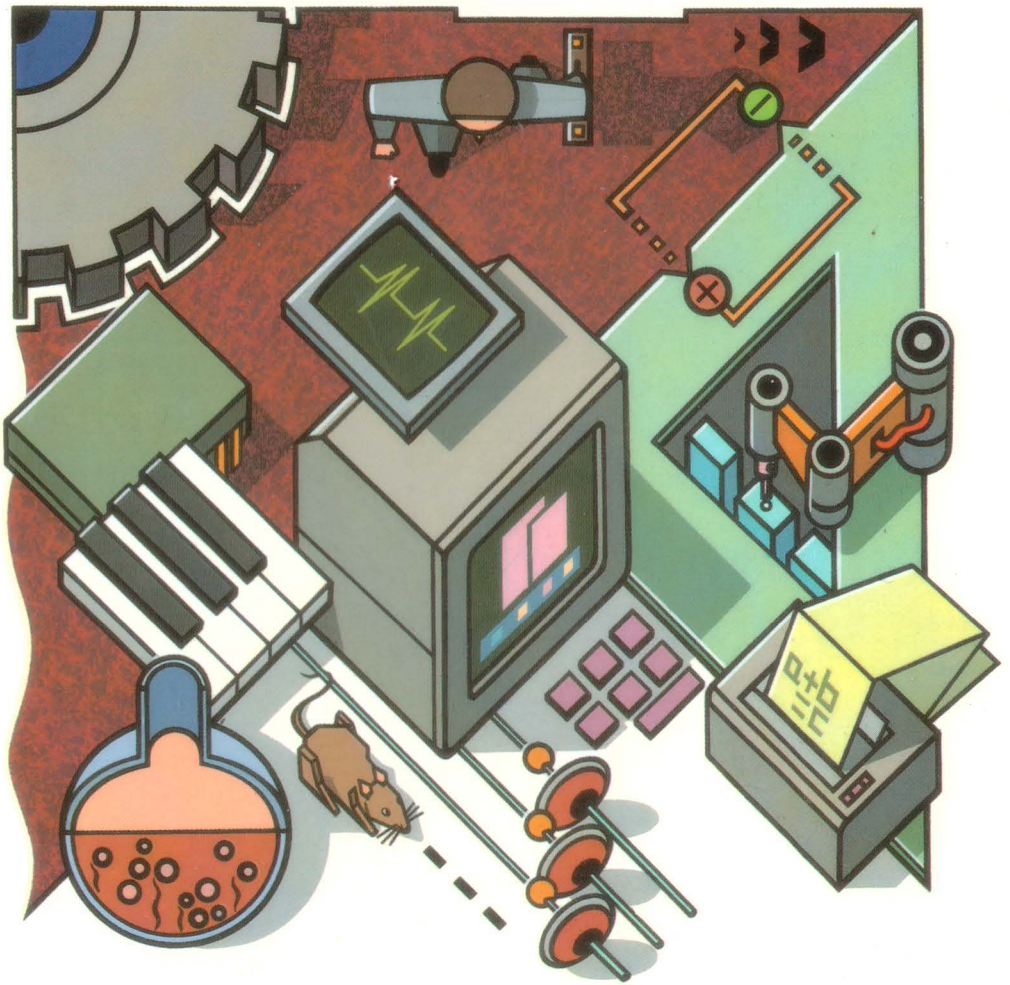
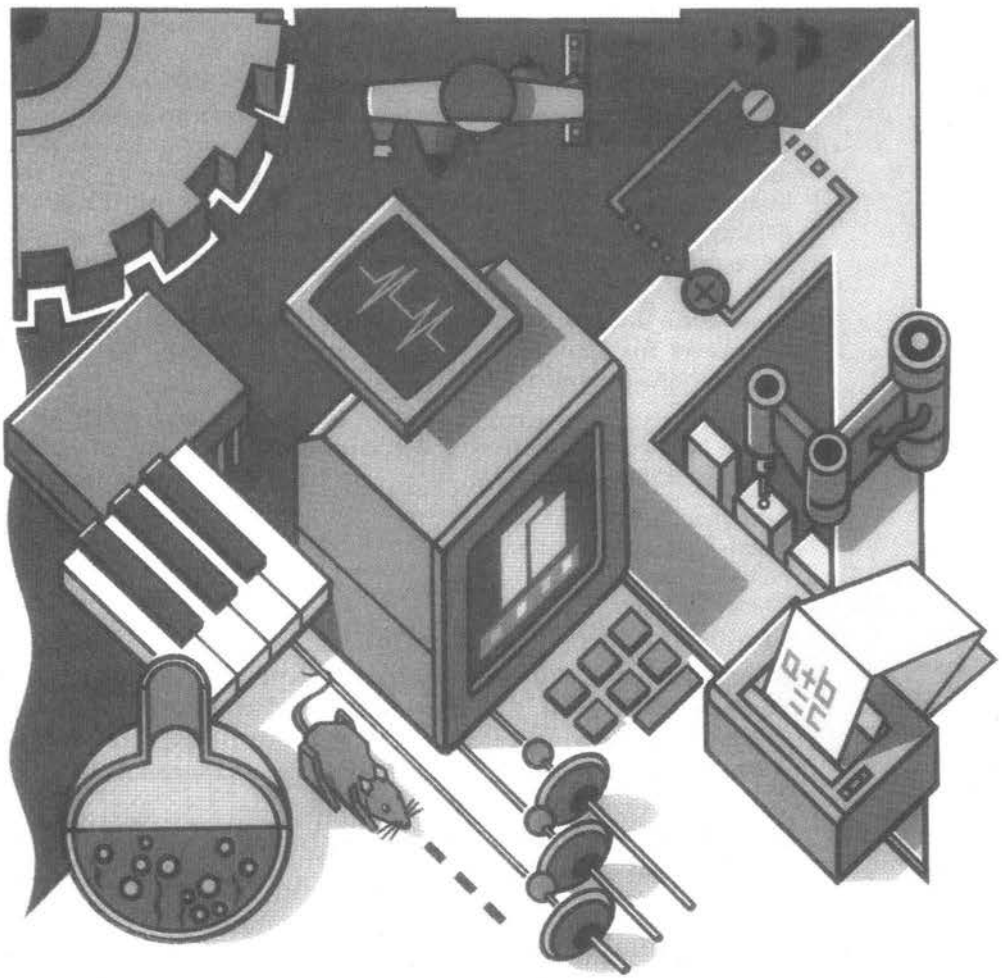
SCSI EXPANSION CARD
USER GUIDE



Acorn

SCSI EXPANSION CARD
USER GUIDE



Acorn

The product described in this Guide and the products for use with it, are subject to continuous development and improvement. All information of a technical nature and particulars of the product and its use (including the information and particulars in this Guide) are given by Acorn Computers Limited in good faith. However, Acorn Computers Limited cannot accept any liability for any loss or damage arising from the use of any information or particulars in this Guide, or any incorrect use of the product. All maintenance and service on the product must be carried out by Acorn Computers' authorised dealers. Acorn Computers Limited can accept no liability whatsoever for any loss or damage caused by service, maintenance or repair by unauthorised personnel.

If you have any comments on this Guide, please complete and return the form at the back of the Guide to the address given there. All other correspondence should be addressed to:

Customer Support and Services
Acorn Computers Limited
Fulbourn Road
Cherry Hinton
Cambridge CB1 4JN

Information can also be obtained from the Acorn Support Information Database (SID). This is a direct dial viewdata system available to registered SID users. Initially, access SID on Cambridge (0223) 243642: this will allow you to inspect the system and use a response frame for registration.

Second edition (covering SCSI Issue 2+)
Published June 1990
Published by Acorn Computers Technical Publications Department
Part number 0473,999
Issue 1

Contents

About this Guide

Introduction

This guide describes how to set up and use SCSI (Small Computer Systems Interface) peripherals with the Acorn SCSI expansion card, Issue 2+. The Issue 1 SCSI expansion card is covered by the *SCSI expansion card User Guide*, part number 0473,911, supplied with Issue 1 cards. Where possible this guide is specific, but the set up and connection of a particular SCSI device will depend upon the nature and type of SCSI peripheral concerned.

Using SCSI with RISC OS

All of this Guide is relevant to RISC OS, however to get your SCSI set up and working you should read the first three chapters detailed below.

To install the SCSI card and a SCSI device, read the first chapter, *Installation*. If you are installing a set-up involving more than one SCSI device then you should also read the second chapter, *Advanced installation*. To configure your devices for RISC OS operation you should read the third chapter *RISC OS operation*.

After following these three chapters your RISC OS SCSI set-up should be working correctly. You only need to read the rest of the guide if you want to find out more about how SCSI operates, or if you want to write a new SCSI device driver for RISC OS.

Using SCSI with RISC iX

If you are going to use SCSI with RISC iX (or both RISC OS and RISC iX), then the specific RISC OS chapters in this guide do not apply to you. To get your SCSI working you should read the first two chapters in this guide as detailed below.

To install the SCSI card and SCSI devices, read the first chapter, *Installation*. If you are installing a set-up involving more than one SCSI device, you should also read the second chapter, *Advanced installation*. Then, to configure your device for RISC iX, read the *RISC iX System Administrator's Guide*. This contains information about how to section a SCSI disc between RISC iX and RISC OS.

After following these instructions your RISC iX SCSI set-up should be working correctly. You only need to read the rest of this guide if you wish to find out more about how SCSI operates.

Chapter summaries

Installation: How to install your SCSI expansion card and how to connect up SCSI devices to it.

Advanced installation: Additional installation instructions that you should read if you wish to install more than one SCSI device on your system.

RISC OS operation: How to configure a SCSI disc for use with RISC OS.

SCSI explained: Background information about how SCSI works. You will need to read this chapter if you want to install a non-standard SCSI device.

Interfacing SCSI and RISC OS: Information about RISC OS that will enable programmers to create RISC OS SCSI drivers for additional SCSI device types.

Specifications: A list of hardware and software specifications for the SCSI expansion card.

The SCSIDM program: Describes the use of the *SCSIDM disc management program in full. This will be of use to users who wish to use the advanced features of the program.

The SCSI device checklist: Complete the device checklist with the parameters that you have assigned to the SCSI devices on your system.

Installation

Introduction

The Acorn SCSI (Small Computer Systems Interface) expansion card allows peripheral devices that conform to the ANSI SCSI specification to be connected to all computers in the Acorn Archimedes and RISCiX workstation ranges.

ANSI compatibility

The SCSI card conforms to the ANSI SCSI specification (X3.131 1986 SCSI standard and the Common Command Set (CCS) document Rev 4B for direct access devices) recommendations, and should operate with devices that also meet these recommendations. Some devices may not support certain CCS-4B optional commands and therefore may not operate properly with the Acorn SCSI card; your supplier will be able to recommend compatible devices. Guidelines to help you select suitable CCS-4B Direct Access SCSI devices are given at the end of this chapter.

If you want to connect a SCSI device that does not conform to the ANSI standard, you should find out all that you can about the device from the manufacturer and then read the chapter entitled *SCSI explained*, which describes in detail the principles of the SCSI interface.

Installation

If you wish to:

- install more than one SCSI device in a system
- connect a non-CCS-4B compatible SCSI device
- alter the termination of the Acorn SCSI expansion card

or if your SCSI set-up is not working correctly, read the chapter entitled *Advanced installation* after you have followed the instructions in this chapter.

Archimedes 305, 310 and 440 owners: You must return your computer to your supplier for a free internal modification before you can install the SCSI card. (Archimedes 400/1 series, A3000 and later computers do not need this modification.)

Follow the expansion card installation instructions

Install the card in your computer according to the *Expansion card installation instructions* supplied with the card.

Connecting devices

This section describes how to connect ANSI conforming devices to your SCSI expansion card.

