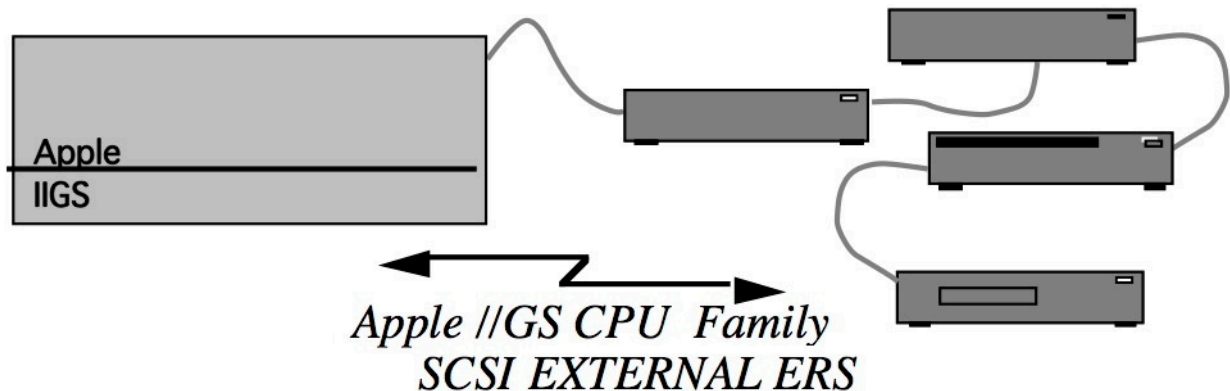


Apple IIgs® GS/OS®  
SCSI Driver  
External ERS  
Ver. 6.1 SCSI-2



Written by: Matt Gulick

© Copyright by Apple Computer, Inc, 1988-1989  
All Rights Reserved

© 2018, SCSI-2 audio commands by Brutal Deluxe Software  
<http://www.brutaldeluxe.fr/>



<b>Preamble.....</b>	<b>6</b>
<b>About This Document.....</b>	<b>6</b>
<b>Physical vs. Logical Devices.....</b>	<b>7</b>
<b>Loading the SCSI Driver .....</b>	<b>8</b>
<b>Physical Structure of an SCSI Driver .....</b>	<b>8</b>
About the Driver Header .....	8
About the Configuration Scripts.....	9
About the Configuration Parameter List.....	9
About the DIB .....	9
<b>Logical Structure of an SCSI Driver .....</b>	<b>11</b>
<b>General Function of the Driver .....</b>	<b>12</b>
<b>Driver Calls.....</b>	<b>12</b>
<b>Caching Data Blocks .....</b>	<b>15</b>
<b>Sparing.....</b>	<b>16</b>
<b>How Device Drivers Get Called .....</b>	<b>17</b>
<b>Driver Startup Call .....</b>	<b>18</b>
Details of the Call. ....	19
What the driver does.....	21
<b>Building the DIBs field by field.....</b>	<b>22</b>
<b>DIB Extension Data .....</b>	<b>24</b>
<b>Building the DIBs in three short SCSI Calls.....</b>	<b>26</b>
<b>Driver Open Call .....</b>	<b>31</b>
<b>Driver Read Call.....</b>	<b>32</b>
<b>Driver Write Call.....</b>	<b>34</b>
<b>Driver Close Call .....</b>	<b>36</b>
<b>Driver Status Call.....</b>	<b>37</b>
\$0000 - Return Device Status .....	42
\$0001 - Return Configuration Parameters .....	43
\$0002 - Return Wait/No Wait Status.....	43
\$0003 - Return Format Options .....	43
\$0004 - Return Partition Map .....	45
\$0005 - Return Last Command Result.....	46
<b>Device Specific Status Calls .....</b>	<b>49</b>
\$8000 - Test Unit Ready; ( All Devices ) .....	54

\$8003 - Request Sense;( All Devices ) .....	55
\$8005 - Read Block Limits;( Sequential Access ) .....	56
\$8006 - Receive QIC-100 System Data;(Apple Tape Drive).....	57
\$8008 - Read; ( Direct Access ) .....	58
\$8008 - Read; ( Sequential Access ) .....	59
\$8008 - Receive; (Processor Devices).....	60
\$8008 - Get Message; (Communication Devices).....	61
\$800D - Read SCSI Defect Data;(Apple Tape Drive).....	62
\$800E - Read Controller Information;(Apple Tape Drive).....	63
\$800F - Read Reverse; ( Sequential Access ).....	64
\$8011 - Read Drive Lines;(Apple Tape Drive).....	65
\$8012 - Inquiry; ( All Devices ).....	66
Additional notes on the INQUIRY command.....	68
\$8013 - Read QIC-100 Information;(Apple Tape Drive) .....	74
\$8014 - Recover Buffered Data;( Sequential Access ) .....	75
\$8014 - Recover Buffered Data;( Printer Devices ) .....	76
\$8019 - Read QIC-100 Defect Data;(Apple Tape Drive) .....	77
\$801A - Mode Sense; ( All Devices ) .....	78
\$801C - Receive Diagnostic Results;( All Devices ) .....	80
\$801F - Read Log; ( Sequential Access Devices ) .....	81
\$8025 - Read Capacity; ( Direct Access ) .....	82
\$8025 - Get Window Parameters; ( Scanners ) .....	83
\$8028 - Read (Extended); ( Direct Access Devices ) .....	87
\$8028 - Read (Extended); ( Scanner Devices ) .....	88
\$802D - Read Update Block; ( Optical Memory ).....	89
\$8034 - Read Position; ( Sequential Access Devices ) .....	90
\$8034 - Get Data Status; ( Scanner Devices ) .....	93
\$8037 - Read Defect Data; ( Direct Access Devices ) .....	95
\$8038 - Read Element Status; ( Changer Devices ).....	96
\$803C - Read Buffer; ( All Devices ) .....	97
\$803E - Read Long; ( Direct Access Devices ) .....	98
\$8042 - Read Sub Channel; ( New for SCSI-2 CD-ROM drives ) .....	99
<i>Sub-Q channel data format</i> .....	100
<i>CD-ROM current position data format</i> .....	104
<i>Media catalogue number data format</i> .....	105
<i>Track international standard recording code data format</i> .....	105
\$8043 - Read TOC; ( New for SCSI-2 CD-ROM drives ).....	107
\$8044 - Read Header; ( New for SCSI-2 CD-ROM drives ) .....	110
\$805A - Mode Sense; ( All Devices ) .....	112
\$805F - Read Log; ( Sequential Access Devices ).....	113
\$80A8 - Read; (Optical Media).....	114
\$80AD - Read Update Block; (Optical Media) .....	115
\$80B7 - Read Defect Data; (Optical Media) .....	116
\$80C1 - Read TOC; (Ruby Drive) .....	117
\$80C2 - Read Q Subcode; (Ruby Drive) .....	118
\$80C3 - Read Header; (Ruby Drive).....	119
\$80CC - Audio Status; (Ruby Drive) .....	120

\$81C2 - Audio Status; (Ruby Drive) .....	121
\$80C3 - Read Header; (Ruby Drive) .....	122

### **Control Call ..... 123**

\$0000 - Reset Device .....	129
\$0001 - Format Device .....	129
\$0002 - Eject .....	130
\$0003 - Set Configuration Parameters .....	130
\$0004 - Wait/No Wait Mode .....	130
\$0005 - Set Format Options .....	131
\$0006 - Assign Partition Owner .....	131
\$0007 - Arm Signal .....	132
\$0008 - Disarm Signal .....	132
\$0009 - Set Partition Map .....	133

