelligent'. Once the drive is 'awake', it controls the data transfers, and the host responds as appropriate (even if the response is to send another command).

The only jumper changes required to SCSI drives are the drive select number (which for a single drive is invariably 0, or no jumper installed), and termination (which is almost always already installed). Other than that, don't fiddle with the drive!

4.4 Disk speeds

A typical SCSI hard disk using standard MFM recording fits 17 sectors of 512 bytes, or 8704 bytes per track. If it rotates at 3600 rpm, a track takes 16.67 mS to pass under the read/write head. This means the 'rotational latency' (time for the start of the next track to come round) will average half that, or a bit over 8 mS. We also have to allow time to work out which track we need next. We then have to move the read/write head to that track. Stepper motor drives, such as the Shugart SA600 series from 1983, or the Miniscribe 3425 from 1986, take 15 to 17 mS to move from one track to the next, but more recent drives tend to be much quicker.

If the drive controller uses run length limited recording (27 RLL), then it fits 25 sectors of 512 bytes per track, and since all other actions are identical, obviously the data transfer rate on and off the disk is higher.

What is the theoretical maximum speed we can attain from our SCSI drive? A whole bunch of conditions have to be right to get the maximum possible speed. These include perfect inter-leaving, 'zero latency', pre-fetch caching, and spiral skews, which are typically only used in expensive mini-computer systems.

One very big factor is the interleave we use. The interleave is the arrangement of sectors on the tracks. Ideally, the interleave should be 1 to 1. That is, sectors 1, 2, 3, and so on should follow each other sequentially on the track. Provided everything is fast enough, you can then start reading sector 2 as soon as you finish with sector 1, and so on. However, if your system (either hardware or software) can't manage to immediately read the following sector, you would have to wait a whole revolution (plus one sector), or nearly 18 mS, for it to come round again. If this happened for every sector, it would take you 18 revolutions (a third of a second) to read one track of only 8.5 kb. Very, very slow.

Interleave factor

The solution lies in the arrangement of the sectors. If it takes you, say, almost two milliseconds to deal with a sector after reading it, then there is time for almost two more sectors to pass under the read write head. If you arrange that logical sector 2 is is not physical sector 2, but rather sector 4, and that logical sector 3 is physical sector 7, and so on, you can then read logical sector one after the other without much of a wait for rotation of the disk. This arrangement, of logical sectors in sequence every third physical sector is (not surprisingly) called 3 to 1 interleave. The actual sector sequence on each track is then 1, 7, 13, 2, 8, 14, 3, 9, 15, 4, 10, 16, 5, 11, 17, 6, 12. As you can see, it now takes three revolutions of the disk to read a whole track, but even this is better than 18 revolutions.

Obviously it is important to keep the interleave as low as possible. Equally obviously, this is highly dependent upon how fast you can actually deal with each sector as you get it from the disk. You can experiment, but an interleave of three usually works best for SCSI drives on the disk controller card.

Zero latency

Given one to one interleave, it would be nice to to arrange 'zero latency' reads and writes. Traditionally, and because it is a damn sight easier to arrange, a disk controller waits until the first sector required comes round before starting to read or write. This means reads or writing a whole track will take, on average, one and a half revolutions, or about 25 mS from the time the head reaches the correct track. However, if we can simply start reading or writing whenever **any** sector of interest occurs, then we will never require more than a single revolution (16.67 mS) to read or write any track. This means that our disk handling software has to offset into the material we are reading or writing to disk, and arrange that it all end up in the correct sequence, even though it wasn't read in sequence. As I mentioned, a tricky bit of programming, and one we didn't try.

Prefetch caching

Prefetch caching implies using some memory as a buffer. Whenever a read request is received, you continue to read from the disk, past the end of the requested data, until you run out of buffer space. If the next read request is for a sector contiguous to the previous ones, you will already have it in memory, and be able to fill that request without a time wasting read from the disk. Again, some tricky programming is needed. About the only caching we do is directory caching (which you can optionally alter).

Spiral skewing

Spiral skewing is done when formatting a disk. If you have 17 sectors per track, then it will take about a millisecond for each track to pass under the head. If it takes, say, 3 milliseconds to move from one track to the next, then it makes sense to have each track formatted so that sector zero is slightly more than 3 sectors past the angular position of the previous track's sector zero. Then, if writing anything contiguously from track to track, you would not have to wait for the rotational latency, even if you didn't have software that could handle zero latency reads and writes. You might notice that there is a skew factor on floppy disks, for interleave purposes, however it is the same from track to track, not spiral skewed.

Theoretical speeds

Okay, if we assume all that, what is our theoretical maximum speed? If we have instantaneous track to track times, one to one interleave, spiral skewing, and no errors, 510k per second for MFM recording, and 750k per second for RLL. However, real disk drives take some time to step from track to track. The stepper motor kinds are specified at round 15 mS. This reduces our theoretical speeds to round 268k for MFM, and 394k for RLL.

We probably also can't handle things fast enough not to need to interleave. At a 3 to 1 interleave, the theoretical top speeds drop to 130k per second for MFM, and 192k for RLL. These appear to be approximately in line with 1985 theoretical figures for an IBM XT using 6 to 1 interleave giving 104k per second, while an IBM AT on 3 to 1 interleave gave 208k per second, provided you didn't change tracks.

Measured speeds on various computers

A top of the line Everex 80386 based IBM clone, with caching on an RLL voice coil drive using 1 to 1 interleave could read a one megabyte file in 7.03 seconds, giving 142k per second. Interestingly, 2 to one interleave slowed it only to 138k per second. More recent versions of MS-DOS appear slightly faster. Although the XT models use DMA for disk access, in the AT model, DMA appears not to be used (probably because IBM's DMA is slower than doing it using the CPU).

Using an Apple Macintosh, and a voice coil drive with RLL controller on 3 to 1 interleave produced consistent averages of 210k per second on contiguous files of 227k and 379k. A file of 803k, not contiguous, but scattered over three areas of the disk, produced average speeds of 133k per second. The Macintosh keeps directory information in memory, which would aid access. Recommended interleaves are improved to 2 to 1 in the Mac SE, and 1 to 1 in the top of the line Mac II. As the Macintosh does not appear to have DMA, there is probably some cute code for the 68000 in there. Interestingly, the Mac also managed 25k per second from a floppy drive.

Matthew Geier did a number of tests on the Applix 1616 using 1616/OS Version 3 in October 1988. Using a cheap ST506 Miniscribe 3650 stepper drive, and an Adaptec 5500 SCSI to ST506 adaptor, and the `coke.snd` file of 318,464 bytes, he had results of 19k per second using SSDCC Version 2.0, and 27k per second using SSDCC Version 2.1, which uses Z80 interrupts. These results aren't very encouraging, but you should note that a 1 to 1 interleave was used, which is not a sensible choice.