You should have:

- the SCSI expansion card already installed in your computer
- the appropriate SCSI interface cables
- the SCSI device(s)
- a SCSI terminator (if your device cannot be terminated internally).

Setting the device id number

Every device is given a SCSI id number in the range zero to six. Normally the first (or only) SCSI device should have an id number of zero. When using RISC OS, only four devices can be supported, and their id numbers must be contiguous, starting from zero. The id number is usually set by moving a jumper or dial on the SCSI device. Consult the device's instruction manual for more information. All devices must have a unique SCSI id number. The id number seven is reserved for the Acorn SCSI card.

Terminating SCSI devices correctly

Proper termination of the SCSI bus is very important to ensure correct operation. Failure to do so may result in data corruption, damage to the device (or SCSI expansion card), or cause the software to hang. The basic rule is that devices on each end of the SCSI bus should be terminated, and there must be no other terminators present on the bus. If you are connecting a single SCSI device to the SCSI expansion card, the card should be terminated, using one of the terminators supplied, and so should the SCSI device, either with an external (free-standing) terminator, or else with an internal terminator inside the device. For more information, read the instruction manual for your device.

If you are going to attach more than one device to your SCSI bus, you must make sure that the bus is correctly terminated. Only the device at each end of the bus should be terminated; all other devices connected to the bus must have their terminators removed

Terminators require power, without which the bus will not function. An external (free-standing) terminator will take power from the SCSI bus. A

self-terminated device will normally provide power to the internal terminator, but not always to the bus. In this case the device will need to be powered on for any device on the SCSI bus to function.

Connecting SCSI devices

Many SCSI devices are delivered with their internal terminator set. If the device is not the last device on the SCSI bus, the terminator must be disabled according to the manufacturer's instructions. For more information, read the instruction manual, or consult the supplier of the device.

The SCSI expansion card supplies terminator power to the SCSI bus via a diode and fuse.

Powering SCSI devices

When powering on or powering off your computer you must follow the following procedure:

- when **powering on** always switch on all of your SCSI devices **before** switching on your computer.
- when **powering off** always switch off your SCSI devices **before** switching off your computer.

You can, however, power some SCSI devices on or off within the bus without any problem (as long as they remain connected to the bus). This is useful for some devices that are used only occasionally, such as tape drives.

Filling out the SCSI device checklist

To help you remember the parameters of the various devices attached to your SCSI bus, you should fill in the SCSI device checklist (at the end of this guide) with details about your SCSI devices. You will then be able to look up the relevant details about each of your devices at a glance.

SCSI device type and name: This column lists device types. Complete this column by putting in the names of your SCSI devices.

SCSI id number: This column suggests which SCSI id numbers should be given to which device types. Write in the id numbers you have assigned.

RISC iX partition: If you are using the RISC iX operating system you may partition your disc for use with both operating systems. Write in which devices have been partitioned for RISC iX.

Device terminated?: Only the first and last devices in the SCSI bus should be terminated. Usually the SCSI expansion card is terminated and set up at one end of the bus. Write down which other SCSI device is terminated (this must be the device at the other end of the SCSI bus).

Selecting a suitable CCS-4B Direct Access device

You should look for the following attributes when selecting a CCS-4B Direct Access SCSI device to work with the Acorn SCSI expansion card:

The device should comply with the ANSI X3.131 - 1986 SCSI Standard, and with the document *Command Command Set (CCS) X3T9.2/85-52 Revision 4B*, as detailed below:

The minimum data requirements are that the device:

1 supports all CCS-4B REQUIRED commands, except RESERVE UNIT and RELEASE UNIT.

2 supports the OPTIONAL commands:

VERIFY

MODE SENSE (with pages 3, 4 and 3F)

MODE SELECT (with CCS-4B page format, ie, the device recognises PF bit)

REASSIGN BLOCKS

READ DEFECT DATA.

3 supports all CCS-4B REQUIRED messages of the Message System, except RESERVATION CONFLICT.

4 requires Group Codes 0, 1 and 5 only (ie, 6, 10 or 12 byte CDBs), and no vendor-unique fields to implement commands.

5 requires asynchronous SCSI bus transfers – the Acorn SCSI expansion card does not support synchronous tranfers.

The sector size must be either 256/512 or 1024 bytes per sector.

Disconnection and reselection

The Acorn SCSI expansion card supports device disconnection and reselection (refer to *Disconnection and reselection* in the Chapter *SCSI Explained* for an explanation). Devices which are able to disconnect and reselect must support the CCS-4B OPTIONAL commands DISCONNECT and SAVE DATA POINTER.

Advanced installation

This chapter is aimed at those people who:

- need to install more than one SCSI device in a system
- have already connected a SCSI device, but cannot get it to operate correctly
- need to alter the termination of the Acorn SCSI expansion card.

You should only read this chapter after you have followed the instructions in the first chapter, *Installation*.

System configurations

The simplest possible system is just one host (the Acorn SCSI expansion card) and one target (the SCSI device). However, one of the advantages of SCSI is that it may share the bus between several different target devices. In total, there may be up to eight devices directly connected to the bus and one of these **must** be the host device. In the case of the Acorn SCSI card and its software only **one** host is supported.

Electrically, there are two very important points which must be considered at all times:

- the bus must be correctly terminated
- data cables of sufficiently high quality must be used.

SCSI cables

The Acorn SCSI card is designed to offer a very high performance system. Even when only asynchronous operation is used, the transfer of data on the bus may be at up to 2Mb/sec. This allows operation with drives whose media transfer rate is 15 Megabits/sec at 1:1 interleave.

Under these circumstances the behaviour of the cable becomes an important component to overall system reliability, especially as comparatively high currents may flow during transitions. It is important to realise that many cables advertised as 'SCSI' are unsatisfactory for high speed systems.

The following requirements should be noted:

1 50-way twisted pair or IDC cable must be used (many cables only have one core of earth or use the screen as an earth return).

2 The characteristic impedance of the cable should be 80-90 ohms minimum (preferably higher but this is difficult if the cable has electrostatic screening). The characteristic impedance is a phenomenon familiar to engineers – it refers to the electrical response of the cable to very high frequency digital signals. It is difficult to measure without the right tools, and should not be confused with the DC resistance, which will normally be a few tenths of an ohm and is unimportant in this application.

3 Good electrostatic shielding, including a full foil shield at the connector, is required to ensure that the system will meet such standards as VDE and FCC.

If a poor quality or inadequate cable is used, data transfers will be unreliable, leading to system crashes, loss of data and general poor performance.

Termination

SCSI devices are 'open collector', and require a resistor for each signal and control line to 'pull up' the line to 5V. To improve the performance of the system when transferring data, a special arrangement of resistance is used at each end of the bus. These are the *terminator packs*. These provide a match for the recommended characteristic impedance, minimising voltage reflections at each end of the cable, and so improving the reliability of transfers.

Important points to note are:

1 There should be a terminator at each end of the bus.

2 There should **not** be any terminator packs fitted to any devices between the devices which are at each end of the bus. If in doubt, consult your supplier to ensure that the terminators have been removed (if these are not required). Too many terminators can damage both the SCSI expansion card and the SCSI devices.

3 The bus should not have any spurs – in all cases ensure that the bus is electrically a single length of cable. Small spurs may exist inside SCSI devices, but they should not exceed 100mm in length.

4 Unterminated buses can be unreliable.

5 Devices which provide terminator power (TERMPWR) to the SCSI bus allow the bus to function if the device is powered down.

Disconnecting and reconnecting devices from the bus

This applies to the physical connection and disconnection, not the electrical connection/disconnection previously described. A device should **never** be disconnected from the SCSI bus without first ensuring that all the devices connected to the bus are already powered down. This is important as damage may otherwise result.

However, it may be permissible to power down a device on the bus, as long as:

- the data cable remains connected, and
- the device is not providing termination to the bus – or if it is, its own power supply is not the sole source of power to its internal terminator.

Device id

Each device normally has a link-selectable device id. An exception is the SCSI expansion card itself, which has a programmable device id selected by software. By default in Acorn SCSI systems the expansion card id is seven.

It may be necessary to disassemble your SCSI devices to change the device id. Refer to the guides supplied with the SCSI devices. If you don't feel confident doing this yourself, contact your supplier for help.

If two devices have the same id, the system will not function correctly.

Simple fault finding

If you are having a problem making your SCSI devices function correctly, then carefully check the following list of important points.

1 Are all the devices ANSI SCSI compatible?

All disc drives should follow the ANSI SCSI CCS 4B recommendation. Acorn software uses a subset of this specification. Other devices may deviate from this specification but the onus will be on the user of the system to provide the necessary software to drive them. The Acorn SCSI system device driver has the necessary 'hooks' to allow this to happen.

2 Are the cables of sufficient quality?

Remember, poor quality cables will result in an unreliable system. The total cable length (including lengths within the system) should be less than six metres.

3 Are the devices connected in a linear fashion?

The SCSI bus should be one continuous line; there should not be any loops or spurs longer than 100mm, including those within devices.

4 Are the power supply connections of the devices adequate?

Beware of poor earthing and earth loops.

5 Is the bus terminated at each end and only at each end?

The Acorn SCSI card is shipped with terminators fitted.

6 Are all the device id links correctly set?

Refer to the manufacturer's information if you are not sure.

7 Are the terminators powered?

Check the terminator links on the devices. It may be difficult to establish whether they are or not. Once again, if you can't understand the manufacturer's information, contact your supplier.

Switch on and try it. If it doesn't work check everything again carefully. If your SCSI device still doesn't work after going through the fault finding procedure, contact your supplier.

Using non-ANSI compatible devices

If you intend to use non-ANSI compatible devices in your system, then it is best not to connect these until you have a system working with any standard disc drives you intend to use.

For each drive, follow the installation procedure as described in the first chapter, *Installation*. Then add in any extra devices and their associated software one by one, carefully checking termination conditions and device id selection. Be sure to check the operation of all previously installed devices. If you have any problems with a non-standard device, you should be able to isolate the device which is causing the problem and refer to the supplier or dealer.

Remember, Acorn only provides interface software for disc drives which comply to ANSI CCS 4B (see the chapter entitled *SCSI explained* for more details). Any other SCSI devices must either be supplied with interfacing programs, or you must write such a program yourself using the information in the chapter entitled *Interfacing SCSI and RISC OS*.

Using SCSI card termination devices

The SCSI expansion card is supplied with two termination devices (fitted). These are a plug-in PCB assembly and a metal-screened terminator. If you intend to fit an internal SCSI drive to your computer, then connect the drive to the 50-way plug on the expansion card, and fit the metal-screened terminator to the SCSI bus connector. To add further peripherals to the bus, simply remove the metal-screened terminator and connect the peripherals (remembering to terminate the last peripheral in the chain).

- If you intend to use the SCSI expansion card *at one end of* the bus, then fit the plug-in PCB assembly to PL2.

  If you need to use the SCSI expansion card *within* the bus, then both terminators must be removed.

RISC OS operation

RISC OS user operation

This chapter tells you how to configure your RISC OS computer so that it can be used with SCSI discs.

The RISC OS SCSI interface provides high level support for SCSI disc peripherals. However the SCSI card provides general support for all types of SCSI devices. Information to enable programmers to generate additional RISC OS SCSI device drivers is given in the chapter *Interfacing SCSI and RISC OS*.

Loading the modules

The two modules that control the SCSI peripheral (the SCSI driver module and the SCSIFS module) are loaded automatically from the ROM on the SCSI card; they do not need any user installation.

The SCSIDM program

Before the disc can be used it has to be formatted and initialised using the SCSI disc management program (*SCSIDM).

Formatting a SCSI disc

The program has a user interface that provides on-line help. For information on any particular command you can also type **help *command***. For example, to get help information for the command `device`, type:

```
SCSIDM>help device
```

A short explanation about the command is displayed on the screen.