### **Device Specific Control Calls ..... 134**

\$8001 - ReZero Unit; ( Direct Access Devices ) .....	135
\$8001 - Rewind Unit; ( Sequential Access Devices ) .....	136
\$8002 - Down Load Code; ( Apple LaserWriter SC ) .....	137
\$8004 - Format Unit; ( Direct Access Devices ) .....	138
\$8004 - Format Unit; ( Printer devices ) .....	139
\$8005 - Send QIC-100 System Data; ( Apple Tape Drive ) .....	140
\$8005 - Draw Bits; ( Apple LaserWriter SC ) .....	141
\$8006 - Clear Bits; ( Apple LaserWriter SC ) .....	142
\$8007 - Reassign Blocks; ( Direct Access Devices ) .....	144
\$8009 - Verify Unit; ( Apple Tape Drive ) .....	146
\$800A - Write; ( Block Devices ) .....	147
\$800A - Write; ( Sequential Devices ) .....	148
\$800A - Print; ( Printer Device ) .....	149
\$800A - Send; ( Processor Devices ) .....	150
\$800A - Send Message;( Communication Devices ) .....	151
\$800B - Seek; ( Block Devices ) .....	152
\$800B - Track Select; ( Sequential Access Devices ) .....	153
\$800B - Slew and Print;( Printers ) .....	154
\$800F - Write Controller Information; ( Apple Tape Drive ) .....	155
\$8010 - Write File Marks; ( Sequential Access Devices ) .....	156
\$8010 - Flush Buffer; ( Printers ) .....	157
\$8010 - Drive Pass-Thru; ( Apple Tape Drive ) .....	158
\$8011 - Space; ( Sequential Access ) .....	159
\$8013 - Verify; ( Sequential Access ) .....	160
\$8014 - Write QIC-100 Information; ( Apple Tape Drive ) .....	161
\$8015 - Mode Select; ( Direct Access Devices ) .....	162
\$8015 - Mode Select; ( Changer Devices ) .....	163
\$8015 - Mode Select; ( Communications Devices ) .....	164
\$8016 - Reserve Unit; ( Direct Access Devices ) .....	165
\$8016 - Reserve Unit; ( Sequential Access Devices ) .....	166
\$8016 - Reserve Unit; ( Apple Tape Drive ) .....	167
\$8016 - Reserve Unit; ( Changer Devices ) .....	168

\$8017 - Release Unit; ( Direct Access Devices ) .....	169
\$8017 - Release Unit; ( Sequential Access Devices ) .....	170
\$8017 - Release Unit; ( Changer Devices ) .....	171
\$8019 - Erase; ( Sequential Access Devices ) .....	172
\$801B - Start/Stop Unit; ( Direct Access Devices ) .....	173
\$801B - Load/Unload; (Sequential Access Devices) .....	174
\$801B - Stop Print; (Printers) .....	175
\$801B - Scan; ( Scanner Devices ) .....	176
\$801D - Send Diagnostic; ( All Devices ) .....	177
\$801E - Prevent/Allow Removal; ( Direct Access ) .....	178
\$8024 - Define Window Parameters; ( Scanners ) .....	179
\$802A - Write (Extended); ( Direct Access Devices ) .....	183
\$802A - Send (Extended); ( Scanner Devices ) .....	184
\$802A - Write (Extended); ( Optical Memory Devices ) .....	185
\$802B - Seek (Extended); ( Direct Access Devices ) .....	186
\$802B - Locate (Extended); (Sequential Access Devices) .....	187
\$802C - Erase; ( Optical Memory ) .....	188
\$802C - Read Generation; ( Optical Memory ) .....	189
\$802E - Write and Verify; ( Direct Access Devices ) .....	190
\$802E - Write and Verify; ( Optical Memory Devices ) .....	191
\$802F - Verify; ( Direct Access Devices ) .....	192
\$802F - Verify; ( Optical Memory Devices ) .....	193
\$8031 - Medium Position; ( Scanner Devices ) .....	194
\$8033 - Set Limits; ( Direct Access Devices ) .....	195
\$8034 - Pre Fetch; ( Direct Access Devices ) .....	196
\$8035 - Synchronize Cache; ( Direct Access Devices ) .....	197
\$8036 - Lock/Unlock Cache; ( Direct Access Devices ) .....	198
\$8038 - Media Scan; ( Optical Memory Devices ) .....	199
\$803B - Write Buffer; .....	201
\$803D - Update Block; ( Optical Memory Devices ) .....	202
\$803F - Write Long; ( Direct Access Devices ) .....	203
\$8045 - Play Audio(10); ( New for SCSI-2 CD-ROM drives ) .....	204
\$8047 - Play Audio MSF; ( New for SCSI-2 CD-ROM drives ) .....	206
\$8048 - Play Audio Track Index(10); ( New for SCSI-2 CD-ROM drives ) .....	208
\$8049 - Play Track Relative(10); ( New for SCSI-2 CD-ROM drives ) .....	210
\$804B - Pause/Resume; ( New for SCSI-2 CD-ROM drives ) .....	212
\$8055 - Mode Select; ( All Devices ) .....	214
\$80A5 - Move Medium;( Changer Devices ) .....	215
\$80A5 - Play Audio(12); ( New for SCSI-2 CD-ROM drives ) .....	216
\$80A6 - Exchange Medium; ( Changer Devices ) .....	217
\$809A - Play Audio Track Relative(12); ( New for SCSI-2 CD-ROM drives ) .....	218
\$80AA - Write; ( Optical Memory ) .....	220
\$80AC - Erase; ( Optical Memory ) .....	221
\$80AE - Write and Verify; ( Optical Memory Devices ) .....	222
\$80AF - Verify; ( Optical Memory Devices ) .....	223
\$80B3 - Set Limits; ( Direct Access Devices ) .....	224
\$80BD - Update Block; ( Optical Memory Devices ) .....	225

\$80C0 - Eject Disk; ( Ruby Drive ) .....	226
\$80C8 - Audio Track Search; ( Ruby Drive ) .....	227
\$80C9 - Audio Play; (Ruby Drive) .....	228
\$80CA - Audio Pause; (Ruby Drive).....	229
\$80CB - Audio Stop; (Ruby Drive).....	230
\$80CD - Audio Scan; (Ruby Drive) .....	231
\$80CE - Audio Control; (Ruby Drive).....	232
<b>Driver Flush .....</b>	<b>233</b>
<b>Driver Shutdown .....</b>	<b>234</b>
<b>Communicating with the Device .....</b>	<b>234</b>
<b>Addendum to SCSI Driver .....</b>	<b>234</b>
Overview.....	234
Description .....	234
Calls.....	235
<b>Configuration Status Parameters.....</b>	<b>235</b>
Disk Calls .....	235
Partition Calls .....	236
<b>Device Driver Error Codes.....</b>	<b>238</b>
<b>Miscellaneous CD-ROM tables.....</b>	<b>239</b>
Table 264 - CD-ROM medium type codes.....	239
Table 265 - CD-ROM device-specific parameter .....	239
Table 266 - CD-ROM density codes.....	240
<b>CD-ROM Parameters Page .....</b>	<b>241</b>
<b>Extracting digital audio through the SCSI-2 bus.....</b>	<b>244</b>
\$D8 – Read CD-DA (for CD-Audio only) .....	245
\$D9 – Read CD-DA MSF (for CD-Audio only) .....	246

## **Preamble**

This SCSI-2 ERS includes the following Apple Computer, Inc. documents :

- CDSC+ Mod to SCSI Driver ERS
- Addendum to SCSI Driver ERS version 2.00

Other references :

- SCSI Manager ERS 6.0 by Apple Computer, Inc.
- SCSI-2 specifications, <http://www.staff.uni-mainz.de/tacke/scsi/SCSI2.html>
- Digital audio on SCSI bus, various sources

This new version of the ERS and GS/OS driver contain the standard SCSI-2 CD-ROM audio commands. Get the latest version of the software by visiting <http://www.brutaldeluxe.fr/>

Antoine Vignau

## **About This Document**

This document explores the task of writing an SCSI (Small Computer Systems Interface) Driver for GS/OS following the block diagram in **Figure 1**. It is assumed that the reader is somewhat familiar with the ANSI® x3.131-1986 and SCSI-2 document and SCSI in general. It is also recommended that the reader have available the most recent revision of the SCSI-2 draft or final spec. This document is necessary because of the devices that have been added to SCSI since the release of the SCSI-1 Spec.