Mark Harvey, who provided the hard disk drivers, set out to find likely causes. His tests looked at the Z80 side and interleave factors, not the transfer to the 68000 (which was considered separately). They showed that running SSDCC Version 2.1 gave poor results (56k per second) on 1 to 1 and 2 to 1 interleaves. Best results (126k per second) were obtained on interleaves of 3 and 4, with larger interleaves getting slower (but still not as bad as 1 to 1).

If the Z80 code were totally altered to use 1024 'INI' instructions (the fastest software method available), the speed from disk could reach a theoretical maximum of 350k per second, while the Z80 to 68000 theoretical maximum was 175k per second, to produce a maximum combined speed of 125k per second. Actual measurements on the Z80 using the 'INI' code produced up to 240k per second on 2 to 1 interleave (1 to 1 was same as with Version 2.1). The 3 to 1 interleave was up to 160k per second, and all larger interleaves were slower. This means that, for the moment, a 3 to 1 interleave is recommended.

Andrew Morton put the SCSI controller direct on the 68000 bus, and used multiblock code (now provided as standard in 1616/OS Version 4) to load the 311k `coke.snd` file in 2.08 seconds, or 149k per second. The new memory board, currently available for $600, has provision for running a high speed SCSI port of this nature.

4.5 SCSI controller chips

The 40 pin 5380 SCSI controller chip used in the Applix 1616 was devised by NCR and is now also made by Advanced Micro Devices. It was one of the first such specialised, one chip, SCSI interfaces, and is widely emulated by later designs. Being cheap, reliable, and widely available, it is ideal for a do-it-yourself computer.

The 5380 SCSI chip appears to the programmer as complicated parallel peripheral chip, controlled by a set of eight registers. When operating, the 5380 acts as an initiator device. That is, it requests a target device (the actual SCSI disk drive) to perform an operation, such as reading a disk block. The protocol used can be broken into phases, which are described below. This level of detail is required only if you are writing your own low level drivers.

4.6 Bus phases

The protocol interplay between initiator and target is broken into five possible phases, of which the Applix 1616 uses only three.

Bus free phase is when there are no active devices on the bus.

Arbitration phase is the time during which it is determined which of multiple initiating devices is to actually gain control of the bus. Since there is only one ititiating deive (the Applix 1616), this is not normally used.

Selection phase is when an initiating device (the Applix 1616) selects the target device (SCSI drive 0 or 1, or whatever), and waits for an acknowledgment from the target.

Optional reselection phase gives the target an opportunity to reconnect to the initiator after it has suspended or disconnected, during an operation previously requested by the ititiator. Thus, if the drive is doing something that will take a while (perhaps a format), it can disconnect, and allow the initiator to control, say, another drive. This is not used in the Applix 1616.

Information transfer phases are the times when data is actually being moved across the bus between the initiator and the target. Three SCSI signals (Input or Output, Command or Data, and Message are used to determine these phrases, so there are eight possible transfer phases, of which two are unspecified and unused. The remaining six transfer phases are Command, Status, Message In, Message Out, Data In, and Data Out. Note how these match to the nature of the actual commands you can give the SCSI drive.

Bus phases	I/O	C/D	/Msg
Data Out	0	0	0
Unspecified	0	0	1
Command	0	1	0
Message Out	0	1	1
Data In	1	0	0
Unspecified	1	0	1
Status	1	1	0
Message In	1	1	1

4.7 SCSI Registers

The 5380 SCSI chip registers are accessed within the Z80 I/O address space as set out below.

Address	Read	Write
$20	SCSI Data	Output Data
$21	Initiator Command	
$22	Mode	
$23	Target Command	
$24	SCSI Bus Status	Select Enable
$25	Bus and Status	Start DMA Send
$26	Input Data	Start DMA Target Receive
$27	Reset Parity/Interrupts	Start DMA Initiator Rec

Data registers

A number of the 5380 registers are reserved for data of various types. These are grouped and described below.

Register 0 ($20): SCSI Data Register. A read only register from which the Z80 can read the active SCSI bus data. Used during I/O data reads, and during arbitration to determine the priority of the arbitrating devices.

Register 0 ($20): Output Data Register. Write only register, used to send data to the SCSI bus, and to assert ID bits during arbitration and selection phases.

Register 6 ($26): Input Data Register. Used to read data latched during DMA or pseudo DMA operations.

Register 1 ($21): Initiator Command Register. A read write register used to assert, de-assert, and monitor the SCSI bus signals /RST, /ACK, /BSY, /SEL and /ATN, in bits 7, 4, 3, 2 and 1. Bit 0 of this register is used to assert or enable the data bus, while bits 5 and 6 are used to monitor arbitration. Bit 5 indicates the device has lost arbitration due to another device asserting /SEL during the arbitration phase, and bit 6 indicates arbitration in progress.

Register 2: Mode register. This enables block mode DMA transfers, determines if the device will be a target or an initiator, enables parity, and determines if interrupts are generated for parity error, DMA end of process, or loss of /BSY, in bits 7, 6, 5, 4, 3 and 2. DMA transfers are enabled by bit 1, and bit 0 starts arbitration.

Register 3: Target command register. If the device is connected as a target (a drive), the microprocessor can control the SCSI bus information phase using bits 0, 1 and 2. These assert I/O, C/D, and /MSG. /REQ can also be asserted by setting bit 3.

If the device is connected as an initiator (the controller), the I/O, C/D and /MSG bits must match the corresponding bits in the SCSI Bus Status Register (4) in order for data to be sent. In DMA mode, the Target Command Register is also used in the generation of a phase mismatch interrupt. This interrupt is generated when /REQ goes active, if the I/O, C/D and /MSG lines do not match the corresponding bits of the Target Command Register.

Bit 7 of this register is used to determine when the last byte of a DMA transfer is sent to the SCSI bus.

Register 4: SCSI bus status. A read only register used to monitor the SCSI bus signals /RST, /BSY, /REQ, /MSG, C/D, I/O and /SEL, and the data bus parity bit.

Register 5: Bus and status registers. A read only register used to monitor the remaining SCSI signals. The main ones are /ATN and /ACK (bits 1 and 0).

4.8 Controller commands

5
Choosing Your Drive

There are a number of factors to consider in choosing your drive. Obviously, a major factor is cost, and what equipment you already own. Another is the speed you require for your particular application. Extra speed will cost money.

The Applix 1616 SCSI interface is totally standard, and should work with any SCSI device. However, many SCSI device manufacturers have made alterations to the commands their device obeys, so that it becomes a superset of the standard SCSI command set. Obviously, these extra commands may not be supported by standard Applix 1616 SCSI code.