The program operates from the command line. If you are currently using the desktop, press the function key F12 to display the command line prompt at the bottom of the screen.

Follow this procedure to format the disc:

1 Start the disc management program by typing **SCSIDM**. Once the program starts, you will see the prompt:

```
SCSIDM>
```

2 Define the device name of your SCSI disc. The device name is a number from zero to six. For example, to select a disc with device name zero, type:

```
SCSIDM>device 0
Current device is 0
Device identifies itself as a RODIME  RO3000S 2.2
```

If the drive has just been powered on, or the computer has recently been reset, the screen may display `Unit Attention`. If this happens, repeat the command.

3 Now format the disc using the command:

```
SCSIDM>format
Keep the existing bad block list? >yes
Interleave? (1) >1
```

The *bad block list* on the disc should normally be included by typing **yes**. The parameter *interleave* determines the spacing of data on the disc, the parameter is usually one, the tightest spacing. Some very fast discs may require a different interleave factor.

4 Once the disc is formatted, it should be verified by typing:

```
SCSIDM>verify
No. of iterations? (1) >1
Reassign bad blocks (yes)? >yes
Verifying...verify OK
```

The number of times the disc may be verified can be changed by altering the number of iterations. Any bad blocks will normally be reassigned to the spare blocks on the disc.

5 The disc can now be sectioned. Unless you are using RISC iX, answer **no** to the RISC iX partition question and the whole of your disc will be used for RISC OS:

```
SCSIDM>section
Do you want a RISC iX partition? >no
```

6 Your disc is now ready for use. Leave the format program by typing:

```
SCSIDM>quit
```

There are further commands available that allow you to remap and redefine any subsequent bad blocks that may occur on your disc. A full explanation of all of the format commands is given in the chapter entitled, *The SCSIDM program*.

If you have more than one disc to format, repeat this procedure for each subsequent disc

Using the disc

Once you have formatted and sectioned your disc(s), as described above, you must configure the number of discs SCSIFS will have to address, which you do by typing:

***Configure SCSIFSDiscs <n>**

where **<n>** is the number of SCSI discs which have been attached. When you have done this, verify that the discs are recognised by SCSIFS by typing the command:

***Devices**

This command produces a table similar to the one shown here, identifying your SCSI devices:

Device	Type	Capacity	Vendor	Product	Revision
0	Direct-access	100 Mbytes	CONNOR	Cp3100-100mb-3.5	0.0C
1	Read-only	500 Mbytes	HITACH	CDR	0004
2					
3					
4					
5					
6					
7	Host		ACORN	SCSI expansion	0000

Now hold down Ctrl and press Break. When the desktop reappears, the SCSI disc icon(s) will be displayed on the icon bar.

Once installed, a SCSI disc can be used just like any other disc.

You can set the directory cache size in SCSIFS just as you can in ADFS, using the command:

***Configure SCSIFSDirCache <n>**

where **<n>** is the size of the cache you want, in Kilobytes. 16Kb is a typical setting.

You can check the current settings of all the configuration values by typing:

***Status**

Note: It may be necessary on occasions to reinitialise SCSIFS – for example, if you attempt to select a drive which has been powered down, or if the drive has not been formatted and sectioned. To reinitialise SCSIFS, type:

***RMReinit SCSIFS**

RISC OS and SCSI naming conventions

SCSI devices used with RISC OS are given drive numbers starting from four. For example, the first SCSI disc is given the SCSI id number 0, therefore its corresponding RISC OS name will be SCSI::4.

The SCSI: prefix is the filing system name which is used to distinguish the SCSI drive from any of the other filing system drives (eg ADFS) that may have the same number. The :4 is the drive number that RISC OS assigns to the drive.

Set the default SCSI drive number used by SCSIFS at power on with the command:

***Configure SCSIFSDrive <n>**

where **<n>** is the drive number to be selected.

SCSI explained

This chapter is aimed at the experienced programmer who has a general knowledge of disc drives, tape drives and operating systems, but does not have much prior knowledge of SCSI. The chapter is a general discussion of SCSI and does not refer specifically to the interfaces for the RISC OS or RISC iX operating systems. It should be read in conjunction with the ANSI SCSI specifications.

This chapter will be of interest to you if you want to:

- understand more about SCSI interfaces
- write a SCSI device driver
- solve a fault in a SCSI system.

The history of SCSI

Prior to the invention of Winchester disc technology, interfaces between hard discs and computers were complex and expensive. In general hard discs were only used with mainframe computers which had proprietary interfaces. When Shugart Associates introduced their early hard discs, they proposed a standard interface between the hard disc and the evolving microcomputer.

The original Shugart Associates Systems Interface (or SASI) allowed up to eight discs to be controlled from one computer with maximum data transfer rates of about 1Mb/sec to 2Mb/sec. SASI became widely used, particularly by minicomputer and workstation manufacturers (although not by the PC market) who introduced it as the standard hard disc interface for their computer systems.

There were a number of reasons for this wide acceptance:

- it was a well-defined interface
- it was independent of the peripheral manufacturer
- it allowed flexibility in drive sizes and performances
- it saved small computer manufacturers from designing hard disc interfaces.

With the rapid growth in the computer market, minicomputer and workstation manufacturers recognised the need for a standard, high performance, general peripherals interface. In response, the American National Standards Institute (ANSI) used SASI as the basis for its Small Computer Systems Interface (SCSI).

With this, SCSI split into two camps. At least one computer manufacturer retained it as a low-cost, relatively low performance disc drive interface. ANSI, meanwhile, moved it towards a complex and sophisticated general 'block' data peripherals interface, designed to serve the needs of the upper end of the computer industry.

This split has led to a great deal of confusion. In attempting to cover a broad spectrum of requirements and as a result of a large number of influences by interested parties, there are now many pitfalls and traps which may snag the unsuspecting user. However, the interface still offers one of the most flexible high performance data interfaces available today and there is no indication of any significant alternative, particularly where there is the facility to mix types of devices.

A summary of important features

- Up to eight devices may be physically connected.
- More than one *initiator* may be connected to the system.
- More than one *target* may be connected to the system.
- Data may be transferred in two ways – asynchronous or synchronous.
- Asynchronous transfers may occur at up to about 2.5MB/sec.
- Synchronous transfers may occur at up to about 4MB/sec.
- Transfers of data may be 'interleaved' – this exploits the feature of reselection.
- Devices may be of various types such as disc, tape or scanner.

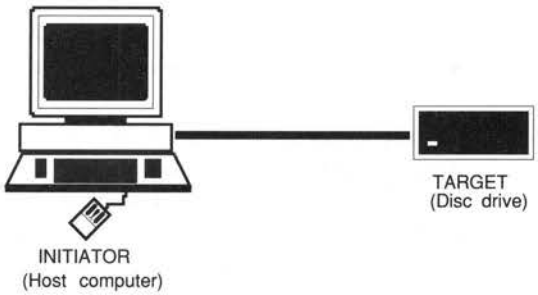
System configurations

Up to eight physical devices may be connected on the SCSI bus. The minimum system is two devices (the Acorn SCSI card and one SCSI device). Devices are either *targets* or *initiators* (under exceptional circumstances they may be both). There must be at least one of each in a system.

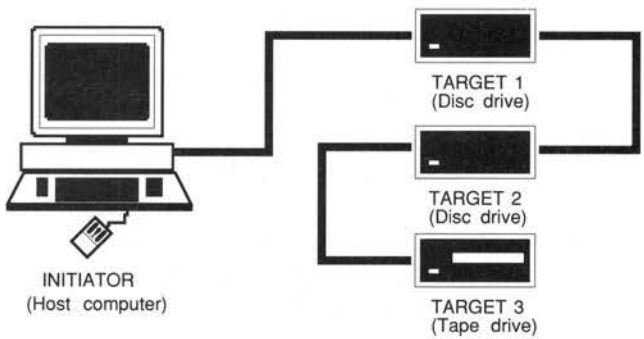
Normally, systems are classed as:

- Single initiator/Single target
- Single initiator/Multiple target
- Multiple initiator/Multiple target

The Acorn SCSI system currently supports only the first two.



A Single initiator/Single target SCSI system



A Single initiator/Multiple target SCSI system

Types of SCSI devices

SCSI devices may be classified into the following categories: block access devices, sequential access devices, and special devices.

Block access devices

In a *Block access device* data on the device may be referred to by an absolute address reference known as a *Logical Block Address* (LBA). To the host computer a Block Access Device will appear as a large array of data blocks, each with its own unique LBA. This array will be monotonic in that each of the possible LBAs will exist. You should refer to the description of remapping later in this chapter, to understand what happens to an LBA in a disc drive if there is a media defect.

A read of a specific LBA will always return the same data until it is rewritten, regardless of read/write operations to other LBAs.

Sequential access devices

Typical Sequential Access Devices are Streaming Tape Drives. In a tape streamer, data may only be located by searching for the data from the start of the tape. Although the data may be grouped together by special markings known as *Filemarks* it is not possible to go directly to a block of data without searching from the beginning (or sometimes the end) of the tape.

Special devices

Any other special SCSI devices may be grouped together as Special Devices. These include:

- scanners
- printers
- processor devices.

For a further description of these refer to the ANSI SCSI specification.

Logical unit numbers

The hardware of SCSI only allows eight devices to be connected on the bus. However, the specification allows for each physical device to be associated with a further eight *Logical Devices*. Each one of these has a unique id which is called the *Logical Unit Number* (LUN). LUNs can be used where more than one peripheral, for example, disc drives, is connected to a SCSI controller card. Each peripheral is identified by a unique LUN.

In modern devices there is normally only one LUN, numbered zero.

Conceptual layers

There are six basic layers into which SCSI may be partitioned:

- threads
- transactions
- transfers
- phases
- bus states
- transfer protocol.

SCSI does not have a clear definition of these layers. However to clarify understanding, all operations on the bus may be classified in this way.

Threads

An *initiator* is a device which starts the process of exchanging information.

Typically an initiator is the microcomputer, and it will request a transfer of data to or from a *target*, such as a disc drive. However, within the initiator there may be a number of separate logical processes wishing to transfer data to or from a number of targets. The SCSI bus provides a way in which a process in the initiator may 'reach out' to data in, for example, a disc drive and also exploit the communications latencies inherent in bulk storage devices.

The 'connection' between the computer and the disc drive which is created is called a *thread*. A thread exists until the process which requested the transfer in the initiator has been completed.

Commands

More than one process in the host may be active concurrently, exploiting the latencies of transfers allows the SCSI bus to be multiplexed between different devices. Every communication using the SCSI bus requires that an initiator should issue a *command* to the target to send or receive data, monitor the status or issue controls to the target.

Transactions

At any one given time the bus is actually *owned* by only one host-target combination. While an initiator is connected to a target, a *transaction* is said to be in progress.

This transfer starts with an Arbitration sequence, where one of the potential bus masters is allowed to resolve ownership of the bus. This is followed by a Selection sequence, in which the successful bus master identifies a target with which to communicate (see Reselection later in this chapter for an explanation of the differences between Selection and Reselection).

Phases

Each part of the transaction is defined by a phase which represents the condition of the bus control lines.

Bus states

The *state* of the bus refers to whether or not it is actually in use at a given time.

The states are:

- bus free
- arbitration
- selection (or reselection)
- information transfer
- low level transfer protocol.

Low level transfer protocol

SCSI is a parallel bus. As each byte is transferred there is a sequence of request and acknowledge strobes known as the Req/Ack handshake. This is described in more detail later.

SCSI bus states

The *state* of the SCSI bus indicates whether or not two devices are actually connected.

Bus free

This is the *idle* condition. No lines are driven (except the RST line during a bus hardware reset).