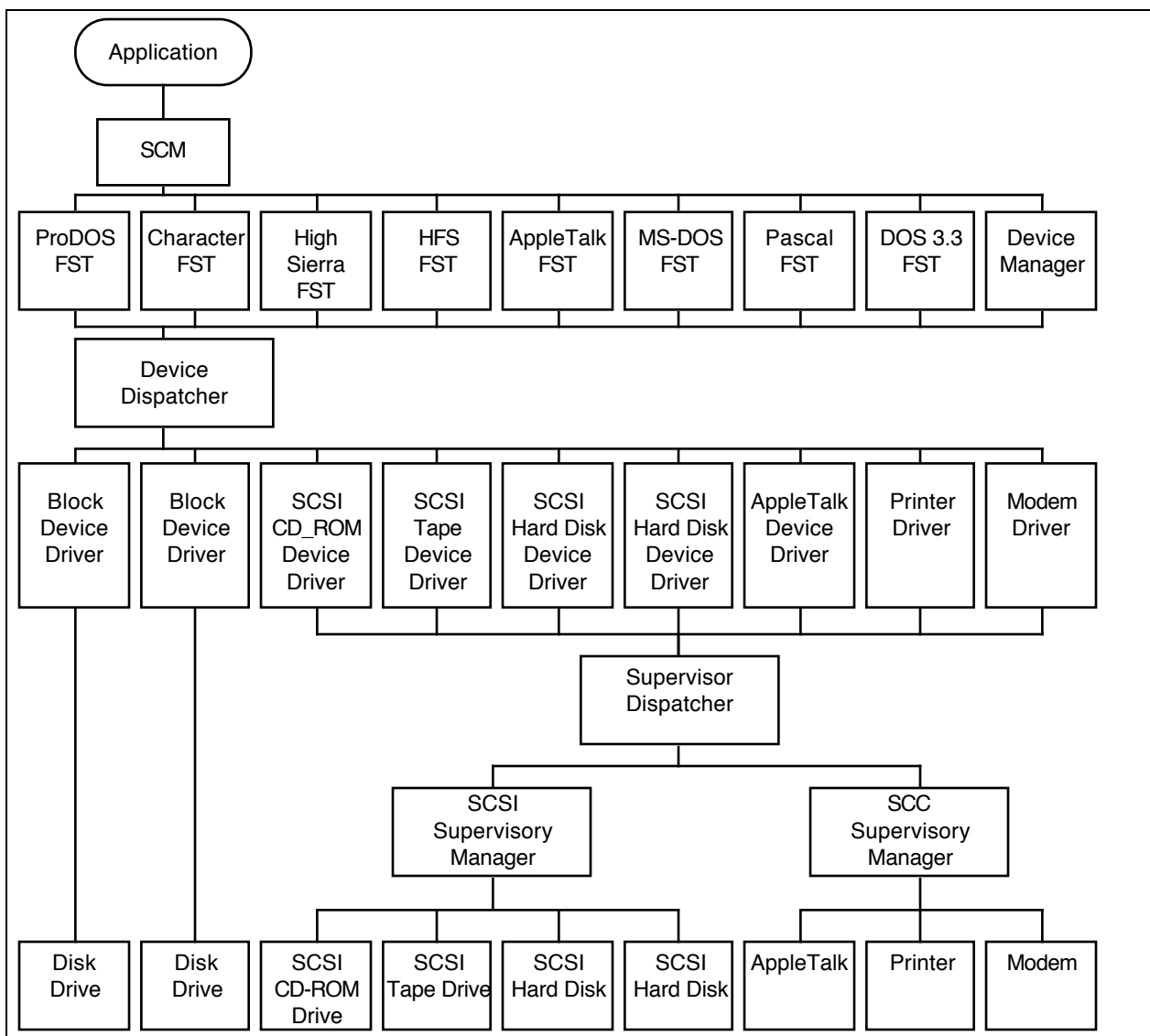


Figure 1

### Physical vs. Logical Devices

In the world of SCSI, the actual box containing the hardware and connectors is considered to be an SCSI Device. Each SCSI Device can consist of one or more Logical Units. An example would be a 40 Mb fixed hard disk with a built in 5 Mb removable hard disk cartridge for backup purposes. These Logical Units can consist of Block and Character Devices in any combination. Because of the way the Finder will represent these devices, they are treated as separate and individual *Physical Devices* in this document. Block Devices can further be partitioned into separate volumes within that device. These are called *Logical Devices* in this document.

## **Loading the SCSI Driver**

The SCSI Driver is a loaded device driver and as such must adhere to a few rules.

As stated in the **GS/OS Device Driver ERS** the SCSI Driver is compacted to object module format type 2.

It has a file type of \$BB.

In addition, the AuxType for the SCSI Driver is \$000001xx. This indicates that the driver is active, that it is indeed a driver, and that at load time there are xx devices supported. The 'xx' represents a value in the range of \$01 - \$3F. This is the number of devices that will be started up at the time that the driver is loaded. This number depends on the group of devices targeted by the driver being loaded. If the device is a character device, this number will be small (\$01, \$02 or greater). If the targeted devices are block devices, there is a possibility that \$3F devices can exist on the system. A worst case example is 8 slots \* 7 SCSI Devices \* 8 Logical Units \* 63 partitions on each of them. This result is \$6E40 Logical Devices. Unrealistic not to mention insane but still possible. At load time the driver can build DIBs for as many of the devices that are online as it wants, but only the first \$3F will be started by the Device Dispatcher. It is then up to the driver to ensure that any remaining devices are also started at a later time using the Post Driver Install call. A method for this is discussed in the Driver Startup Call discussion.

## **Physical Structure of an SCSI Driver**

An SCSI driver consists of a driver header, configuration script, configuration parameter list, Device Information Block(s) (DIB) and the driver code segment. If the device driver supports more than one device type, a configuration script, configuration parameter list and DIB must be provided for each device. Each device may have it's own individual code segment or a common code segment may be shared by all devices supported by the driver.

## **About the Driver Header**

The header is used when loading the driver. It indicates where the configuration parameter lists and DIBs are located. The device dispatcher loads only the driver, DIBs and configuration parameter lists using an initial load call to the system loader. The header contains the following information:

Word	Offset to 1st DIB
Word	Count of number of devices
Word	Offset to 1st configuration parameter list for device #1
Word	Offset to 1st configuration parameter list for device #2
Word	Offset to 1st configuration parameter list for device #3
Word	Offset to 1st configuration parameter list for device #4
etc.	