Many SCSI devices are sold without any detailed information or manuals. Even though the Applix 1616 has provision to allow any SCSI command to be generated and sent to the SCSI device, if you can't obtain a manual, you will not be able to make use of this facility. For example, although the 1616 will talk to SCSI tape drives, Mark had to battle for a long time before he could make a SCSI tape respond correctly.

5.1 Drives generally available

- Applix sell a full size (5.25 inch form factor) Seagate ST225N 20 megabyte SCSI drive, and a 40 megabyte Seagate ST251N SCSI drive. These have the advantage that the SCSI electronics are built into the actual drive. These are among the less expensive SCSI drives available, and are known to work well with the Applix 1616. May require an additional power supply.
- More compact 3.5 inch form factor 20 megabyte Miniscribe 3425S SCSI drives are also available from Applix. Other capacities may also be available. These can usually run from your existing Applix power supply, and are an elegant solution to the drive problem, as they can be mounted inside your Applix case. The disadvantages are the higher price, and limited capacity.
- Voice coil drives in both 5.25 and 3.5 inch form factors can be obtained. These are high quality, very fast, and correspondingly very expensive. They can be obtained in sizes ranging from 40 megabyte to 700 megabyte. The Applix 1616 SCSI interface has been tested, and works, with several large drives, including an Adaptec 4000A SCSI adaptor driving a 190 megabyte Maxtor, and a 380 megabyte CDC SCSI drive. However we do not normally use such large, expensive drives, and so are unable to offer much advice on the merits of large, fast drives (except that they seem very fast!)
- You can also use a SCSI adaptor board to convert an IBM style ST506 hard drive into a SCSI drive. This option is attractive to constructors who already own IBM style ST506 hard disks, or who have access to cheap supplies of such drives. Various people have used the Adaptec 4000A, 4070 and 5500 models. Use of a 4070 RLL controller is only advisable if your ST506 style drive is rated for RLL operation. Use of non-RLL drives with the 4070 controller is usually contrary to manufacturer's advice (but does increase your drive capacity by 50%).

  Xebec and Omti 3100 adaptors have also been used. The Omti 3110 will probably act pretty much like an Adaptec 4000A. If you use a SCSI adaptor, you will almost certainly have to provide an additional power supply.

5.2 MFM and RLL encoding

Run length limited (RLL) drives are actually identical to standard drives. The difference lies in the manner in which the SCSI (or other) controller encodes the data stored upon them. It is probable that drives rated for RLL by a manufacturer are simply standard drives, that have been tested to the tighter timing limits of the RLL encoding.

The original data storage encoding for floppy disks was frequency modulation (FM). Each 0 bit in the data was recorded as a space, each 1 as a pulse. Between **every** bit position was a pulse, known as the clock pulse. Because you are guaranteed a clock pulse between every bit, it is very easy to synchronise with and recover the actual data. You can see that twice as many pulses occur during a string of 1s as during a string of 0s (hence the name frequency modulation), while the average is 1.5 pulses per bit. However you must leave room on your disk for two pulses per bit, and thus the data density is a half less than what could theoretically fit on the disk.

Modified frequency modulation (MFM), the process used on standard Applix double density floppy disks, and also on most hard disks, is a method of using fewer pulses to encode the same data. It simply uses a different set of rules to encode the data and clock pulses.

A data bit of 1 is always encoded to a space followed by a pulse. That is, it never occupies more than one pulse.

A data bit of 0 can be encoded in one of two ways. **If** there was **no** pulse at the end of the previous bit, it is encoded by a pulse followed by a space. If there was a pulse at the end of the previous data bit (that is, if it was a 1), then the 0 bit is encoded to two silences.

This MFM scheme ensures at least one silence between pulses (so they can be tightly packed without actually running together), but never more than three silences in a row (so you can still relatively easily recover a clock). The data density is therefore 0.75 of a pulse per bit (on average), or twice that of FM (hence the common name ‘double density’).

Run lengths are the number of spaces between pulses. We saw that FM never has more than 1, while MFM can have up to 3. **If** your controller can reliably keep track of when pulses **could have** occurred, then you can allow more spaces between pulses. The encoding we generally hear called RLL is actually a special case of run length encoding, and is really 2,7 RLL (other common versions are that used by Apple, and the more recently used advanced RLL). In 2,7 RLL, you never have less than two spaces together, nor do you ever have more than seven spaces together.

In operation, groups of between 2 and 4 data bits are encoded as in the table below.

data bits	encoding
0 0	1 0 0 0
0 1	0 1 0 0
1 0 0	0 0 1 0 0 0
1 0 1	1 0 0 1 0 0
1 1 0 0	0 0 0 0 1 0 0 0
1 1 0 1	0 0 1 0 0 1 0 0
1 1 1	0 0 0 1 0 0

Obviously, the maximum run length (7) determines how accurate the controller and drive timing must be, while the minimum determines how tightly the data can be packed on the disk. The result is about 50% denser than standard MFM.

The major disadvantage of RLL encoding is that the timing is more precise. As a result, many older model drives are not capable of being used with RLL controllers. Some manufacturers, such as Seagate and Miniscribe, specifically warn against using their MFM style hard drives with RLL controllers. The same makers sell (at slightly higher cost) RLL rated versions of the same drives. The difference is probably only in the precision (or testing) of the drive electronics. My own preference is to stick to MFM models, however many Applix owners have used RLL controllers without problems, and are happy with them.

5.3 Seagate

The ST225N drive normally requires +5 volts at 1.2 amps, and +12 volts at 0.9 amps, with a start up current of about 2.5 amps. The ST251N is similar. This sometimes exceeds the start up capabilities of an Apple style power supply, so a separate disk drive power supply may be required. IBM clone supplies seem the cheapest solution, at round $100. These power supplies are often inconveniently noisy, and will not fit in your Applix case.

Seagate SCSI drives include the following:

Model	Size	Heads	Cylinders	Access mS	Power	Form Factor
ST125N	21.5	4	407	8	9	3.5 RLL
ST138N	32.2	4	615	8	9	3.5 RLL
ST157N	48.6	6	615	8	9	3.5 RLL
ST177N	60.8	5	926	8	9	3.5 RLL Servo
ST1096N	83.9	7	910	8	9	3.5 RLL Servo
ST225N	21.3	4	615	20	17	5.25 MFM
ST251N	43.2	4	630	8	13	5.25 RLL
ST277N	64.9	6	630	8	13	5.25 RLL
ST296N	84.9	6	820	8	13	5.25 RLL
ST4192N	168.5	8	1147	5	26	5.25 RLL Full height

Seagate drives starting with a 1 are 3.5 inch form factor, and will probably fit within the Applix case and work off its power supply. Drives starting with a 2 are 5.25 inch form factor, and may require an additional power supply. The ST4192N is a fast, high capacity, full height drive.