Arbitration

When a device requires to connect to the bus, it must first arbitrate to see if the bus is free. This arbitration sequence is complicated by the possibility of two or more devices arbitrating at the same time. A competing device first checks that neither BSY or SEL are asserted. It then asserts BSY and the data bus line D0-7 corresponding to its id. After an *arbitration delay* it samples the data bus. If a data line corresponding to a higher priority device is on the bus, then it stops attempting to arbitrate for the bus. If it sees that another device has asserted SEL during the arbitration delay then it immediately stops arbitration until the bus is again in the *bus free* state.

Selection

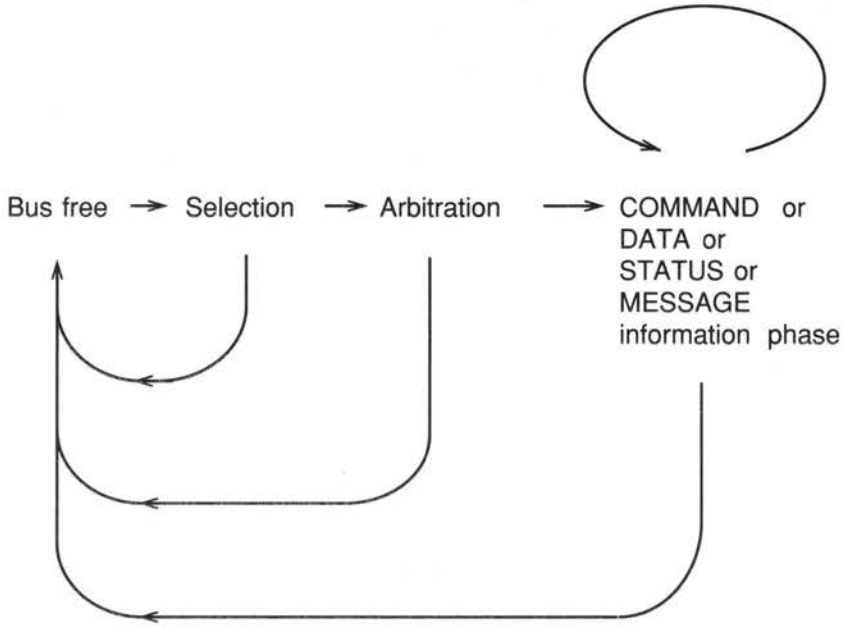
At the end of the arbitration delay, if no other higher priority device has gained the bus, the device drives SEL active. It asserts both its own id and the id of the device to which it wishes to connect. It then releases BSY. The target device sees the condition of BSY not asserted whilst both SEL and its own id are asserted. It then asserts BSY. The initiator detects this condition and

releases SEL. The arbitration phase is complete and the target has been selected. During this selection phase I/O is **not** asserted. In reselection, the process is similar but I/O is asserted.

Information phases

Information phases information can now be transferred on the SCSI bus with the information phases. BSY remains asserted (by the target) until the target releases the bus at the completion of the transfer.

This may be the end of the command transaction, or as part of a disconnection-reselection sequence.



The SCSI bus state cycle

Bus information phases

These are:

- Data in phase
- Data out phase
- Command phase
- Status in phase
- Message in phase
- Message out phase.

The *phase* of the bus is identified by the state of the control lines C/D, I/O and MSG as follows:

	MSG	C/-D	I/-O
Data out	0	0	0
Data in	0	0	1
Command	0	1	0
Status in	0	1	1
*	1	0	0
*	1	0	1
Message out	1	1	0
Message in	1	1	1

where 1 means active (driven *low*) and 0 means inactive.

In all cases the phase of the bus is determined by the *target*. This may seem contradictory, but, after the initiator has made the initial selection of the target, the actual transfers are controlled by the target.

Command phases

This transfers a number of bytes (normally between six and twelve) to the target in which the initiator specifies the operation to be performed, such as read, write or format.

Data phases

This is the phase in which a block of data is transferred between initiator and target. The direction is IN for target → initiator and OUT for initiator → target.

Status phases

At the end of each transaction the target reports a status to the initiator (except under some exceptional error circumstances). The status phase is thus always target → initiator.

Message in phase

This is a way for the target to make specific protocol requests to the initiator. A typical and mandatory message is 'Command Complete', which is sent by the target at the end of each transaction to inform the initiator that the target has completed the operation.

Message out phase

The *Message Out* phase is used by the initiator to notify the target of specific protocol requests. Typically these are connected with selecting SCSI bus transfer modes and responding to error conditions. However, as it is the target which determines the bus phase, the initiator must signal to the target its requirement to generate such a message. This can be done by asserting the ATN (or Attention) line. The target should respond to this (at its own convenience) with a message out phase. When this occurs the initiator should respond with the appropriate message.

SCSI bus messages

A full list of the possible messages which may be exchanged between an initiator and a target would be too long to include here. The full definition of the messages, and the exact circumstances under which they are exchanged may be found in the ANSI specification and the manual for the appropriate device. The following is a list of the more common messages and a brief description of their function.

- Bus Device Reset

  This message is sent from the initiator to reset any target devices in the system.

- Command Complete

  This message is returned by a target as part of the completion of a normal transaction.

- Identify

  Used mainly during selection and reselection sequences to inform devices of the id and Logical Unit Number (LUN) to which they are connected.

- Message Reject

  Sent by a target when it does not wish to accept the most recent message the initiator has sent.

- Save and Restore Data Pointers

  These are used as part of the selection and reselection sequence to indicate to the initiator where to resume the transfer sequence. The Restore Data Pointers message is also used as part of the recovery from bus error conditions. This message instructs the initiator to resume transfers from the location in memory to which the transfer had reached at the conclusion of the last successful data information transfer phase.

- Disconnect

  This message is sent by a target to indicate to the initiator that it is about to disconnect (ie the present physical path is about to be broken). It means that the current operation will be completed by a reconnection later.

Disconnection and reselection

All large capacity peripheral devices are electromechanical; for example disc drives, tape drives and CD-ROM devices. As such, they all suffer from mechanical latency, which is the delay in positioning the transducer (eg read/write head for a disc drive) relative to the data, such that it may be converted to electrical signals. This latency is typically 5–50 milliseconds for a fast disc drive, and may be up to one to two minutes for a tape drive. During this time, any computer system which is trying to transfer data has to wait.

Why a device should disconnect

In the case of high performance machines, particularly those with multitasking capability, this otherwise wasted time can be used for additional processing.

To facilitate this, SCSI has the capability of allowing the initiator to issue requests to transfer data with more than one target device concurrently. The mechanism for this is that where a device recognises that the request cannot be immediately serviced, and where the appropriate control signals have indicated that the initiator can support more than one concurrent command, it releases the bus, making it available for the host to initiate another transfer data to another target. The releasing of the bus is called *disconnection*. At a later stage, when it is ready to resume the transfer, the target will attempt to regain the bus by informing the initiator that it is ready to resume. This sequence of events is called *reselection*.

The SCSI specification does not indicate the exact circumstances under which this should happen. Typically a disconnection will be initiated when a seek of greater than two cylinders occurs for a disc drive, or when a tape needs to search for filemarks.

At the beginning of the command the initiator indicates to the target that disconnection is allowed by asserting ATN.

Mechanism of disconnection

When the target decides that there will be a delay before a command can be completed, and if the initiator has indicated that disconnection is allowed, the target will change the bus control lines to signal the Message in phase and assert REQ to send the target either a Save Data Pointers message or a Disconnect message (Disconnect without a Save Data Pointers message is known as 'Implied Save Data Pointers').

The initiator software should identify this condition and save the current transfer count for the thread that was established so that the command can be completed later.

Reselection sequence

At some later time the disconnected target will wish to reselect. It is possible that other devices will have selected and disconnected in the interim period.

The target may only begin to reselect when the bus is free. It then enters the reselection phase and if the target successfully arbitrates for the bus, the transfer can be resumed.

After the reselection has completed, the target will send the initiator an Identify message with a Message in information phase. This allows the initiator to restore the data pointers and transfer count of the previous thread established to that target.

After the Message in phase, the target will select either Data in, Data out, or a Status in phase to complete the transfer. It is possible for a further disconnection/reselection sequence to occur if the target again decides to do so.

SCSI commands and command descriptor

The way in which a thread of communication may actually transfer data is by software in the computer, typically a device driver, issuing a command to the target.

A command is transferred to the target in a Command Descriptor Block (CDB). Standard definitions for CDBs are 6, 10 or 12 bytes, although it is possible for a vendor to define a new CDB type. The size of the CDB is encoded in the first byte of the CDB. Typically SCSI Protocol Controllers (such as the WD33C93A SBIC used in the Acorn SCSI card) interpret the first byte to detect the length of the CDB automatically. The CDB data generally contains six fields (many of which are not used for most commands):

- the Group of the command
- the OpCode of the command
- the Logical Unit Number
- the Address of Data transfer (LBA)
- the Number of Blocks or Bytes to transfer
- the additional control field.

The first byte is Group and Opcode. There are eight possible groups, of which normally only Group 0 and Group 1 commands are used, although many devices use Group 7 commands for maintenance and test operations. The Opcodes are Read, Write and Format. Some typical Opcodes will be described later.

The second byte (Byte 1) has a Logical Unit Number encoded in its three most significant bits.

Issuing a command to a device

The device driver will receive an instruction to transfer a block of data, together with information describing its location in memory, and the location of the data on the target device. It must calculate the number of logical blocks of data that the transfer represents. The Command Descriptor Block (CDB), is then constructed. In the case of the Western Digital WD33C93A SCSI controller as used in the Acorn SCSI Card the CDB is loaded into registers within the WD33C93A. Other registers are loaded to specify the number of bytes to be transferred and the id of the target etc.

A special control code is issued to the WD33C93A to instruct it to negotiate for the bus, and then select the target device when it has gained control of the SCSI bus.

Until the transfer is completed, the Thread is said to exist between the data area in the computer and the corresponding data area in the target device.

General commands for all device

- Read – move data from target to initiator
- Write – move data from initiator to target
- Request Sense – returns error status
- Inquiry – describes the target device
- Mode Sense – return target setup parameters
- Mode Select – set target setup parameters.

Particular commands for disc devices

The operations for disc drives (and other Block access devices) which are normally used are:

- Read Capacity – return the size of the drive
- Format – reinitialise data fields
- Verify – check media
- Remap Defects – reassign defects.

Particular commands for tape devices

Special commands for sequential access devices – normally streaming tape drives include:

- Write Filemarks – place id markers on tape
- Rewind – return tape to start
- Space – move along data/filemarks
- Retension – spool forwards and backwards and reset heads.

For a full definition of these commands see the manufacturer's manuals and the ANSI specifications.

SCSI low level interface

The SCSI bus consists of nine data lines and a number of control lines as follows:

Electrical interface

BSY	When asserted indicates that the bus is in use.
SEL	Indicates that the ownership of the bus has been determined.
C/D	(Command/Not Data) indicates whether a command or status is being signalled or whether a transfer of data in or out is occurring.
MSG	Asserted when a message is being transferred.
I/O	When asserted information is being transferred from the target to the host.
ATN	Used to indicate a special Attention condition.
RST	Bus hard reset signal.
REQ	Transfer request (always from target).
ACK	Transfer acknowledge (always from host).
D0..7	Data lines 0 to 7 which carry SCSI data.
DBP	Data bus parity signal used in bus parity.

Note that all SCSI signals are open collector (or open drain) and are active low, (asserted by turning on a driving device). This means that they must be terminated by resistor packs. These resistor packs must be of the correct value and correctly located on the bus to function properly. With high performance SCSI systems this is very important.