### **About the Configuration Scripts**

The structure of the configuration scripts is not completely defined at this time. The current drivers define 2 long words of zero as the Configuration Parameter Lists.

Please refer to the **GS/OS Device Driver ERS** for a discussion of this information.

### **About the Configuration Parameter List**

The Configuration Parameter List contains device dependent information used in configuring a specific device. The configuration parameter list may be pre-configured through the use of the configuration program. Additionally, the configuration parameter list may be examined or modified by the status and control calls supported by the device driver.

In addition to the configuration parameter list a default configuration parameter list should be located contiguous to each configuration parameter list.

Please refer to the **GS/OS Device Driver ERS** for a discussion of this information.

### **About the DIB**

The DIB must contain the following information:

Link Pointer.	
Entry Pointer.	
Device Characteristics.	
Block Count.	<i>It should be noted that this parameter may be dynamic if the device supports assorted types of removable</i>

*media or partitioned removable media. In this case any status, or control call that detects online and disk switched should update the block count in the DIB after media insertion.*

Device Name. *The device name must be in upper case with the MSB off.*

Slot Number. *Bits 0 through 2 indicate the slot while bit 3 indicates that the slot is internal or external.*

Unit Number.

Device Version Number.

Device Type ID. *A list of device types and their associated ID numbers is shown below:*

\$0000	Disk ][ (includes Duodisk, Unidisk & Disk//c)
\$0001	Profile 5 meg
\$0002	Profile 10 meg
\$0003	Disk 3.5
\$0004	SCSI (generic)
\$0005	SCSI HD
\$0006	SCSI Tape
\$0007	SCSI CD-ROM
\$0008	SCSI Printer
\$0009	Modem
\$000A	Console
\$000B	Printer
\$000C	Serial LaserWriter
\$000D	AppleTalk LaserWriter
\$000E	Ram Disk
\$000F	Rom Disk
\$0010	File Server
\$0011	IBX
\$0012	AppleDesktop Bus
\$0013	Hard Disk - (generic)
\$0014	Floppy Disk - (generic)
\$0015	Tape Drive - (generic)
\$0016	Character - (generic)
\$0017	MFM Floppy Disk - Super Drive
\$0018	Network (generic - AppleTalk)
\$0019	Sequential access device
\$001A	SCSI Scanner
\$001B	Other Scanner
\$001C	LaserWriter SC
\$001D	AppleTalk Main Driver
\$001E	AppleTalk File Service Driver
\$001F	AppleTalk RPM Driver
\$xxxx	Apple 40 MB Tape Drive
\$xxxx	SCSIoid this an that Driver

**Note:** *The list of device types provided above does not in any way indicate that Apple Computer, Inc. intends to provide devices or device drivers for the types listed. In creating this list, we tried to think of as many unique device types as possible that may need*

*a unique device type ID assignment. Device type ID numbers are assigned by Apple Computer, Inc. If you are developing a driver for a device type that is not listed, you will need to get a device type ID assignment from Apple Computer, Inc.*

Head Device Link.

Forward Device Link.

**Note:** *Both link fields are maintained on a by device basis. These links imply a link between two or more logical devices within a single physical device. A value of zero indicates no link.*

DIB Extension Ptr. *This long word pointer points to where the Extended data of the DIB Begins.*

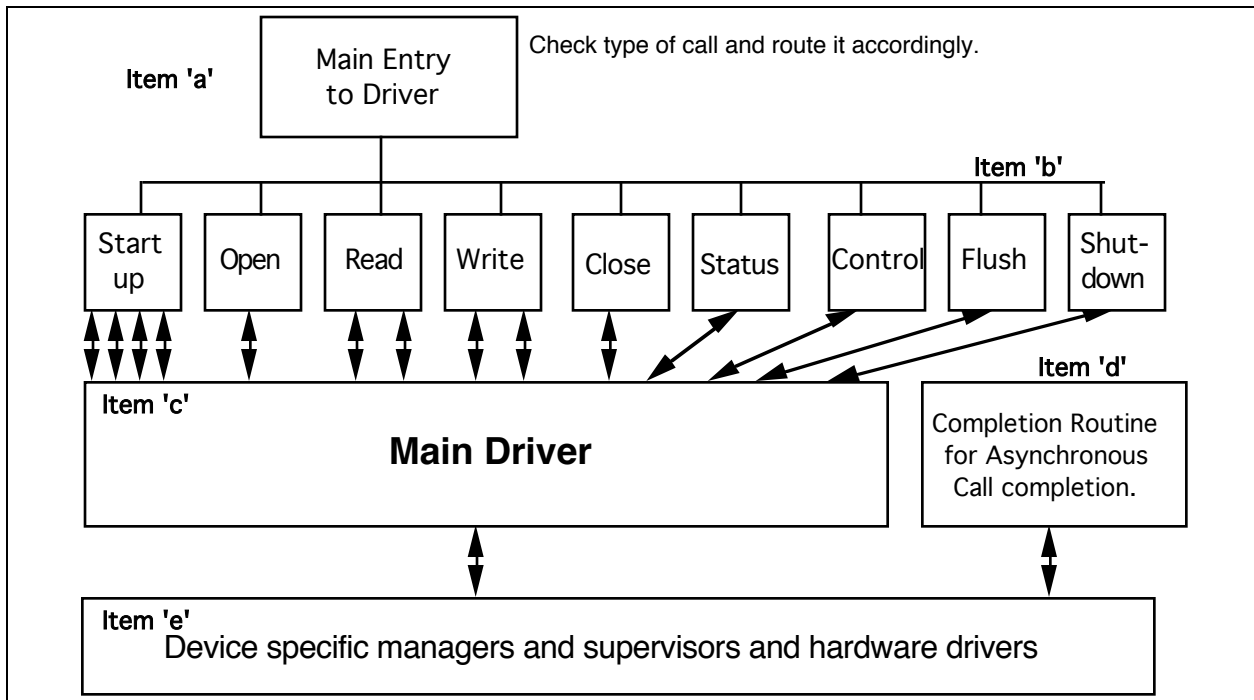
DIB Device # *This is the device number assigned to this DIB by the OS and should never change, even when the device goes offline, until the DIB receives a valid shutdown call.*

These fields are discussed in detail as it applies to an SCSI Driver in the section of this document dealing with the actual building of the DIBs.

## **Logical Structure of an SCSI Driver**

The SCSI Driver has a main entry point (**Item 'a', Figure 2**) that when called decides the type of call being issued and routes the call to the appropriate call filter (**Item 'b', Figure 2**). These filters, depending on the call, gather information from one location and route it to another. This could be information from the Application to the device or vice-verse. The call may also result in the DIB being updated with any changes at that time.

If the filter needs to talk to the device, it issues calls to the SCSI Device sending them via the Main Driver (**Item 'c', Figure 2**) section of the overall device driver. Each call to the main driver results in one or more calls to the Supervisory Dispatcher (**Item 'e', Figure 2**) for routing to the SCSI Supervisory Manager (SCSI Manager) which uses this data to communicate with the target device via the hardware. The drivers that are written correctly will never have carnal knowledge of the hardware environment, and if any hardware changes are made, the drivers will be completely unaffected.



**Figure 2**  
(Internal structure of SCSI Driver)

## General Function of the Driver

The SCSI Driver, like all other drivers, translates calls from an FST via the Device Dispatcher into a format understood by the SCSI Manager. This is the same set of calls that are sent to other drivers and their input/output data structure are as outlined in the picture '**CALLS TO DEVICES**'.