5.4 Miniscribe 3425S

These 3.5 inch form factor SCSI drives were available from Applix in the past. They were an attractive choice, since they will run off the Applix 1616 internal power supply, drawing 0.75 amps from the 12 volt line (two exceed the capacity of the Apple supply however). They can be easily mounted inside the Applix 1616 case. Applix now tend to supply Quantum and other drives instead.

One nice feature of the 3425S is that the drive motor will turn off when you park the heads. I find this feature attractive, as it leads to quiet contemplation of your work, without sacrificing the use of a hard drive when required.

Miniscribe SCSI drives include the following:

Model	Size	Heads	Cylinders	Access mS	Power	Form Factor
8425S	21.3	4	612	68	12.5	3.5
8051S	42.0	4	745	28	8	3.5
9380S	347	15	1224	16	18	5.25 FH

6
SCSI adaptors

This conversion option can be attractive due to the relatively low cost of the slower IBM style ST506 drives, which can sometimes be obtained second hand. Applix generally don't use this option, so don't expect them to be able to give too much specific advice if you get stuck. In fact, unless you know what you are doing, don't try this option unless you have help available from someone who has done it (given that, I'll add that I didn't have many problems when testing a variety of drives while writing the manual).

Surplus Adaptec controllers are often available from US sources such as Computer Surplus Store, or Timeline, at round US$100 for 4070 model, and round US$125 for 4000A and 5500 models. Freight and sales tax (24%) will increase these prices considerably. No cheap Australian source is known, however various Applix owners sometimes import a bulk order. These are basically an 8085 CPU, DMA, ST506 and SCSI support and interface chips, and a 1 kbyte sector buffer. They also include diagnostic routines, and a superset of the SCSI command set.

Xebec adaptors have also been tested, and are often cheaper than the Adaptec. The version tested had the disadvantage of requiring the drive parameters sent to it whenever it was reset. This means that you could not boot from the hard disk. Omti adaptors have also been used.

You will also require an external power supply, as adaptors draw 1.5 amps from the +5 volt line. An IBM clone supply round $150 is a good choice, but is noisy, and will not fit in your Applix 1616 case. You will also require additional 20 way and 34 way cabling between the adaptor and the ST506 drive. This may make the total cost comparable to a SCSI drive. Manuals for each model are available from *Computer Surplus Store* for US$8 each (but postage will be at least that, as they are round 300 pages!)

The great advantage is that a 4000 model can support *two* ST506 drives (as can an IBM power supply). A 5500 can support up to *four* ST506 drives, and has a 2k buffer, twice that in the 4000 models. If you have expansion in mind, the Adaptec can be a very worthwhile option, as your second drive cost is much lower than adding a second SCSI drive.

The Adaptec controllers are the same form factor as 5.25 inch drives, and have mounting holes that should allow them to be secured on standoffs directly above the drive electronics. You will need to obtain the standoffs, and my drives never seem to have the holes in the correct spot.

6.1 Description

The 4000A (which is compatible with the earlier 4000 and 4010 combination) and the 4070 are identical in operation, except that the 4070 uses 2/7 Run Length Limited (RLL) encloding of data, rather than the conventional MFM method. This means that instead of 17 sectors of 512 bytes per track, you can get 26 sectors of 512 bytes per track, about a 50% increase in capacity. The downside of this is that your ST506 drive **must** be RLL rated. Most recent drives are capable of running with RLL encoding, however, some manufacturers (such as Seagate and Kolak, which have the same designer) still explicitly specify that you must not run their standard drives as RLL, and some spokesmen have stated that drives can be damaged (their RLL certified drives are usually round $10 extra). Older drives are generally not capable of running reliably using RLL. I'm cautious, and generally don't run RLL (although I've had no problems), but lots of people do use it without problems. Drives that are known to not run with RLL include original Seagate ST506 5 megabyte, and the 3.5 inch Tandon TM362.

Adaptec 4000 and 4070 adaptors have two socketed termination resistor packs (RP3 and RP4), which provide 220Ω to +5 volts, and 330Ω to ground. Do not remove them! The last drive connected must also have termination resistor packs enabled or in place.

There are a number of jumper blocks on the Adaptec 4000A and 4070 adaptors. If you have only one SCSI device (the Adaptec) connected to the 1616, do not install any jumpers. For the eternally curious, the purpose of the jumpers is described below. The first three are used to set the SCSI address of your drive. If you have only one drive, we expect it will be drive 0 (thus, no jumpers).

Jumper	Description	Jumpered	No jumper
A-B	LSB of SCSI address	1	0
C-D	part of SCSI address	1	0
E-F	MSB of SCSI address	1	0
G-H	DMA transfer rate	Sysclock/4	Dataclock/2
I-J	Extended command set	Enabled	Disabled
K-L	Not used		
M-N	Seek complete signal support	Enabled	Disabled
O-P	Self Diagnostics	Enabled	Disabled

There is an additional set of jumpers on the 4000A adaptor, controlling write precompensations (which is never used in RLL drives).

R-PU	Write Precompensation off	On	Off
R-S	Same as reduced write current	On	Off
R-T	All tracks, all drives	On	Off

The 4000 and 4070 adaptors have built in diagnostic routines that can be used for system troubleshooting, or simply to reassure you the board is working. Remove all cables, and connect a jumper across 0-P. Power up the board. If it is functioning correctly, the LED will flash continuously, with a period between 0.5 and 1 second (some 4000A boards flash the LED faster than the manual claims).

If there is a problem, the LED will stay lit for 6 seconds, flash once for one second (indicating the start of the diagnostics), and then flash in 0.5 second bursts. This will be repeated as long as the O-P jumper is in place. The number of 0.5 second bursts is the error code.

No burst	8085 subsystem
1	8156 RAM
2	Firmware
3	AIC-010 or related
4	AIC-010 or related
5	AIC-300 or related
6	AIC-010 bus

6.2 Drive Parameters

When first formatting an ST506 style drive, you are required to provide certain parameters to the `adaptec` program on your hard drive users disk. The information required for formatting should be available in your drive manual, however many drives are sold without manuals (indeed, many vendors don't appear to understand why someone may need a manual).

Below are some typical drive parameters that should work (corrections and additions are sought). All are 17 sectors of 512 bytes, except those marked RLL, which are 25 sectors. All have step rate acceptances of less than 12 microseconds (most buffer the pulses). All will probably accept a head landing zone 4 cylinders past the last data track. Failing that, use the last cylinder as a landing zone.