Req/Ack handshake of data

Originally all information interchanges between initiator and target had one low level protocol. This was a simple asynchronous Req/Ack handshake. The mechanism was as follows:

For initiator → target transfers:

1 Target asserts Req by driving it low.

2 Initiator drives the data onto the data bus.

3 Initiator asserts Ack by driving it low.

4 After a settling delay the Target strobes in the data and deasserts Req to show that this has happened.

5 The Initiator deasserts Ack to begin the next cycle.

For target → initiator transfers:

1 Target drives the data bus to the new data value.

2 After a settling delay the Target asserts Req to show that there is valid data.

3 The Initiator asserts Ack and strobes in the data.

4 The Target deasserts Req.

5 The Initiator deasserts Ack to begin the next cycle.

The ANSI SCSI specification defines the timings in detail.

Synchronous and asynchronous transfers

In an attempt to improve performance the simple asynchronous transfer protocol has been enhanced with the synchronous protocol. In this each byte is not individually strobed by a full two part handshake. As each byte is transferred the target will issue a Req strobe. To indicate that each byte has been received the initiator issues an Ack strobe. However, the two are not directly synchronised. To ensure that sufficient Reqs and Acks are issued the target and initiator 'negotiate', as part of an initialisation sequence, a Req/Ack handshake offset which represents the maximum number of Acks which may be outstanding at any given time.

Normally, this will represent the maximum size of any FiFo in the initiator and target. It allows a significant increase in transfer rate up to 5Mb/sec.

Synchronous transfer protocol is an option not currently supported by Acorn SCSI software.

The synchronous transfer may only occur during the Data phase (Data in or Data out).

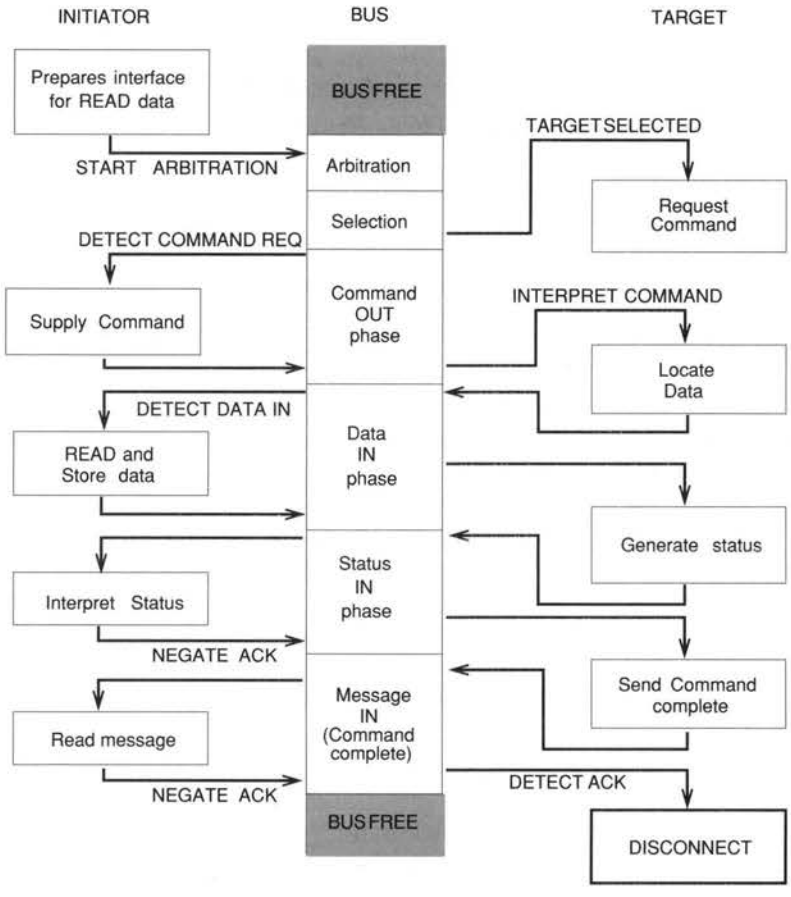
Negation of Ack

In certain circumstances (normally the processing of error conditions) it becomes necessary for the target to temporarily suspend transfers on the bus. The initiator continuously asserts the Ack signal until it is ready to continue.

The WD33C93A will interrupt the host computer but leave the Ack asserted. The computer can then decide what to do next and negate the Ack signal when it is ready to continue.

A typical disc read operation

A typical (and simple) phase sequence for the read of a block of data without reselection:



Read without reselection

Device error conditions

There are three classes of errors associated with the use of SCSI:

- errors within the target device
- SCSI bus errors
- errors within the initiator device.

Errors within the target device

These are errors which the target firmware identifies and reports to the initiator via standard SCSI commands.

Normally, when a command finishes, the target returns a SCSI status byte to the initiator. If there is no error this has the value 0 which is known as a *Good Status*.

If the *Device Check Condition* bit is set in the SCSI status byte, then this indicates to the initiator that target has detected an error or exception condition. The normal response to this is for the target to issue a *Request Sense* SCSI command. The target should then return data to indicate the error. This is called the *Sense Data*. Typically 12 to 18 bytes are returned. This is called *Extended Sense*. For a full definition refer to the SCSI ANSI specification, and also to the device manufacturer's manual. There is a range of interpretation as to the nature of the Sense Data.

However, for disc drives which comply to CCS 4B the interpretation is well defined.

The request sense command

Sense data is returned to the initiator device with the Request sense command. This command can specify the number of bytes of sense data to return. For disc drives compliant to CCS 4B it is mandatory to support the Extended Sense data format. This has the following information fields:

- Error Code – this is always zero
- Error Class – this is always seven
- Valid Info Bit – this is 1 if Information Field is valid
- Sense Key – a code defining the type of error
- Information Field – the location of the error if it is a media defect
- Additional Sense – an additional error code byte at an offset of 12 within the sense data. This may be used to give more information about the error.

CCS 4B defines the meanings of the Sense Key and the Additional Sense byte.

Types of errors indicated by the sense key

The device errors reported are normally one of:

- hardware error
- illegal request
- medium error
- unit attention.

For a full list see ANSI X3.131-1986 Table 7-6.

Hardware and medium errors are normally more fully described by the additional sense byte. For example, a medium error may have an additional sense byte of 12H which is defined by the CCS 4B specification (ANSI X3T9.2/85-52 Rev 4B) as 'No Address Mark found in id field'.

Illegal Request indicates that the CDB used to send the command which caused the Check Condition had an error in one of its fields. Specific errors depend upon the command and the implementation of the command by the manufacturer. The ANSI specification, the Command Set and the manufacturer's manual must all be studied to determine the cause of this error.

Unit Attention is a 'pseudo-error' condition, which all disc drives which are compliant to CCS 4B must support. It indicates that the rejected command was the first command to be received since the drive was last powered on or was reset (either by a hardware Bus Reset or since receiving a reset command). Note that the exact rules governing a *Device Check Condition* generated by a Unit Attention state are complex. For example, an *Inquiry* command **does not** affect the Unit Attention condition. Unit Attention should not be confused with the Attention condition which an initiator may generate as part of a bus error handling protocol.

SCSI bus error conditions

These are errors which are caused by problems in the low level protocol. The most common errors are:

- parity error
- failure to select a device (device selection timeout)
- data path error.

If a transient parity error is generated and the system has parity enabled, then the detection of this error is done by the device which is receiving the

data at that time. This may be the initiator or the target depending upon the current bus direction.

Generally, parity errors are handled by aborting the current command. This may be done in a number of different ways depending at which stage the error occurred. There are also a number of options, for example:

1 If the error was in a command phase, the target may send a 'Restore Pointers' message. This will indicate that the initiator should repeat the CDB.

2 The target may immediately go to the *Bus Free* state without generating a Sense Key error.

The handling of errors is not well described in the ANSI X3.131 specification but CCS4B does have a section recommending error handling (page 41 of X3T9.2/85-52). Some manufacturers publish information describing the response of their drives to such error conditions.

If a device fails it is most likely that it will not respond to being selected. This will give a *Selection Timeout* error.

A data path error results from a failure in a bus driver IC or a circuit failure such as a dry joint or open circuit. It is not possible to predict exactly what may happen in all such cases. The most likely effect is that the system will 'hang' and a device driver should have an overall transaction timeout and return with some form of error.

Data path errors in the Acorn SCSI Card may result in the Western Digital WD33C93 returning an 'Unexpected Bus Phase' error – this may help to isolate the problem.

Errors within the initiator device

The complexity of the SCSI protocol means that an electrical error or IC failure within the initiator interface will have unpredictable results.

Once again a transaction timeout in the device driver software may provide some indication of the problem.

Defect handling

In all disc drives (not just SCSI), minor flaws in the media may occur at manufacture or during use. These 'media defects' may arise because of chemical imperfections in the media, or sometimes because of slight mechanical damage.

How a defect is remapped

As has been seen earlier, SCSI embodies the principle of Logical Block Addresses in that the host operating system does not need to be concerned with the actual physical location of data on the disc. Because of this, SCSI drives typically offer the facility of remapping media defects. This remapping is done by using spare sectors which are made available at manufacture. The exact mechanism by which the sector may be remapped varies from drive to drive.

There are typically three mechanisms used for remapping:

- Each track is formatted with a 'spare' sector.
- For a given cylinder, the spare sector of one track may be used with another track.
- Spare tracks are made available for cases where there are more on one track than may be remapped by one and two above.

Not all drives support the second mechanism.

Where a defect occurs in a sector which is the only defect on that track, this is normally remapped with the spare sector operation. The technique is as follows:

1 Recover the data by reading and applying correction if possible.

2 Mark the bad sector as unusable.

3 Transfer the data into the spare sector.

4 Verify that the spare sector data can be read.

5 Mark the id field of the spare sector as that of the original, now defective, sector.

An important point is that there is no 'look up' mechanism to determine the defective sector – it simply isn't there any more, as the spare sector now has the same id field. When the drive is searching for the id field as part of the data read it will now find the spare sector instead of the defective sector.

When all the available spare sectors on a cylinder/track have been used up then a spare track must be used. These are normally reserved on the disc just beyond the main data area (that is nearer to the hub). Generally there are about 10 spare tracks/100Mb of disc capacity.

The algorithm is typically as follows:

1 Find spare track and format it with the correct interleave and sector size.

2 Copy over all the data from the track which has the new defect on it.

3 Verify the data on the new track.

4 Mark the old track with a defective track mark to indicate that it is no longer in use.

Defect descriptor tables

There are three tables used to determine to the drive where a defect may lie:

- production defect list – PLIST
- growing Defect List – GLIST
- defect map.

Unfortunately these tables are confusingly different, although for quite valid operational reasons, and so must be studied carefully to be fully understood.

The PLIST is created at manufacturing time (normally) as part of the manufacturing process. Frequently it is first determined using an analogue scanning technique and then extended with digital scanning techniques. The PLIST is expressed in 'bytes from index' format because at the time it is determined, the drive format is not known.

The GLIST is created by the drive as the result of executing a *remap* command.

The defect map is created by maintenance software from the results of a verify operation on a disc drive. The defect map refers to defective sectors by their Logical Block Address.

Defect scanning – the verify command

The SCSI command provided to allow the user to scan a disc drive for defects is the *verify* command. This command performs a read of the media without transferring the data across the SCSI bus.

If a defect is encountered the *Device Check Condition* status is generated. The initiator must then issue a *Request Sense* command to determine the fault. Note that the device driver must be able to return the data from the Request Sense to the verifying program, as it is this information which indicates whether it is a media defect and, if so, the location of the defect.

As the media is scanned, then the location (as a Logical Block Address) of any defects which have not been previously remapped will be identified. This

can be built into a defect map.

It should be noted that *verify* does not have a Data Phase.

The remap defects command

After the defect list has been created, it is sent to the device with the *Remap Defects* command. For each defect which is supplied the drive will perform its remapping as described previously. All the supplied defects will be added to the GLIST.

The GLIST refers to defects in 'Physical Sector Format'.

Final points

As has been described, SCSI has evolved over a period of time, and it is only recently that standardisation across the industry has been possible with the introduction of the ANSI standard and then the Common Command Set.