## Driver Calls

All SCSI Drivers accept a standard set of calls. These calls include:

```

DRIVER_STARTUP           (Not an Application Call)
DRIVER_OPEN
DRIVER_READ
DRIVER_WRITE
DRIVER_CLOSE
DRIVER_STATUS
DRIVER_CONTROL
DRIVER_FLUSH             (Not an Application Call)
DRIVER_SHUTDOWN          (Not an Application Call)
    
```

The details of each driver call will be described individually, and are described in detail on the following pages.

## CALLS TO DEVICES

DRV\_STARTUP

In a few cases some of these calls result in nothing more than returning to the caller. An example is the **DRIVER\_OPEN** call to a block device driver. Other calls cause the driver to set internal flags for future reference (ie. altering wait/no-wait mode). Yet other calls result in a single or even several calls being sent on down the line to the target device.

At the completion of the call the DIB may be updated with new information, the requested data are returned from the device if a GET\_INFO, READ or a STATUS call was issued or data are sent on to the device in the case of an SCSI FLUSH, WRITE or a CONTROL call.

It is also up to the driver, in the case of an SCSI Block Device, to be aware of and account for the partition map of a device. It is the responsibility of SCSI Block Device drivers to interpret the information in the partition map if it exists. This is done not only at startup time, but also if a disk switched condition exists or an application issues a Write\_PMap, or Assign Partition calls to the device in question. The later three situations cause the driver to re-build the DIBs belonging to that device. This is discussed in detail later in this document.

The device specific Status and Control calls available to the File System Translators (FSTs) as well as applications have a structure after which all calls to the Main Driver are structured. Even the Startup, Read, and Write calls along with the other types of calls are modified into a single or group of SCSI Status and/or Control calls. These calls use the same control and status codes as defined for the outside world while talking to the driver.

There are of course codes defined that are at first glance in violation of the spirit of the status and control calls. This was necessary for device intelligent applications that need to issue some of the more powerful SCSI supported calls. The read call is translated to a status call and the write to a control call. To allow for this the Command Codes are styled after those defined by the ANSI® X3.131-1986 and SCSI-2 documents. The text below was extracted from the former.

*The operation code (Table 6-1) of the command descriptor block has a group field and a command code field. The three-bit group code field provides for eight groups of command codes. The five-bit command code field provides for thirty-two command codes in each group. Thus, a total of 256 possible operation codes exist. Operation codes are defined in sections 7 through 13.*

- Group 0 - six-byte commands (see Table 6-2)*
- Group 1 - ten-byte commands (see Table 6-3)*
- Group 2 - reserved*
- Group 3 - reserved*
- Group 4 - reserved*
- Group 5 - twelve-byte commands (see Table 6-4)*
- Group 6 - vendor unique*
- Group 7 - vendor unique*

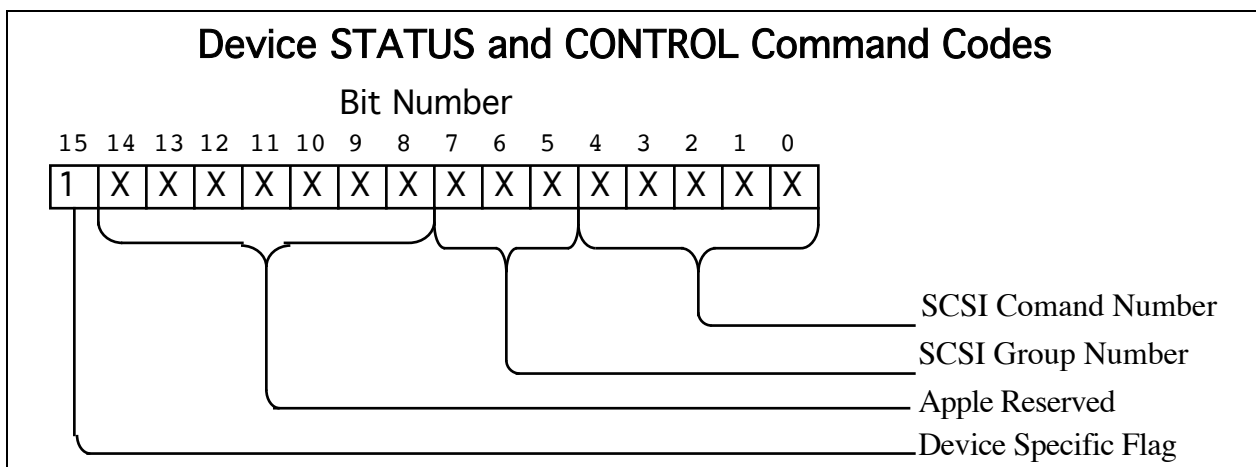
The Driver uses the least significant byte of the control and status codes to be the SCSI command and in the most significant byte we use the high bit to



indicate a device specific call. All other bits are currently reserved by Apple Computer, Inc. Please see **Figure 3** below for a bit by bit example of the Command Code structure.

**It is important that the Application MUST verify the Device ID before issuing any device specific call.**

Because the command codes are structured this way, there are holes in the command code list for the Control Calls as well as the Status Calls. It is entirely possible for the two sets of calls to be put together in a single list without any duplication of command numbers. Because of this, the driver must make the distinction of what qualifies as a Control Call and whether to allow that call to be also issued as a Status Call. This is also true in the reverse situation. It is possible for a command number to be a Status Call to one type of device and a Control Call to another type. Drivers that handle multiple device types will need to distinguish one from the other.



**Figure 3**

### Caching Data Blocks

Caching of blocks from a SCSI Block Driver, as well as any other driver that accepts multi-block calls, can be tricky. A great deal of attention must be paid to what is in the cache, what gets updated, what doesn't and when. Another problem is verifying that each block is 512 bytes long. There are some drives that have been formatted to contain tag bytes with each block making the block 532 bytes long. When a multi block request is received, great attention must be paid to where the data goes. The 513th byte of the request must go to block 2 of the request while 532 bytes (512 data and 20 of zeros) are written to each block. The Apple Driver uses data chaining. Be cautious when updating the cache. This is of the utmost importance.

The Apple SCSI Drivers use the following method of performing cache related reads and writes.

When a read call is issued to the driver, the driver checks the cache to see if the requested blocks are there. If the call is requesting more than 1 block, it starts its check with the first block of the request. If that block is in the cache, it is then moved to the user's buffer at the correct location for block # of the request and block size. It then checks the next block if the current one is cached. If it finds a block that is not cached, the driver then, starting with the first block not in the cache, reads the remaining data from the drive. Once the data is read, the driver then picks up where it left off checking the cache. Every time a block is found that is in the cache, the data read from the drive is updated to contain the data that is in the cache.

The write call is handled slightly different but is also easier. When a write call is received that is not a deferred write, then the entire transaction is written to the disk. Following that, the driver then checks the cache to see if any of the blocks that were just written are also contained there. Every time it finds a block that is cached, the driver updates the cached info with the data that was just written to the device.

Write deferred calls are also fairly simple. The driver, starting with the first block of the request, checks to see if the block is in the cache. If it is there, then the data is updated. If the block is not there, then it is added.

### **Sparing**

Sparing is used to reassigning a block that has a lowered read reliability, all failed writes will be spared, and reads are spared only when the data can be retrieved. The following algorithm should clarify what actually takes place.