6.3 Table of drives

6.3.1

Maker	Model	Size	Heads	Cyl	Wr Pre	Reduce Wr
ALPS	DRNO10A	10.00	2	615		
	DRNO20A	20.00	4	615		
	DRMO10A		10.00	2	615	
	DRMO20A		20.00	4	615	
Atasi	3020	16.80	3	645		
	3033	28.10	5	645		
	3046	39.30	7	645		
	3051	42.90	7	704		
	3075	67.10	8	1024		
	3085	67.1	8	1024		
BASF	6185	23.00	6	440		
	6186	15.30	4	440		
	6187	7.70	2	440		
BULL	D530	25.00	3	987		RLL rated
	D550	42.90	5	987		RLL rated
	D570	59.00	7	987		RLL rated
	D585	67.10	7	1166		RLL rated
CDC	9415-36	28.93	5	697	256	210
	9515-3	18.20	3	697		
	94155-67		7	960		
	94155-85	71.00	8	1024		FH
	94155-86		9	925		
	94155-96		9	1024		
	94155-129	102.00	8	925	RLL	FH
	94205-52	42.00	5	989		
	94355-100	83.00	9			
	94355-150	128.00	9			
CMI	3212	10.16	2	612	128	
	3426	20.32	4	612	256	
	5206	5.30	2	306		
	5412	10.16	4	306	128	
	5616	13.40	6	256		
	5619	15.24	6	306	128	
	6213	11.10	2	640		
	6426	21.25	4	640	256	
	6640	31.87	6	640	256	
	7660	50.10	6	960		
	7880	66.80	8	960		
Cogito	CG906	5.30	2	306		
	CG912	10.70	4	306		
	PT912	10.70	2	512		
	PT925	21.30	4	612		
Connor	CP342	41.00	5	980		
Cynthia	520	57.35	7	987		
Disktron	526	21.3	8	306		

Fuji	303-52	42.00	8	615	
	305-39	32.00	4	615	RLL
	309-26	21.00	4	615	
	309-39	32.00	4	615	RLL
	309-58	49.00	6	615	RLL
Fujitsu	M220	5.60	2	306	
	M2223A	10.00	4	306	
	M2224A	15.00	6	306	
	M2230AT	5.60	2	320	
	M2233AS	10.62	4	320	128
	M2233AT	10.62	4	320	128
	M2234	15.70	6	320	
	M2235AS	21.25	8	320	128
	M2241AS	26.30	4	754	
	M2242AS	43.81	7	754	
	M2243AS	68.85	11	754	
Hitachi	520-1	10.16	4	306	128
	520-2	15.24	6	306	128
	520-3	10.16	4	306	128
IMI	2306H	5.30	2	306	
	2312H	10.70	4	306	
	5006H	5.30	2	306	
	5012H	10.70	4	306	
	5018H	16.00	6	306	
Kalok	KL320	20.42	4	615	300
	KL330	30.60	4	615	RLL
Kyocera	LT200	21.00	4	615	
	LT300	32.00	4	615	RLL
	LT2000	21.00	4	615	
	LT3000	32.00	4	615	RLL
Lapine	3065	10.16	4	306	0
	3512	10.16	4	306	0
	3522	10.16	4	306	128
	TL200	20.42	4	615	
	Titan	20.00			RLL rated
Maxtor	1065	55.90	7	918	
	XT1085	71.00	8	1024	
	XT1105	82.70	11	918	
	XT1120	101.00	8	1024	RLL
	XT1140	120.00	15	918	
	XT1160		15	918	
	XT1240R		15	1024	
	XT2085	70.20	7	1224	
	XT2140	110.30	11	1224	
	XT2190	160	15	1224	
Micropolis	1302	20.67	3	830	
	1303	34.45	5	830	
	1304	41.34	6	830	
	1323	34.00	4	1024	
	1323A	42.50	5	1024	
	1324	51.00	6	1024	
	1324A	59.50	7	1024	
	1325	67.10	8	1024	
MicroSci	HH312	10.18	4	306	128
	HH315	10.70	4	306	
	HH325	20.32	4	612	128

	HH330	21.30	4	612		RLL rated
	HH612	10.16	4	306	128	
	HH625	21.30	4	612		
	HH725	20.32	4	612	128	
	HH738	21.33				RLL rated
	HH1050	41.00	5	1024		RLL rated
MicroStore	MS212R	10.70	4	306		RLL rated
Miniscribe	2006	5.30	2	306		
	2012	10.16	4	306	128	
	3006	5.30	2	306		
	3012	10.16	2	612	128	
	3053	44.6	5	1024		5.25 Voice coil
	3085	71.3	7	1170		5.25 Voice coil
	3212	10.16	2	612	128	
	3412	10.16	4	306	128	
	3425	20.42	4	615	128	
	3650	42.25	6	809		
	3675	63.3	6	809		RLL?
	4010	8.40	2	480		
	4020	16.70	4	480		
	5330	52.10	6	480		
	5338	32.00	6	612		
	5440	33.40	8	480		
	5451	42.6	8	612		
	6032	25.50	3	1024	512	
	6053	42.50	5	1024	512	
	6074	59.50	7	1024	512	
	6085	71.3	8	1024		5.25 FH Voice coil
	6128	110.1	8	1024		5.25 FH RLL?
	8212	10.21	2	615	128	
	8425	20.42	4	615	128	
	8438	32.7	4	615		RLL?
	9380	338.20	15	1224		RLL rated
Mitsubishi	MR522	21.30	4	612		RLL rated
MMI	M112	10.16	4	306	128	
	M125	20.32	4	612	128	
	M225	20.32	4	612	128	
	M325	20.32	4	612	128	
NEC	3126	20.00				RLL rated
	5124					
	5126	20.00				RLL rated
	5224					
	5244					
Newberry	NDR1065	53.34	7	918		
	NDR1085		8	995		
	NDR1105		11	995		
	NDR1140		15	995		
	NDR2190		15	1244		
Okidata	OD526	21.30	4	612		RLL rated
	OD540	33.40	6	480		RLL rated
Olivetti	HD662/11	10.00	2	612		
	HDD662/12		20.00	4	612	
Otari	514	10.16	4	306	128	
Priam	502	46.00	7	755		
	504	46.00	7	755		
	514	117.20	11	1224		