Non-ANSI compatible devices

Efforts so far have been concentrated upon disc drives and block access devices generally. There are a wide range of devices which do not comply to the ANSI specification.

Acorn SCSI software provides 'hooks' to allow programs to interface to devices which are not fully ANSI compatible.

The future of SCSI

Nearly all high performance interface devices may now be connected by SCSI to computers. It represents a flexible and high performance data transfer mechanism with a wide range of implementation options. The standardisation offered by ANSI SCSI allows many high performance peripherals to be interchanged without significant reinstallation. The ANSI committee has produced a SCSI–2 specification to allow for advances in peripherals technology and to extend the scope of SCSI still further.

Interfacing SCSI and RISC OS

This chapter gives a list of SCSI related RISC OS commands. These are mostly of interest to programmers wishing to write alternative SCSI drivers for RISC OS.

Introduction

The SCSI device driver for RISC OS is a relocatable module whose interface consists of the following SWI calls:

- SCSI_Version
- SCSI_Initialise
- SCSI_Control
- SCSI_Op
- SCSI_Status
- SCSI_Reserve
- SCSI_List

and the following * command:

- *Devices

The interface is designed to enable programmers to write alternative device drivers for RISC OS.

The SWI calls

All SWIs conform to the RISC OS standard:

- Any registers not explicitly defined as result parameters are preserved.
- The processor flags N, Z and C are preserved.
- Errors are indicated by setting V and returning with R0 pointing at an error block. (If the bit 17 clear version is called, control passes to the error handler instead.)
- The SWI cannot be called from the background unless this is specifically documented.

- All SWIs preserve the interrupt state unless stated otherwise, and so they may be called from interrupt and event routines.
- All SWIs are re-entrant.

Device and host ids

The device id is a composite of the SCSI id number, the logical unit number (LUN) within the device and the logical slot number of the SCSI card. They are packed into eight bits as follows:

- bit 2..0 SCSI device id (0..7)
- bit 4..3 logical SCSI card number (0..3)
- bit 7..5 logical unit number (0..7).

The SCSI card numbering is based on the relative slot positions of any SCSI cards in the machine. Thus in a machine with two cards, in physical slots 2 and 4 both are referred to as logical SCSI card numbers 0 and 1. In a machine with one card it may be placed in any slot, where it is referred to as logical SCSI card 0.

The host id is a similar composite, but of SCSI id number and logical slot number, with no LUN field:

- bit 2..0 SCSI device id (0..7) – usually 7
- bit 4..3 logical SCSI card number (0..3).

The host device id is usually seven and there must be no other devices on the bus with the same id.

> WARNING:
>
> The use of SWI SCSI_Control 1 (Abort op), or the use of command timeouts, may prove unreliable with certain versions of the WD33C93A SCSI bus interface controller. The current version of the chip, revision C, is known to exhibit problems, which are expected to be solved in later versions.
>
> The SBIC has tendency to hang if a Raise ATN command (issued by the device driver abort command code) is issued during a data-out phase.

SCSI_Version (&403C0)

Determines the version number of the drivers

On entry

No parameters

On exit

R0 bits 0–7 software minor version number; 0 for any release
bits 8–31 software major version number (eg 100 for 1.00)

R1 bitset of software features

R2 bits 0–7 hardware minor version number, 0 for any release
bits 8–31 hardware major version number (eg 100 for 1.00)

R3 bitset of hardware features

Use

This is used to determine the version number of the drivers, whether it supports any particular later extension, and what particular hardware implementation is present.

The bitset of software features is defined as follows:

Bit	Effect
31	Supports target mode
30	Supports configuration
29	Supports reading control lines
28	Supports device reservation
27	Supports multiple SCSI cards (and hence SWI SCSI_List)
26	Supports *SCSIBlock
25..16	Reserved, must be 0
15..7	Reserved, must be 1
6	Supports *Devices
5	Supports scatter list for SWI SCSI_Op
4	Supports messaging
3	Supports automatic Request Sense
2	Supports command queueing
1	Supports background data transfer
0	Supports target disconnect/reconnect

The bitset of hardware features is defined as follows:

Bit	Effect
31	Supports configuration
30..16	Reserved, must be 0
15..1	Reserved, must be 1
0	Supports target mode

Errors: none.

SCSI_Initialise 0 (&403C1)

Reset bus

On entry

R0 0
R1 host id

On exit

No results

Errors

&20100 – No room for SCSI driver workspace.
&20103 – SCSI bus failed to reset.
&20104 – Invalid SCSI host id.

Use

Resets all options set by SCSI_Control to their default states, and removes any device reservations. In addition, it sets a new host ID and causes a full reset of the SCSI bus by driving the SCSI bus RESET line.

The default states set for each device are:

Reservation	none
Timeout = 0	no timeout
Error response	report check condition as an error
Queue behaviour	wait until slot becomes free
Disconnect	allow disconnect/reselect

SCSI_Initialise 1 (&403C1)

Reset device

On entry

R0 1
R1 device id
R8 access key

On exit

No results

Use

Reset device – initiates the reset operation and returns immediately.

Errors

&20105 – Invalid SCSI device id.
&2010E – SCSI driver reservation error.

SCSI_Initialise 2 (&403C1)

Determine device – performs an Inquiry command to see what type the device is and then performs the appropriate Read-Capacity commands depending on the device type.

On entry

R0 2
R1 device id
R2 pointer to buffer for description (16 bytes, word aligned)

On exit

Buffer filled in

Offset R2 →

0 Type =	&00	Direct-access	(eg magnetic disc)
	&01	Sequential-access	(eg magnetic tape)
	&02	Printer device	
	&03	Processor device	
	&04	Write-once read-multiple	(eg some optical discs)
	&05	Read-only	(eg some optical discs)
	&06–7E	Reserved	
	&7F	Logical unit not present	
	&80–FF	Vendor unique	

1 Removable medium bit (bit7)/Device-Type Qualifier (bits 0 to 6)
2 ISO (b7 to 6)/ECMA (b5 to 3)/ANSI (b2 to 0) version
3 Reserved (reserved field returned by inquiry command)
4 Additional length, extra data that could be obtained by an inquiry command
5 Reserved (by device driver – currently zero)
6 Reserved (by device driver – currently zero)
7 Reserved (by device driver – currently zero)
8 to 11 word highest logical block number (converted to normal)
12 to 15 word block length (arm byte sex)

Errors

&20105 – Invalid SCSI device id.
&2010A – Timeout.

Bytes 0 to 4 are the first five bytes returned by an inquiry command, byte 5 is provided to indicate how much additional data (vendor id, product id etc) could be obtained by an Inquiry command. Bytes 5 to 7 are reserved by the device driver.

Bytes 8 to 11 (word 2) and 12 to 15 (word 3) are the highest block count and block size returned by a read capacity command, but they have the byte sex reversed to suit ARM's LDR instruction.

SCSI_Initialise 3 (&403C1)

Enumerate device – performs an Inquiry command to see what type the device is and then performs the appropriate Read-Capacity. Data returned as formatted text.

On entry

R0	3
R1	device id
R2	pointer to buffer for description (word aligned)
R3	buffer size

On exit

The data returned is a zero-terminated string of the same format as that printed by *Devices, ie:

```
ddd  tttttttttttttttt cccc Mbytes vvvvvvvv pppppppppppppppp rrrr
```

where:		
	ddd	device number
	tt...tt	device type
	cccc	device capacity
	vv...vv	vendor
	pp...pp	product
	rrrr	revision number

Errors

If the device does not respond, a string containing just the device number is returned.

If the device responds, but the Read-Capacity call fails, the capacity field will be set to `unknown`.

SCSI_Control 0 (&403C2)

Abort device

On entry

R0 0
R1 device id
R8 access key

On exit

No results

Use

Abort device – cancels any outstanding operation on a device. Abort op should normally be used instead wherever possible.

Errors

&20105 – Invalid SCSI device id.

&2010E – SCSI driver reservation error.

Issues various control commands determined by a reason code in R0. Any additional parameters are given in subsequent registers.

SCSI_Control 1 (&403C2)

Abort op

On entry

R0 1
R1 device id
R2 operation id as returned by SCSI_Op
R8 access key

On exit

No results

Use

Abort op – cancels a particular operation on a device.

Errors

&20105 – Invalid SCSI device id.

&2010E – SCSI driver reservation error.

Note: SCSI_Control 2 is reserved for future use.

SCSI_Control 3 (&403C2)

Set timeout

On entry

R0 3
R1 device id
R2 timeout value in centi-seconds or zero for no timeout
R8 access key

On exit

R0 preserved
R1 preserved
R2 previous timeout setting

Use

Set timeout – sets the default timeout for a device.

Errors

&20105 – Invalid SCSI device id.
&2010E – SCSI driver reservation error.

SCSI_Control 4 (&403C2)

Set error response

On entry

R0 4
R1 device id
R2 0 If device returns check condition, return that as an error message. This is for badly behaved devices with non-standard request sense return blocks.

1 If device returns check condition do the request sense automatically, report the error.

2 If device returns check condition, do a request sense automatically, if unit attention, ignore it and reissue the command, else report the error.

–1 No action, return current setting.

R8 access key.

On exit

R2 Previous setting

Use

Set Error Response – controls the behaviour of the driver on receipt of a check condition status.

Errors

&20105 – Invalid SCSI device id.
&2010E – SCSI driver reservation error.

SCSI_Control 5 (&403C2)

Set queue behaviour

On entry

R0 5

R1 device ID

R2 0 If queue is full, loop until a slot becomes free. If a command operates totally in the background, control returns to the caller as soon as the command is queued. If the command operates in the foreground, control returns when the command completes.

1 Reject if queue is full. The command is accepted only if it can be queued/run without waiting. If a command operates totally in the background, control returns to the caller as soon as the command is queued. If the command operates in the foreground, control returns when the command completes.

2 Reject if the device is busy (or queue is full). The command is accepted only if there are no other queued/running for this device.

3 Reject if expansion card is busy. Accept command only if it will run immediately.

–1 No action, return current setting.

R8 access key

On exit

R2 Previous setting

Use

The driver supports disconnection/reselection and queueing of commands, this implies that when a command is issued, the expansion card hardware may be in use servicing another command, a command may be queued/running for the target device or the queue may be full.

Errors

&20105 – Invalid SCSI device id.

&2010E – SCSI driver reservation error.

SCSI_Control 6 (&403C2)

Disconnect

On entry

R0 6

R1 device ID

R2 0 Allow disconnect/reselect (ie, when the bus selection phase is complete, the driver will attempt to send an IDENTIFY message with bit 6 set. This tells the target that it may disconnect if it wants to).

1 Inhibit disconnect/reselect. Once started, a command will proceed to completion without allowing any other command to start/reconnect (ie, when the bus selection phase is complete, the driver will attempt to send an IDENTIFY message with bit 6 clear).

2 Inhibit disconnect/reselect by not sending an IDENTIFY message

–1 No action, return current setting.

Note: Codes 0/1 cause SWI SCSI_Op to attempt to send an identify message once the bus selection phase is complete. The target may choose not to take the message (not an error), in which case the driver proceeds to the command phase. The target may reply to the message with a message reject. These two cases do not produce an error from SCSI_Op, as the command may still complete successfully. It will however remain connected all the time.

Applications/filing systems are NOT expected to issue this call, as all software/hardware should work happily regardless of disconnections that may occur.

It should only be used by:

1 Users, in their boot files, if their particular devices misbehave when sent messages

2 Application and filing system software specific to a SCSI device that is known to misbehave when sent messages.

On exit

R2 Previous setting
R8 access key

Use

Disconnect – Allow/Inhibit disconnect/reselect, device must be idle.

Errors

&20105 – Invalid SCSI device id.
&20109 – Device not idle.
&2010E – SCSI driver reservation error.