A read call is issued to the device specifying a particular block. The device reads the block checking the data for correctness. If the data is damaged, an error is returned, and the call is retried. If this retry is needed more and more before the correct data is returned, then the block is reassigned and the data is written to the new block before returning. If the block is so bad that the data could not be read, then an error is returned to the caller and the block is not spared. (It is possible that a later read will be successful). If the block is damaged, then the next write will cause the block to be spared. This is due to the inherently destructive nature that a write has over the previous data.

## How Device Drivers Get Called

The SCSI Device Driver's main entry point is called by the Device Dispatcher in full native mode (16 bit 'm' & 'x') via a 'JSL' instruction. Prior to calling the SCSI device driver, the device dispatcher sets the transfer count to zero. The processor state prior to device driver dispatching is set as follows:

Accumulator	=	Call Number
Y Register	=	Unspecified
X Register	=	Unspecified
P Register	=	0=m=x=e
Direct Page	=	GS/OS Direct Page
Data Bank	=	Unspecified
Stack Pointer	=	GS/OS stack
System Speed	=	Fast Mode

The data bank is preserved by the Device Dispatcher. A standard parameter block is set up on GS/OS direct page prior to dispatching to the SCSI device driver. Currently, there is no workspace available for a drivers use on GS/OS direct page. Any SCSI Drivers requiring direct page must roll their own either when the driver is started or as a result of an open call. The driver must release this direct page as a result of either a shutdown or close call respective to which call acquired the direct page. Device drivers must not permanently modify any GS/OS direct page location with the exception of the transfer count which indicates the number of bytes processed by the driver.

**DEVICE DRIVERS MUST NEVER ACCESS GS/OS DIRECT PAGE USING ABSOLUTE OR ABSOLUTE LONG ADDRESSING MODES** (Rumor has it we will move the location of GS/OS direct page with every release!!!). To access the GS/OS Direct page the driver saves the direct Page address and retrieves it for use as an index when needed (See code example below).

```

tdc                                ;Get Direct Page address
sta    |scsi_dp
      .
      .
      .
ldx    |scsi_dp
lda    >call_number,x

```

SCSI Drivers must return via an 'RTL' instruction in full native mode (16 bit 'm' & 'x'). The register states on return to the device dispatcher are as follows:

Processor Status	=	Carry set if error, Carry cleared if no error
Accumulator	=	Error code if error, \$0000 if no error
Y Register	=	Unspecified
X Register	=	Unspecified
Direct Page	=	GSOS Direct Page
Data Bank	=	Unspecified
Stack Pointer	=	GSOS stack

### **Driver Startup Call**

Call Parameters :      Device Number      ≠      \$0000  
                         Call Number        =      \$0000  
                         DIB Pointer

Device Number.      *This word parameter specifies the target device. This parameter must be nonzero.*

Call Number.        *This word parameter specifies the type of call. Must be a zero for this call.*

DIB Pointer.        *This longword points to the DIB for the device being accessed. For the first time that call is issued, this points to the DIB structure included with the driver when it was loaded.*

This call is supported by both character and block SCSI drivers and is executed by GSOS during initialization or after loading a driver in preparation for launching an application. This call performs any tasks necessary to prepare the driver for operation. This includes memory allocation. Character device drivers maintain an internal flag indicating when the device is open; therefore, the open flag should be set to not open by this call.

**This call must not be issued either by an Application or an FST!!!**

The device dispatcher sets the DIB pointer on direct page and in the devices DIB it sets the DIB\_DEVICE\_NUMBER prior to issuing a startup call to the device. This parameter is used by devices that support removable partitioned media. Each media partition will be accessed through a separate device driver<sup>2</sup> as if that partition was a separate device. Since multiple devices can share a common media it is necessary to maintain links between these devices in order to manage disk switch and off line conditions that may have to be reflected at each linked device. **!!!!!!The device driver is responsible for maintaining these device links and uses the DIB device number to**

---

<sup>2</sup> Each DIB is considered to have it's own separate driver, but it may actually be that one common driver code segment services more than one DIB.

**initialize the head link and forward link in the DIB!!!!** It should be noted that if the driver is replacing the boot device, the DIB device number will not remain constant. The slot and unit number contained in a DIB is not considered valid by the device dispatcher until the device has successfully returned from the startup call. If the slot and unit in the DIB match the boot device's slot and unit then the driver will be relocated in the device list to device #1. In this case, the DIB device number will be updated to device #1 but only after startup has been completed. In this case the driver can only consider the DIB device number to be correct on the second device access.

### **Details of the Call.**

When GS/OS is started, it finds the SCSI Driver in the DRIVERS sub-directory, loads it and issues a DRIVER\_STARTUP call. The driver startup call does several things at this time to prepare the driver for use in the GS/OS environment. As part of the driver object code is a template for the building of the DIBs needed for the devices that the driver will be in communication with. In **Figure 4** is a layout of the DIB as described in the Device Driver ERS.