	519	159.80	15	1224		
PTI	PT338	30.00	6	615		
Quantum	Q520	17.00	4	512	256	
	Q530	25.50	6	512	256	
	Q540	34.00	8	512	256	
Rodime	201	5.30	2	306		
	201E	10.62	2	640	0	210
	202	10.70	4	306		
	202E	21.25	4	640	0	210
	203	16.00	6	306		
	203E	31.87	6	640	0	210
	204	21.30	8	306		
	204E	42.50	8	640	0	210
	252	10.16	4	306	0	
	342	10.70	4	306		
	351	5.30	2	306		
	352	10.16	4	306	0	
	RO3055	45.60	6	872		
	RO3065	53.10	7	872		
	RO5090	74.60	7	1224		
Seagate	ST125	21.40	4	615		
	ST138	32.10	6	615		
	ST138R	32.70	4	615	RLL	
	ST151	42.50	5	977		
	ST157R	49.10	6	615	RLL	
	ST212	10.16	4	306	128	
	ST213	10.21	2	615	300	
	ST225	20.42	4	615	300	
	ST238R	30.03	4	615	RLL	
	ST250R	42.3	4	667	RLL	
	ST251	40.84	6	820	256	
	ST277	60.06	6	820	RLL	
	ST4026	20.42	4	615	300	
	ST4038	30.42	5	733	300	
	ST4051	40.55	5	977	300	
	ST4053	44.50	5	1024		
	ST4077R	68.20	5	1024	RLL	
	ST4096	80.20	9	1024		
	ST4144R	122.7	9	1024	RLL	
	ST406	5.08	2	306	128	
	ST412	10.16	4	306	128	
	ST417	15.24	6	306	128	
	ST419	16.00	6	306		
	ST425	21.30	8	306		
	ST506	5.08	4	153	128	
Shugart	SA604	5.60	4	160		
	SA606	8.40	6	160		
	SA607	5.40	2	311		
	SA612	10.80	4	311		
	SA706	5.30	2	306		
	SA712	10.16	4	306	128	128
Syquest	SQ306R	5.3	2	306		
	SQ312RD	10.16	2	612		
	SQ319R	RLL version of 312				
	SQ325	10.36	4	312	128	
	SQ325AF	20.32	4	612		

	SQ338	30.00	6	612		
Tandon	252	10.16	4	306	128	
	262	20.32	4	612		
	362	20.32	4	612		
	501	5.30	2	306		
	502	10.16	4	306	128	
	503	10.16	4	306	128	
	TO703	30.30	5	695		
	TM755	43.00	5	980		RLL rated
Toshiba	MK56FA	67.00				RLL rated
Tulin	TL213	31.87	6	640	256	
	TL220	21.25	4	640	256	
	TL226	21.25	4	640		
	TL238	32.00	4	640		RLL rated
	TL240	31.87	6	640		
	TL258	48.00	6	640		RLL rated
	TL262	20.32	4	612		
	TL338	32.00	4	640		RLL rated
	TL358	48.00	6	640		RLL rated
	TL362	21.25	4	640		
Vertex	V130	25.80	3	987		
	V150	43.00	5	987		
	V170	57.35	7	987		
	V185	67.1	7	1166		RLL rated

7
Z80 SSDCC ROM Commands

This chapter is very heavily based on Mark Harvey's original notes, which detail the commands he has implemented in his SCSI code in the Version 2.0, 2.1, 2.2 and 2.3 Z80 EPROMS. These commands correspond to, and expand upon, the Interprocessor Communication commands detailed in the Disk Co-processor manual.

This is a superset of the original version 1.4 EPROM, with SCSI control added.

All the original documented commands work the same as in version 1.4. There are also a few others that were mainly used in development of the SCSI software. Some may be useful either at a latter stage, or for those who are adding a SCSI device that has not been tested before.

This is valid of roms dated 20-NOV-88 and later (found by using the `sdc_id.xrel` program in the `/bin` subdirectory on the Hard Drive Users Disk).

Details of formatting and preparing a new hard disk are given in the section on utilities, earlier in this manual.

The commands are as follows:

7.1 Block read 01

unit, blockhigh, blocklow,
<errorcode or 0> <1024 bytes>

All commands are normally described as single bytes. The command byte (in this case hex 01) is sent to the command port at $FFFFD1. Then the rest of the command parameters are sent to the data port at $FFFFC1. Valid unit numbers are 0 to 16 (0 for drive 0, 1 for drive 1, etc). The block value required is sent high byte first, then low byte.

The Z80 performs the physical read, and returns an error code to the 1616. If the error code is 0, the 1616 may read out the 1024 bytes of data. If the error code is non-zero, an interpretation of it may be obtained with command 3 (below).

7.2 Block write 02

unit, blockhigh, blocklow, data
<errorcode or 0>

Similar to the block read, except that the 1616 sends 1024 data bytes to the Z80, and then must wait for the physical write to complete before an error code is returned. This may not be the best choice in terms of speed, but it does mean the system is less fragile.

7.3 Display error code 03

errorcode
<string> <0>

The 1616 sends the command code, and the error code byte (the error code byte is that returned by the Z80 following, say, a block read or block write). The Z80 returns an ascii string, of varying length, for human interpretation, followed by a zero (null) byte.

7.4 Format (Floppy) 04

unit, $B5, $7E, ntracks, skewtable
<errorcode or 0>

description	to Z80	from Z80	No. of bytes	
Command	04 hex		1	
Unit No.	xx		1	
Magic No.	xx		2	B57E hex
No. of tracks	xx		1	
skew table	xx		10	side 0 is first 5 bytes
EC		xx	1	

To physically format a floppy disk, the 1616 sends $04 to the command port. Next the unit number is sent to the data latch. Unit number is, as always, between 0 and 16, with 0 being drive, 1 being drive 1, etc. The $B57E is simply a magic number, used as a check against the accidental issuing of format commands. The number of tracks (usually 80, or possibly 40) is next. Last is the 10 byte skew table. Wait for an error code to be returned.

The sector skew table consists of two consecutive 5 byte tables. The first is for side 0 of the disk, the second 5 bytes for side 1. All sectors receive the same format pattern with this command. The Z80 driver code expects sectors to be numbered from 1, not from 0 (don't blame us, the FDC chip makers all do it that way). The skew table that is sent by the `blockdev.xrel` utility program is:
Side 0 1, 4, 2, 5, 3
Side 1 3, 1, 4, 2, 5

7.5 Format type 2, command 05

This format command is not documented by Applix, and can be considered abandoned by them for the moment. It was intended to permit interactive track by track formatting, and sector skewing. It is used by Greyham Stoney's Shareware software for special formats, such as IBM and CP/M formats.

7.6 Flush buffers 06

Flushes the disk buffers, if any. If there are no buffers, takes no action. If there is no action for a few seconds, check for a changed disk.

7.7 Read Z80 ram 07

Z80addrh, Z80addrl, lengthh, lengthl
<data>

Specify the Z80 address required, in high byte, low byte format, followed by the number of bytes to read, in high byte, low byte format. Reads 'LENGTH' bytes from the Z80 memory, starting at the Z80 address specified.