SCSI_Op (&403C3)

Issues a SCSI command to a device and can be called in the background

On entry

R0 b0–b7 device id

b25–b24 00 → no data transfer, 01 → read, 10 → write, 11 → reserved

b26 Scatter bit. If set, R3 is a pointer to a scatter list

b27 If clear, poll escape during transfer and abort if escape pressed

b28 If set, repeatedly retry on timeout

b29 Set if a background transfer (possibly 0 length).

R1 Length of SCSI control block.

R2 Ptr to SCSI control block.

R3 RAM pointer for start of transfer, or pointer to a scatter list of address length pairs, if bit 26 of R0 is set. After each chunk in the scatter list has been transferred, the address should be incremented and the length set to 0.

R4 Length of foreground part of transfer (in bytes).

R5 Timeout in centi-seconds or 0 for default timeout.

R6 Address to call back when transfer complete if background bit set.
The call is made in IRQ mode with IRQs disabled. The routine addressed by R6 may enable IRQs if it wishes to do so. This does not effect its re-entrancy.
If an error has occurred, V is set, R0 points to a non-dynamic error block, (ie a pointer to a fixed block in the module rather than built up in a buffer), R1 indicates the cause of an error and R2 should be the logical address on the device where the error occurred.

R7 Workspace pointer to return in R12 for background call back.

R8 Access key.

On exit

V clear

R0 Returns a 32 bit id incremented for each operation. Used to cancel op.
R3 Updated to indicate how much of transfer was successful. (If scatter list, pointer to first unused/partially used entry.)
R4 Updated to be amount untransferred.

V set

R0 Points to an error block.
R1 Indicates the cause of an error.
R2 Holds the logical block address of the device on which the error occurs.
R3 Updated to indicate how much of the transfer was successful. (If scatter list, points to first unused entry. Entry is updated to indicate amount transferred.)
R4 Updated to be the amount transferred.

Errors

See the following section, *SWI error messages* for more information.

SCSI_Status 0 (&403C4)

Check device status

On entry

R0 0

R1 device id

On exit

R0 status

1 – idle

2 – busy

Use

Check device status – returns the status of the selected device.

Errors

&20105 – Invalid SCSI device id.

Device reservation

The following SWIs use R8 as an access key:

- SCSI_Initialise (with R0 = 1 or 2)
- SCSI_Control
- SCSI_Op

Any of these SWIs can return the error &0002010E (Reservation error) if an attempt is made to access a reserved device without quoting its key.

The SWI SCSI_Reserve is used to control this mechnaism. This SWI allows the caller to claim exclusive use of a device, and may prevent data corruption problems that may arise if a program (by accident or by design) issues a Write/Format/Mode Select command to a device currently mounted by SCSIFS. It also provides for a clean change of device ownership from one user to another, as SCSIDM, for example, is entitled to modify your disc (to map out defects, format and partition it), but can only do so once SCSIFS has closed down all open files and dismounted the disc.

Passing access key value to SCSI SWIs

The access key (typically a workspace pointer) should be passed in R8 and is only checked if the specified device is claimed and the call/reason code is dangerous.

Note: BASIC only passes R0...R7 on a SYS call. This does not matter unless the BASIC program wishes to reserve the device. In this case, the program should include a small piece of ARM assembler code that loads the required access key into R8, then calls the required SWI.

SCSI_Reserve 0 (&403C7)

Claim – claim exclusive used of a device

On entry

R0	0
R1	device id
R2	release address
R3	workspace pointer, passed in R12 when release address is called
R8	access key

On exit

VC

VS and R0 → error

Use

This call attempts to claim exclusive use of the given device. If the call is succesful, the caller will be granted sole use of the dangerous SCSI driver calls for that device, and may use them itself, only by supplying the access key registered at Claim time. Other callers may only use safe calls, such as SCSI_Initialise-DetermineDevice.

If the device has already been claimed, an error message Reservation error will be returned.

Errors

&00020105	Invalid SCSI device id
&0002010E	SCSI driver reservation error

SCSI_Reserve 1 (&403C7)

Force claim

On entry

R0 1
R1 device id
R2 release address
R3 workspace pointer, passed in R12 when release address is called
R8 access key

On exit

VC

VS and R0 → error

Use

This call is similar to Claim, but if the device has already been claimed, it will ask the current claimant to release it by calling the release address registered for that device.

The current owner should try to tidy up, and call Release, but may refuse by returning VS and R0 pointing to an error message.

Note: This call is used by SCSIDM to ask SCSIFS to close files and dismount.

Errors

&00020105 Invalid SCSI device id
&0002010E SCSI driver reservation error, or an error message passed back by the current claimant.

When the current claimant is called, R0 will hold a reason code indicating:

R0 = 1 called from ForceClaim

SCSI_Reserve 2 (&403C7)

Release – will remove the reservation of a device.

On entry

R0 2
R1 device id
R8 access key (to prove you actually claimed the device)

On exit

VC
VS → error

Errors

&00020105	Invalid SCSI device id.
&0002010E	SCSI driver reservation error.

SCSI_List (&403C8)

This is used by the SCSILog module as part of the multiple SCSI card initialisation sequence.

On exit

R0 Null terminated list of expansion card addresses.

Use

This call returns a pointer to a null terminated list of SCSI expansion cards available to SCSIdriver. The addresses are those of the slow access space for the expansion card, ie &03240000, 03244000, &03248000 or 0324C000, for expansion card slots 0,1, 2 or 3.

SCSIFS SWI calls

The following SWIs are provided by SCSIFS:

SCSIFS_DiscOp	(&40980)
SCSIFS_Drives	(&40982)
SCSIFS_FreeSpace	(&40983)
SCSIFS_DescribeDisc	(&40985)

For entry and exit conditions, see FileCore_DiscOp, FileCore_Drives, FileCore_FreeSpace and FileCore_DescribeDisc in the *RISC OS Programmer's Reference Manual*.

The SWI SCSIFS_TestReady (&40986) is described on the next page.

SCSIFS_TestReady (&40986)

On entry

R1 Drive number (4...7)

On exit

R0	0	drive not present
	1	not present, or present but not ready
	2	drive present and ready

SWI error messages

This section contains a list of error messages relating to the SCSI SWI calls.

The errors reported by the SCSI driver fall into three classes:

- Those generated or detected by the driver.
- The interpretations of the SCSI status byte returned on command completion.
- The interpretation of the sense data (collected on Check Condition status if the error response is set to 0 or 1).

Device driver errors

Error numbers start at &00020100

&00020100	No room for SCSI driver workspace
&00020101	Unknown SCSI SWI number
&00020102	Unknown reason code for SCSI SWI
&00020103	SCSI bus failed to reset
&00020104	Invalid SCSI host id
&00020105	Invalid SCSI device id
&0002010A	Timeout, during selection phase
&0002010B	Timeout, during any other phase
&0002010C	Command queue not empty
&0002010D	Command queue full
&0002010E	SCSI driver reservation error
&0002010F	Invalid parameters
&00020110	Parameter error
&00020111	Not callable from IRQ routine (reported by expansion card loader)
&00020112	Operation aborted
&00020113	Unexpected disconnection

Interpretation of the returned SCSI error byte

On completion of the command, the target returns a status byte. The vendor unique bits are masked out and the following interpretations placed on the result:

00000000 *Good* – The target has successfully completed the command.

Any status code other than *Good* is reported as an error, these are:

00000010 *Check condition* (if error response set to level 2)

00001000 *Busy*.

Any other value, and all values with bit 7 set, is reported as error Target status – unknown.

The RISC OS error messages returned are:

&00020180	Target status – check condition
&00020181	Target status – busy
&00020182	Target status – unknown

If the error response level is 1 or 2, then on *Check condition*, the driver will automatically perform a request sense.

Interpretation of the sense data

The first byte of returned sense data is masked to leave the error-class/error-code code bits, these are interpreted as:

0x to 6x	vendor unique error	(non-extended sense)
71 to 7E	reserved error	(extended sense)
7F	vendor unique error	(extended sense) These are reported as: Target error – unknown
70		(extended sense) The error is given by examination of the sense key in byte 2 of the sense data.

Interpretation of the 'sense key' field of an 'extended sense' data block

Error numbers start at &000201C0

&00201C0	Target error – No sense
&00201C1	Target error – Recovered error
&00201C2	Target error – Not ready
&00201C3	Target error – Medium error
&00201C4	Target error – Hardware error
&00201C5	Target error – Illegal request
&00201C6	Target error – Unit attention
&00201C7	Target error – Data protect
&00201C8	Target error – Blank check
&00201C9	Target error – Vendor unique
&00201CA	Target error – Copy aborted
&00201CB	Target error – Aborted command
&00201CC	Target error – Equal
&00201CD	Target error – Volume overflow
&00201CE	Target error – Miscompare
&00201CF	Target error – Reserved sense key.

Non-extended sense, or extended sense, but error code <> 0

&00201D0	Target error – Unknown.

Errors produced by calling SWI SCSI_Op

R0 → error message	A standard RISC OS error block, consisting of a one word error number, followed by an error message.
R1 = error indication	This is a composite of the LSB of the above error number and the status, sense-errorclass and sense key that produced the error.

R2 = logical address on the device where the error occurred.

Device driver detected errors

R1 MSB	The LSB of the RISC OS error number, ie 0 to 255
R1	0 (see note below)
R1	TBA
R1 LSB	TBA
R2	Undefined
R3	Undefined
R4	Undefined.

Status byte errors

Non-extended sense returned

R1 MSB	The LSB of the RISC OS error number. ie 128 to 195
R1	0 (see note below)
R1	0
R1 LSB	Returned status byte (unmasked)
R2	Undefined
R3	Indicates amount of data transferred
R4	Amount of data not transferred.

Returned sense data

R1 MSB	The LSB of the RISC OS error number, ie 196 to 255
R1 UMB	Bits 23 to 16 hold byte 0 of the returned sense data. Bit 23 is the Valid bit and is set if the Logical block address in the returned sense data (in R2) is valid. Bits 22 to 20 are the error class. Bits 19 to 16 are the error code.
R1 LMB	Bits 15 to 13 are the vendor unique bits from byte 1. Bits 12 to 8 are zero.
R1 LSB	Bits 7 to 0 are zero.
R2	Logical block address returned in bytes 1 to 3 of the returned sense data (the vendor unique bits masked out).
R3	Indicates amount of data transferred.
R4	Amount of data not transferred.

Extended sense

R1 MSB	The LSB of the RISC OS error number, ie 196 to 255
R1	Byte 0 of the returned sense data. Bit 23 is the Valid bit and is set if the Information bytes in the returned sense data (now in R2) are valid. Bits 22 to 20 are the error class. Bits 19 to 16 are the error code
R1	Segment number ie byte 1 of the returned sense data
R1	Byte 2 of the returned sense data. Bit 7 is Filemark. Bit 6 is EOM. Bit 5 is ILI. Bit 4 is reserved. Bits 3 to 0 hold the sense key.
R2	The information bytes returned as bytes 3 to 6 of the sense data, (byte sex reversed to suit ARM).
R3	Indicates amount of data transferred.
R4	Amount of data not transferred.

NOTE: bit 23 of R1 always indicates data in R2 is valid, as it is either the valid bit returned by the request sense command, or zero.

*Devices

Displays information about the devices attached to the SCSI bus.

Syntax

*Devices

Parameters

None

Example

The command displays information in the following style:

Device	Type	Capacity	Vendor	Product	Revision
0	Direct-access	100 Mbytes	CONNOR	Cp3100-100mb-3.5	0.0C
1	Read-only	500 Mbytes	HITACH	CDR	0004
2					
3					
4					
5					
6					
7	Host		ACORN	SCSI expansion	0000

The device types allowed are:

- direct-access
- sequential-access
- printer
- processor
- WORM
- read-only
- reserved
- LUN not present
- unknown.