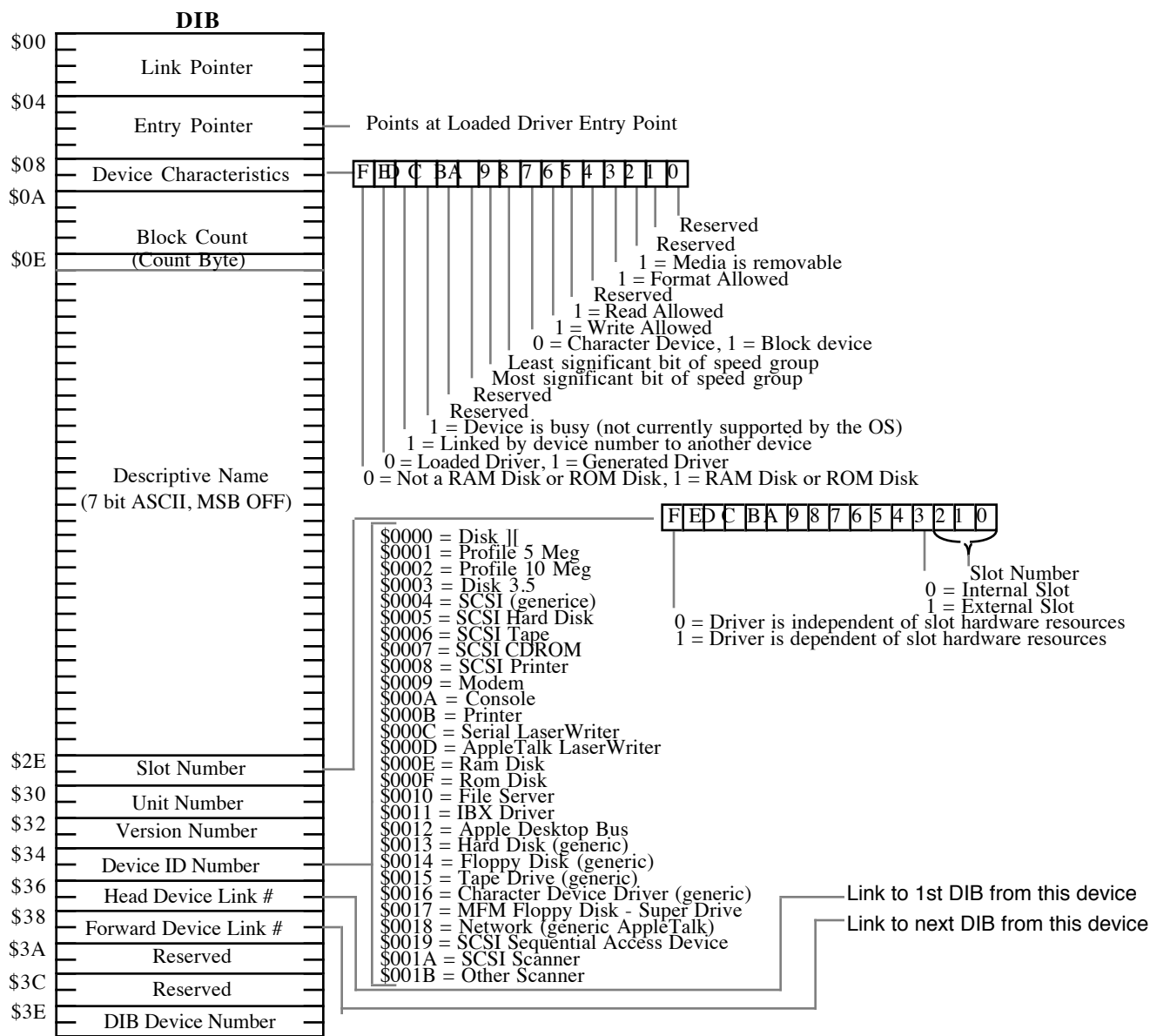


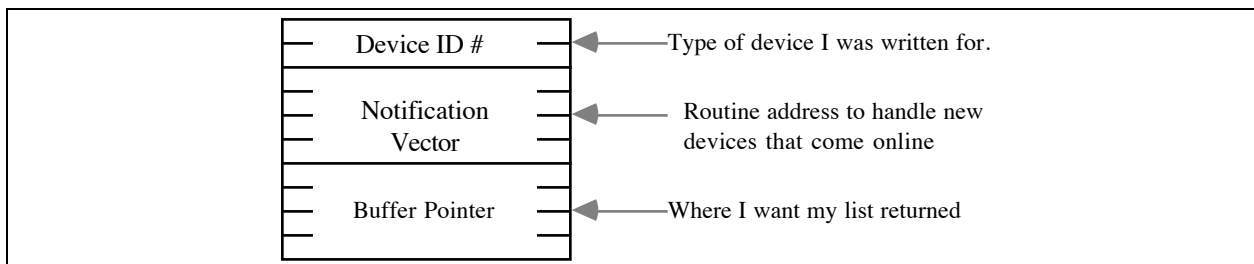
Figure 4

Some of the information in the DIB is standard for all DIBs associated with this driver and are setup as data statements in the DIB template used by the driver. This template could also double as the first DIB if the driver is unsuccessful in finding devices to communicate with. By so doing, it is possible for the driver to remain active rather than being purged if there are no devices active. This way if a device becomes active and we are notified, the driver will establish communication with the device without rebooting.

In addition to the standard fields defined for a DIB are several other fields that aid in the management of the SCSI Device for which the driver was written. These extensions are driver dependent, and just because one driver uses them, not all need to. For the GS/OS Drivers under development at Apple for SCSI Devices these structures will be used as defined. The DIB Extension as defined for this driver are discussed in detail in the section titled '**DIB EXTENSION DATA**'.

### What the driver does.

On startup, the driver will need to get the ID for the SCSI Manager then issue a Request\_Devices call to the SCSI Manager through the Supervisor Dispatcher. The parameter list (**Figure 5**) will be used by the SCSI Manager to build a list that is returned in the buffer specified.



**Figure 5**

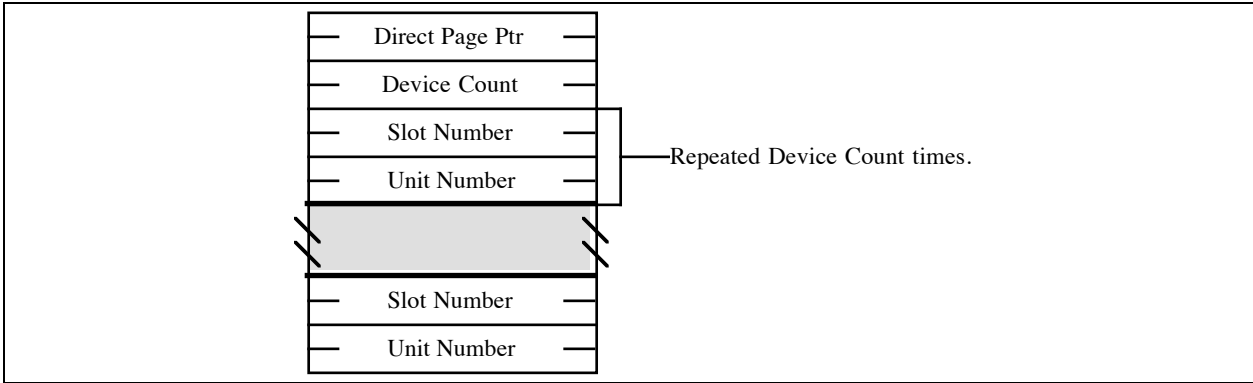
The *Device ID #* is of the same set of IDs as those defined by the SCSI INQUIRY call. This is to tell the SCSI Manager what types of devices we want to identify. The SCSI Manager will return to the driver a list of the currently connected devices of this type. It is from this list that we will build our DIBs.

The next field is the Event Notification Vector. This is the address of the routine that the SCSI Manager should call when a new device comes on line that is of the type specified in the device type field, or any other async event notification.

The last field is the buffer to which the list of devices would be returned.

The return data (**Figure 6**) starts with a word pointer. This is the area that has been set aside for use by the driver as its direct page. It is 256 bytes long and is shared by all SCSI drivers. All data stored here by the driver should be considered to be destroyed upon exiting the driver. The next word is the device count. This is the number of entries contained in the buffer. The list itself is made up of a two word pair for each device returned. The first word is meaningless and is used to fill in the 'SLOT' word in the DIB. The second word is the unit number. This is the value we will use when calling the SCSI Manager. The low 10 bits are reserved for use by the SCSI Manager. The high 6 bits can be used by the driver but must be zero when calling the SCSI Manager. These are used by the driver of block devices to

give partitions a unique unit number. If this were not done, the slot and unit number of all the partitions on a device would be the same and all but one of them would be shut down.



**Figure 6**

The driver now knows how many devices are currently attached to the system. It must be noted that these devices are physical devices as far as the driver is concerned. Even though only 1 device is returned, it may be that there are 10 partitions on it. This will only be possible on block devices and the CD-ROM. The Operating System itself will not be able to distinguish between a physical or a logical device except by checking the linked device bit in the device characteristics.

As each DIB is built it will be necessary for the driver to issue a few SCSI calls to the device to get the needed information. The method of discovery for this information will be discussed in two ways. First will be a field by field description of what is needed and what calls to issue to get the information. Second will be a discussion of the SCSI calls needed and what data requirements they will fulfill. This will hopefully reduce any confusion as to how the information for the DIB is found. Also all pictures will be given in the section detailing the SCSI calls. They will have more meaning there.

### **Building the DIBs field by field.**

**Link Pointer:** This field is used to connect the DIBs for this driver to each other for management purposes. This will point to the next DIB built in sequence and should never change unless the memory where the DIB is located is freed up. This would be done through the memory manager and is not recommended. If a DIB is no longer in use, it should then be marked first as a disk switch then as off line. This Memory is now available to the driver if a DIB space is needed for this physical device only.

**Entry Pointer:** This is nothing more than a pointer to the drivers main entry point.



**Device Characteristics:** The default for this field in binary would be \$00x000110xx0xx00 for a character device and \$00x000111xx0xx00 for a block device where x represents information we do not yet know.