7.8 Write Z80 ram 08

Z80addrh, Z80addrl, lengthh, lengthl, data

The 1616 writes 'LENGTH' bytes into the Z80 memory, starting from the specified Z80 address.

7.9 Call a Z80 program 09

Z80addrh, Z80addrl

Causes the Z80 to perform a 'CALL' instrustion to the address supplied. Upon return from the called program, the Z80 resumes normal operation. $7800 and up is as good a place to experiment as any.

7.10 Read Z80 ROM version $0A (dec 10)

<ROM version>

Upon receiving this command, the Z80 returns a version byte for its current EPROM.

7.11 O/S version byte $0B (dec 11)

Version byte

The 1616O/S tells the disk controller which version it is.

7.12 Set floppy step rate $0C (dec 12)

unit, rate

This command is used to alter the floppy disk(s) head step rate. It is intended for use with slower drives. The values that work are:

Value	Step rate used
0	2 mS
1	3mS
2	6mS (default rate)
3	12mS

You may notice that these rates do not correspond with the step rates quoted in some WD1772 specification sheets. Trust us, the specification sheets are wrong.

7.13 Set interrupt $0D (dec 13)

flagh, flagl

Included for use with Minix. Not used under 1616 OS V3.1. The idea of this interrupt is that the Z80 will interrupt the 1616 after it has read/written a block of data. The interrupt is sent just before the error code is returned.

MSB of the first byte (bit 7) will set an interrupt for unit 0 (/f0) and the LSB (bit 0) of the second byte will set it for unit 15.

Note :- at the moment, only a zero or non zero is tested for. i.e. if the interrupt flag is non-zero, then the Z80 will interrupt the 1616. To be inproved upon in latter versions, if needed or requested.

Greyham Stoney's disk software provides this command also, generating the interrupt on EIRQ1.

7.14 Read interrupt value $0E (dec 14)

<byte>, <byte>

Returns the value of the interrupt flag, for units 0 to 15. The unit encoding scheme is the same as Set Interrupt above. That is, MSB (bit 7) of the first byte is unit 0, LSB (bit 0) of second byte is unit 15.

7.15 not used $0F (dec 15)

7.16 Read sector $10 (dec 16)

unit, blocksize, track_no, sector, side
<errorcode or 0>, <data>

Reads data from a floppy in unit 0 or 1, in any standard sized sector length. Block sizes are as follows, 0 = 128, 1 = 256, 2 = 512, 3 = 1024. The amount of data read (if the errorcode is 0) is equal to the block size. The internal ROM uses these routines for normal use.

Note. Sectors start from 1 while tracks start from 0. (Don't blame me, it is the standard IBM format!)

7.17 Write sector $11 (dec 17)

unit, blocksize, track_no, sector, side, data
<errorcode or 0>

Writes data to a floppy in Unit 0 or 1 in any standard sized sector length. Block sizes are as follows, 0 = 128, 1 = 256, 2 = 512, 3 = 1024. The amount of data written must equal the specified block size. The internal ROM uses these routines for normal use.

Note Sectors start from 1, while tracks start from 0. (Don't blame me, it is the standard IBM format!)

7.18 Re-zero scsi device $21 (dec 33)

(no parameters required)

Sending this command will cause the SCSI device to rezero. Usually the drive will slowly step to end of disk, and quickly return home (not always the case). This is a good command to test to see if the scsi bus is talking OK. It dosn't have much use, but you can see/hear something happen if everything is OK.

7.19 Reset scsi bus $22 (dec 34)

(no parameters required)

Causes the Reset line on the SCSI bus to become active. This will/should reset all devices on the bus. Again not much use except for testing purposes.

7.20 Start/stop scsi $23 (dec 35)

start/stop

The stop command will park the heads, and in some cases stop the motor (Miniscribe 8425S).

The start/stop byte is 0 to park/stop the drive, and 1 to start it up.

If you have a SCSI - ST506 interface, such as the Adaptec 4000A or 5500, more than one drive can be conected to them. In this case they are referred to as Logical Units. i.e. drive 0 is LUN 0 (Logical Unit Number), drive 1 is LUN 1, up to LUN 7. This is as many as the SCSI standard will handle. In this case the LUN is encoded in the 3 MSBs of the start/stop byte.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	start/stop bit
x	x	x	0	0	0	0	0

i.e. To park the heads of LUN 2 then the byte would be 40 hex. To park the heads of LUN 3 then the byte would be 60 hex. To start up LUN 3 the byte would be 61 hex.

Mark Harvey's utility programs `shutdown` and `startup` provide a command line version of this command.

7.21 Format scsi $24 (dec 36)

$A4, $5B, interleave, formatlist
<errorcode>

description	to Z80	from Z80	No. of bytes	
Command	24 hex		1	
Magic Number	xx		2	(A45B hex)
Interleave	xx		1	
Format list	xx		1	
Error Code		xx	1	

Format the **whole** scsi device, not just a partition. The *magic number* $A45B must be used, to as precaution against accidental formats.

The interleave can be anything between 1 and Max No. of sectors on a track - 1. For fairly standard MFM drives, this is usually 16 or 17. For RLL drives, it is usually 24.

The drive is formatted with a sector size of 512 bytes. Internal routines read in 2 sectors to make up 1 K blocks. This was done because not all SCSI drives can or will format using a 1k sector size.

The **formatlist** is a "magic number" that seems to vary from drive to drive. You **must** find documentation on your drive or controller to use this correctly.

The Miniscribe 8425S requires that the number be 18 hex (24 decimal) to format the drive with the primary defect list intact, and wipe out any grown lists. i.e. return drive to an 'as shipped' condition.

The Omti controller takes this byte to be the MSB of the track from which to start formatting.

All I can suggest is try 18 hex first. If that dosn't work then use 0.

Note Formatting is made a bit easer by using `HDCONFIG.XREL`, on the hard drive users disk. The numbers are entered in DECIMAL so the 18 hex becomes 24 decimal.

SCSI - ST506 (Adaptec) interface users note. If you want to format a drive other than LUN 0, then the Format List byte is encoded with the LUN as described in the start/stop command. That is, the most significant three bits of the byte are used to encode the Logical Unit Number. LUN2 means that $40 is added to the Format List byte. LUN3 means $60 is added to the Format List byte.

Details of Format List numbers for other brands of drives are most urgently sought, as are appropriate drive manuals. We have Adaptec 4000A and 5500 adaptor manuals, and Miniscribe 3425S manual.

7.22 Sense condition $25 (dec 37)

<data>

This command requests the condition of the drive, if an error has occured. This command should follow an error in reading or writing to the drive. The data that it returns will be four bytes as follows.

byte 1 - Error No.
byte 2 - MSB of block it happened on.
byte 3 - middle byte of block No.
byte 4 - LSB of block No.