Specifications

Hardware performance

The expansion card is based on the ANSI X3.131 – 1986 SCSI standard and the document Common Command Set (CCS) Revision 4B, relating to direct access devices.

- Up to seven SCSI devices can be supported on the bus.
- Up to eight SCSI devices can be supported by each SCSI device.
- The expansion card supports reconnect and bus arbitration as defined in the SCSI specification. It is the only initiator on the SCSI bus.
- More than one SCSI expansion card can be installed at the same time. Each SCSI expansion card controlling a different SCSI bus.
- The expansion card is capable of a data transfer rates in the order of 1.25MB/sec.

Software performance

- Asynchronous performance is supported using groups 0, 1 and 5.
- Dynamic reconfiguration of the SCSI bus is supported.
- With RISC OS, Direct access devices are directly supported by SCSIFS. Printer, Processor, Write once read multiple (WORM), Sequential access, Read only direct access and reserved classes are not directly supported.
- With RISC iX Direct access and Sequential access devices are directly supported. Printer, Processor, Write once read multiple (WORM), Read only direct access and reserved classes are not directly supported.
- A generic interface is provided for devices not directly supported.
- Drives conforming to SCSI document CCS Rev 4B are supported
- Under RISC iX, performance has been optimised to provide efficient data transfer between the internal hard disc or external SCSI disc and a SCSI streaming tape drive.

Interface specifications

Connectors

On the back panel there is a 50-way standard SCSI socket.
On the main PCB there is a 50-way plug (for connecting an internal drive)

Output levels

Low state (assertion)	0V to 0.4V 48mA at 0.5V sink capability
High state (negation)	2.5V to 5.25V

Input levels

Low state (True)	0V to 0.8V 0.4mA at 0.4V load
High state (False)	2.0V to 5.25V
Hysteresis	0.2V minimum

Termination (internal)

220 ohm to +5V
330 ohm to Ground

Termination external (TERMPWR)

Pin 26 provides +5V (nominal) for external termination. This supply is protected via a diode and fuse.

Standard backplane connector interface

64-way DIN 41612 plug. Rows A and C are used.

SCSI connector pin connections

Pin	Function
26	DB(0)
27	DB(1)
28	DB(2)
29	DB(3)
30	DB(4)
31	DB(5)
32	DB(6)
33	DB(7)
34	DB(P)
35	Ground
36	Ground
37	Ground
38	Terminator power
39	Ground
40	Ground
41	ATN
42	Ground
43	BSY
44	ACK
45	RST
46	MSG
47	SEL
48	C/D
49	REQ
50	I/O

Pins 1 to 25 are all connected to ground (0 volts), except pin 13 which is left open.

PCB link settings

There are various link options on the SCSI expansion card, and these are listed below:

EPROM size selection						
EPROM	LK1	LK2	LK3	LK4	LK5	LK7
27128	C	O	O	O	C	C
27256	O	O	O	C	C	C
27512	O	C	O	C	O	C
27C101	O	C	C	C	O	O
	O - open		C - closed			

The default setting is for a 27256 EPROM.

LK8 and LK9 allow larger SRAM devices to be fitted.

LK10 and LK11 switch the reset line for initiator or target mode:

Mode	LK10	LK11
Initiator	O	C
Target	C	O

The SCSIDM program

*SCSIDM

Configure a Direct Access SCSI device

Syntax

```
scsidm [ device ] [ command [ argument...]]
```

Description

SCSIDM is used by the system administrator to configure a Direct Access SCSI device. SCSIDM may be used to:

- format a device
- verify a device
- create and manipulate partition tables
- associate textual names with RISC iX partitions
- define and display a list of defective data blocks
- run a set of diagnostic routines
- read/write mode sense/select data.

Without any arguments, SCSIDM will prompt for commands from the standard input. If arguments are supplied, SCSIDM interprets the first argument as a device, the second as a command, and all remaining arguments as parameters to the command. The standard input may be redirected causing SCSIDM to read commands from a file.

If SCSIDM prompts for a command, any necessary parameters may be passed with the command, if they are not, SCSIDM will prompt for them. Whenever SCSIDM prompts for input, entering `help` or `?` should produce a few lines of context sensitive help, after which SCSIDM re-prompts for the input. Commands may be abbreviated; overleaf is the list of recognised commands:

Parameters

quit Leave SCSIDM.

? [*command ...*]

help [*command ...*]

Print a short description of each *command* specified in the argument list, or, if no arguments are given, a list of the recognised commands.

device [*devname*]

Select *devname* as the current device, or, if no argument is given, print the current device.

diagnose

Ask the current device to run a set of internal diagnostics. The results can be no more sophisticated than either 'pass', or 'fail' with an appropriate error code returned by the device.

format [*defects* [*interleave*]]

Format the current device using the given *interleave*. *Defects* is a yes/no boolean which indicates whether the current grown list of defects (GLIST) is to be preserved. *Format* will ask for confirmation before the command is started, and this has to be given in response to a prompt. It cannot be passed as an argument.

mapdefects [*fname* [*truncate*]]

Add the list of (hexadecimal) bad block numbers in *fname* to the GLIST. *Truncate* is a yes/no boolean signalling whether or not to truncate the list of defects in *fname*; this is usually a good idea, as the blocks are no longer defective once they have been mapped out.

namepart [*ptlist* [*text ...*]]

Define new text strings for the partitions named in *ptlist*.

section [*dual* [*asize*]]

Section the current device into RISC OS and RISC iX areas. If the boolean *dual* is 'no', then the entire disc is used for RISC OS without further ado. If, however, *dual* is 'yes', then *asize* is rounded up to the nearest number of whole cylinders (a historical restriction imposed by RISC OS), and the rest of the disc space is available for use by RISC iX.

partition [*ptlist* [*sizes*]]

Set up RISC iX partitions within the RISC iX section of the disc. *Ptlist* specifies which partitions are to be defined, and *sizes* is a list of start block and length for each partition in *ptlist*. Note that the start of

a partition is measured from block 0 on the disc, not the first block of the RISC iX area.

prdefects

Print the GLIST for the current device.

prptable

Print the RISC iX partition table, and a map of the disc space occupied by all defined partitions.

scanptable

Use when the partition table gets corrupted. This command reads **every** block on the disc, looking for spare copies of the partition table which are written by the *partition* command, separate tables being identified by a timestamp. When the scan is complete, *scanptable* lists all blocks which look like a partition table (ie have the correct magic number). It then provides the capability to inspect these tables in greater detail, and then choose one as the replacement for the lost or corrupt table.

sensemode [*pages* [*pagno* [*rtype*]]]

Provide an interface to the SCSI mode sense command. *Pages* is a yes/no boolean signalling whether or not to work with the CCS–4.B standard for page formats: stating 'no' restricts the command to those bytes defined in the mode sense parameter list. If *pages* is 'yes', then *pagno* gives the page number, while *rtype* can be one of the following:

0 – report current values
1 – report changeable values
2 – report default values
3 – report saved values.

selectmode [*pages* [*pagno* [*dosave* [*blist* *data*]]]]

Provide an interface to the SCSI mode select command. *Pages* is a yes/no boolean signalling whether or not to work with the CCS 4B standard for page formats: stating 'no' restricts the command to those bytes defined in the mode select parameter list. If *pages* is 'yes', then *pagno* gives the page number; *dosave* is a yes/no boolean flagging whether to save the new data, or simply update the current values (a distinction defined by SCSI); *blist* is a comma-separated list of bytes within *pagno* to be changed; and *data* is a comma-separated list of hexadecimal values for the bytes in *blist*.

verify [*niter*[*automap*]]

Verify the current device. *Niter* is the number of iterations for the main verify loop. If the boolean *automap* is 'yes', then any defects detected will automatically be written to GLIST at the end of the current iteration. If *automap* is 'no', then *verify* will report all bad block found, but do nothing about them.

Bugs

The maximum interleave allowed on a format is one less than the number of tracks per sector on the drive. If mode select is used to change the block size on a device, this also affects the number of tracks per sector, but neither figure changes until the format takes place. So an interleave that was acceptable to the format command may be rejected by the device as an 'illegal request: illegal field in command data block'.

The SCSI device checklist

This section contains the SCSI device list, you should complete the list with details of each of the SCSI devices on your system. You will then be able to look up the configuration of your SCSI devices quickly and easily.

- **SCSI device type and name**: This column lists device types. Complete this column by putting in the names of your SCSI devices.
- **SCSI id number**: This column suggests which SCSI id numbers should be given to which device types. Write in the id numbers you have assigned.
- **Device terminated?**: Only the first and last devices in the SCSI bus should be terminated. Usually the SCSI expansion card is terminated and set up at one end of the bus. Write down which other SCSI device is terminated (this should be the device at the other end of the SCSI bus).

SCSI device type	**SCSI ID number** (suggested)	**Device terminated?** (yes/no)	**RISC OS drive number**
SCSI expansion card	id 7 (default)	Default terminated	——
Disc drive #1	1st SCSI device is id 0		SCSI::4
Disc drive #2	id 1		SCSI::5
Disc drive #3	id 2		SCSI::6
Disc drive #4	id 3		SCSI::7
Disc drive #5 or streamer #2	id 4		
Streamer #1	id 5		
Processor or printer	id 6		

SCSI device checklist

End-user licence conditions

1. Definitions

The following expressions have the meanings given here:

'Acorn' means Acorn Computers Limited, being either owner of all intellectual property rights in the Software, or having the right to grant licences of the Software.

'Developer' means any third party software developer who retains copyright in the Software.

'Documentation' means the printed user documentation supplied with the Software inside the pack.

'Software' means the programs contained in object-code form on the disc(s) supplied with these conditions:

2. Licence

Acorn grants you a personal non-transferable non-exclusive licence (or sub-licence), as follows:

(1) You may copy the Software for backup purposes, to support its use on one stand-alone Acorn computer system. (Separate provision for site licences is made on form APP157 available from your Acorn Authorised Dealer.)

(2) You must ensure that the copyright notices contained in the Software are reproduced and included in any copy of the Software.

(3) You may not:

(i) copy only part of the Software; or

(ii) make the Software or the Documentation available to any third party by way of gift or loan or hire;

(iii) incorporate any part of the Software into other programs developed or used by you; or

(iv) copy the Documentation.

3. Term

This licence remains in effect unless you terminate it:

(1) by destroying the Software and all copies, and the Documentation, or

(2) by failing to comply with the Conditions.

4. Limited warranty and disclaimer of liability

(1) Acorn warrants the disc(s) upon which the Software is supplied to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of purchase, as evidenced by a copy of your receipt. Your Acorn Authorised Dealer will replace a defective disc if returned within ninety (90) days of purchase.

(2) The Software is supplied 'as is'; neither Acorn nor the Developer makes any warranty, whether express or implied, as to the merchantability of the Software or its fitness for any particular purpose.

(3) In no circumstances will Acorn be liable for any damage, loss of profits, goodwill or for any indirect or consequential loss arising out of your use of the Software, or inability to use the Software, even if Acorn has been advised of the possibility of such loss.

5. General

These conditions supersede any prior agreement, oral or written, between you and Acorn relating to the Software.

Index

E

F

H

I

L

N

O

P

R

S

T

U

V

Reader's Comment Form

Acorn SCSI Expansion Card User Guide, Issue 2

We would greatly appreciate your comments about this Guide, which will be taken into account for the next issue:

Did you find the information you wanted?

Do you like the way the information is presented?

General comments:

If there is not enough room for your comments, please continue overleaf.

How would you classify your experience with computers?

☐ **First-time user** ☐ **Used computers before** ☐ **Experienced user** ☐ **Programmer**

Cut out (or photocopy) and post to:

Dept RC, Technical Publications
Acorn Computers Limited
645 Newmarket Road
Cambridge CB5 8PB.

Your name and address:

This information will only be used to get in touch with you in case we wish to explore your comments further.


Acorn