The LUN is encoded in the top 3 bits of byte 2.

The error number also seems to vary from drive to drive. Refer to the manual on your drive/controller for details of drives we do not cover.

7.23 Send scsi command $26 (dec 38)

command block, data_to_pointer, data_from_pointer
<errorcode>, <status>, <message>

description	to Z80	from Z80	No. of bytes	
Command	26 hex		1	
SCSI command block		xx		10
data pointer				
data to scsi	xx		2	Z80 address
data pointer				
data from scsi	xx		2	Z80 address
EC		xx	1	
status		xx	1	
message		xx	1	

This has been included so as the 1616 can send a command block to the SCSI device. Obviously, you need a SCSI manual to make use of it.

To use this command you need to know a couple of things about the command.
i.e. What sort of data and quantity the command wants, etc.
Where in Z80 memory is it to be put (the area above $7800 is often safe).

First, all the data that the SCSI will want is uploaded into the Z80 ram by using a 'WRITE RAM 08' command. Then this 'SEND SCSI' command is sent, with the Z80 address of where the data is to be found (data pointer - data to scsi), and the address of where the data from the command is to be put (data pointer - data from scsi). Use the 'READ RAM 07' command to retrieve the data.

Returned is an Error Code, status byte and a message byte. The status and message bytes are returned from the scsi device. The error code is usually a copy of the status byte.

7.24 Read status and message bytes $27 (dec 39)

<status>,<message>

The status and message bytes returned from the last SCSI command are returned.

7.25 Set SCSI ID $28 (dec 40)

ID

The SCSI ID byte is encoded into the appropriate bit of the data byte sent. Bit 0 is ID 0, bit 1 is ID 1, bit 2 is ID 2, and so on, to bit 7.

When selecting a SCSI device, the destination address and source address bits have to be placed on the SCSI bus. For normal operation, the destination address is read from the table created by `hdconfig.xrel`, however there are some cases where it needs to be set up manually (formatting a drive, for example).

Note that it will have to be changed after every reset, if the default value is not used.

7.26 Set Z80's SCSI ID $29 (dec 41)

ID

Sets up the ID for the Z80 SCSI. Defaults to $40, or ID 6.

The data byte sent is encoded in the same manner as the destination SCSI ID (Set SCSI ID $28). You should not normally need to alter this value from the default, however if the SCSI bus has more than one master, a conflict may occur. To solve this, the original SCSI ID (that is, Set Z80's SCSI ID) can be changed.

Note that this will need to be set again after each reset, if the default is not used.

7.27 Disk change detect $40 (dec 64)

unit

The 1616 sends this command byte ($40 or 64) when it detects a change of the disk in the unit. This is sent only if the disk controller EPROM > Version 1.4.

Greyham Stoney's shareware software accepts this command, but ignores it, since the SSDCC can detect a disk change before the 1616/OS.

7.28 Level 0 reset $41 (dec 65)

7.29 Level 1 reset $42 (dec 66)

7.30 Level 2 reset $43 (dec 67)

7.31 Error messages

Here is a list of error messages from the SSDCC Version 2.2 EPROM, as interpretted by using the hard disk utility `readerror.xrel`. Usage is simply `readerror errorcode`, where *errorcode* is a decimal number.

1	FDC still busy
2	Timed out reading - drive is busy
3	Record not found
4	Lost data
5	Need to force and interrupt to FDC - busy at moment

6	Write protected
7	Format error
8	CRC error in data
9	Assemble date
10	SCSI bus in a free state
11	Read error xx on SCSI, ID x, LUN x, Block No xxxxxx
12	Write error xx on SCSI, ID x, LUN x, Block No xxxxxx
13	Error in sending SCSI command block
14	Error in bus phase
15	SCSI ID x not responding
16	Unit number requested is too big
17	Bus timeout
18	Can not read partition table on SCSI device
19	Unexpected loss of /BUSY on SCSI device
20	Error returned from SCSI controller
21	Invalid error number
22	No /RDY signal from /Fx
23	Read/Write error on SCSI, ID x, LUN x, Block xxxxxx
24	Function not operative at the moment
25	Bad magic number for format
26	Invalid error number
27	Invalid error number
28	Invalid error number
29	Invalid error number
30	Invalid error number
31	Invalid error number
32	128 byte sector disk - No read performed
33	256 byte sector disk - No read performed
34	512 byte sector disk - No read performed
35	1024 byte sector disk - No read performed
36	Invalid error number

7.32 Some background information

Memory Usage

The Z80's stack pointer is set to $7800 on startup, so any memory above this point will be safe to use. The local storage variables start from $6000 hex and extends up. Aprox 1.5 K is used all together.

Memory map

ROM	0000 to 5FFF	
RAM bank 0	6000 to 7FFF	
RAM bank 1	8000 to FFFF	(if 32k chip installed)

Interrupts

The Z80 is set to an interrupt level of 1, and should be returned to this state, if the user program changes it. The NON-MASKABLE interrupt is used in Vers 2.1, 2.3, etc., and must be strapped on the SSDCC. The maskable interrupt has a jump vector set to $7800 hex. Any user installed interrupt driven programs, such as SCC drivers, can/should be located here. The address at $7800 is set to a warm start at reset time.

Jump table

There is a jump table at $100 hex. Most of these are NOT tested. (ANDREW, MORE DETAILS?) The first 2 have been tested, and are cold and warm boot.

Things NOT to change.

- The IX and IY registers are pointers used internally. Must be restored to orignal values for floppy and error checking to work.
- Interrupt level of the Z80.

Any additions, changes or things you would like added, just ask contact Mark Harvey via Applix, and he'll see what he can do.

8
Appendix

8.1 SCSI connector

(50 way)

Pin#	Name	Description
2	B0	Data bit 0.
4	B1	Data bit 1.
6	B2	Data bit 2.
8	B3	Data bit 3.
10	B4	Data bit 4.
12	B5	Data bit 5.
14	B6	Data bit 6.
16	B7	Data bit 7.
18	PARITY	Data bus parity bit.
20,22,24	COM	System common ground.
25, 26	TERMN PWR	Normally connected to +5v. (as standard linking on SSDCC, may need to be cut for some SCSI drives).
28,30	COM	System common ground.
32	/ATN	Attention signal.
34	COM	System common ground.
36	/BUSY	Busy signal.
38	/ACK	Acknowledge signal.
40	/RESET	SCSI bus reset signal.
42	/MSG	Message signal.
44	/SEL	Select signal.
46	CTRL/DATA’	Control/Data signal.
48	/REQ	Request signal.
50	IN/OUT’	Input/Output signal.

All odd number pins (except pin 25) connect to system common ground.

Index

Table of Contents