The first x (left to right) is the linked DIB bit. This is used to indicate that this device contained two or more partitions. When one of the DIBs is ejected, and this bit is set, the the DIB will be marked offline and those linked DIBs will remain online. Ejection will only take place when the last online DIB is ejected.

The next x is *Write Allowed*. This is returned by the MODE SENSE command. Bit 7 of byte 2 in the 4 byte header set to 0 indicates that the media is write allowed. See the ANSI® X3.131-1986 SCSI document on page 110.

The next x represents the *Read Allowed* bit. **!!!! As far as I can tell at this time, all SCSI devices are read allowed except maybe an SCSI Printer but then the driver will know who he is talking to. Right? !!!!!**

Next in the lineup is the *Format Allowed* bit. This bit is not as easy as the previous bits but is still readily available to the driver. This is found by first issuing an INQUIRY call then by checking the group 0 call bitmap for bit 3 of byte 0. If this bit is set then the device will accept FORMAT calls.

NOTE: Refer to command bitmap discussion.

Lastly, is the media removable? This is contained in the data returned by the INQUIRY call also. Bit 7 of byte 1 in the 5 byte header if set to 1 indicates that the media is removable.

**Block Count:** This information is taken from the READ CAPACITY Call in the case of block devices. If the block device contained several partitions, then this information will be extracted from the partition map entry for this logical device.

**Descriptive Name:** This field contains the device name that is used by the Operating System. In the case of a hard disk this will be 'APPLESCSI.HD00.00' where the first two 0's following the HD refer to the device number and the two 0's following refer to partitions on that device. For other devices, the names are similar and indicate the type of device. Refer to the following list:

APPLESCSI.HD00.00	Apple Hard Disk and other direct access devices.
SCSI.TAPE00.00	Sequential Access Devices
APPLESCSI.Printer00.00	SCSI Printer Devices
APPLESCSI.Proc00.00	SCSI Processor Devices (ie. other computers, etc.)
APPLESCSI.WORM00.00	Write-Once Read-Many
APPLESCSI.CDROM00.00	CD-ROM Devices
APPLESCSI.Scanner00.00	SCSI Scanners
APPLESCSI.Optical00.00	Optical Memory
APPLESCSI.Changer00.00	Changer Devices (ie. jukebox, etc.)
APPLESCSI.Com00.00	Communication Devices

APPLESCSI.TAPE00.00

Apple's Tape Drive & 3M MCD-40/SC Tape Drive

**Slot Number.** This is returned as part of the data from the Get\_Devices call to the SCSI Manager. It has no meaning to the SCSI Driver.

**Unit Number.** This is also returned by the Get\_Devices call but it does have meaning for us. This two byte value is in reality two fields put together to form the unit number. The least significant 10 bits are the physical device number as it appears to the SCSI Manager. This has no meaning for us except in the case of a device going off-line or if the media is removed. We need some way of identifying all the devices that we have built DIBs for that are affected by this occurrence. This will be discussed later in full detail. The most significant 6 bits represents the logical device or partition on the actual physical device.

**Version Number:** This, as discussed above, is the version of the driver that we have written.

**Device Number:** Also discussed previously, this is the number associated with the type of device our driver was written for.

**Head Link Device Number:** This is the device number used as an entry to the device list where the pointer to the first DIB for this device resides in memory. Each device will have a series of DIBs depending on the number of partitions residing on that disk.

**Forward Link Device Number:** This is the device number used to locate the next DIB that follows this one in our set of DIBs for the Physical device.

**DIB Extension Pointer:** This is a long word pointer that points to an area used for additional data about the device that the driver needs to maintain for it's operation.

**DIB Device Number:** This is the number used to reference this DIB and is used in the Forward and Head Link Device Number fields.

### **DIB Extension Data.**

In each DIB built by this driver, there is an extension that contains data required by this driver to function in the best possible manner. Each field used by this driver is discussed below.

The order in which these are presented is not necessarily the order of their occurrence. There are at least two reasons for this.

First, things change. This is not a problem as long as the code knows what is taking place. And since the code built the extension, this is not a problem.

Second and most important. The Driver is the only thing in existence that should be looking at what is stored there in the first place.

**Starting Block Number:** This long-word value is used only when partitions are present. Normally, 0's will be stored here. When a request is made to read a specific block number, this value will be added to give us the physical block number on the device where the data requested resides. For character devices, this has no meaning.

**Head Pointer:** This three byte field is an actual address for where the first DIB for this device can be found. This again is only used when partitions are present.

**Forward Pointer:** This three byte field points to the next DIB associated with this device in the same manner as the Head Pointer points to the first DIB for this device.

**Memory DIB Count:** This word value is only valid when within the first DIB of this memory segment. It is used to indicate how many active DIBs remain. During a shutdown call to any DIB within this memory segment, this value will be decremented and upon reaching zero, this memory segment will be released via the memory manager.

**Device Flag:** This group of sixteen bits is used for flags such as Internal Busy (the device has a pending call currently), no-wait mode, Device Online, and Device.

**Handle:** This long-word value is the handle associated by the memory manager with this memory segment. Any call to the memory manager concerning this segment will be referenced by this handle. Normally this will be when this segment is disposed of as per the discussion for the Memory DIB Count above.

**Block Size:** The data for this four byte field is also taken from the MODE SENSE call to the device through the SCSI Manager. This information is three bytes long and will be in bytes five through seven of the *Block Descriptor* field of the returned data. It will also be in a High to Low format and is stored in the DIB extension.

**Max Command:** This word value is the maximum command accepted by this device. SCSI has allowed for commands in the range of \$00-\$FF, but it may be that this device will accept no command greater than \$25, for example. There is no need to waste the user's time if we know the command will not be accepted.

**Command Bitmaps:** The DIB extension data is built from the data returned in the INQUIRY call. These data can not simply be moved in. They must be handled on a group by group basis. Not all groups will be represented in the return data. This will be explained in more detail in the next section.

**SCSI Manager Command Area:** The following several bytes are used as work space for the command structures sent to the SCSI Manager.

**Completion Vector:** This section contains code that when called will locate itself and clear an internal busy flag for this device when an

asynchronous completes. Until this flag is cleared, no other commands to this device will be accepted. If sent, the driver will wait for the busy bit to clear rather than return an error to the caller.

### **Building the DIBs in three short SCSI Calls.**

Above we briefly described how to build the DIB on a field by field basis. Not much detail was given because of the inefficiency of that method. Here we will describe in detail the calls to be made to the SCSI Device and show what data fields to expect in return.

Besides the Get\_Devices call that we have already issued, we will need to make three additional calls to the device. To get all the device information that we need to build the DIB we need to issue the INQUIRY, MODE SENSE and READ CAPACITY commands. We will start with the INQUIRY Command.

**INQUIRY:** In **Figure 8** we have the command block for the first of the calls that we need to issue. This is the INQUIRY call and is covered in the ANSI® X3.131-1986 SCSI document starting on page 69. The only field that we need to worry about is the *Allocation Length*. If this is zero then none of the important values will be returned. To account for any anomalies between devices, it might be wise to set this to \$FF.

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation Code \$12							
1	Logical Unit #							
2	Reserved							
3	Reserved							
4	Allocation Length							
5	Vendor Unique		Reserved				Flag	Link

**Figure 8**

In response to this call the device will return the structure shown in **Figure 9**. Although we need only a couple of fields from this to finish off our DIB, it might be prudent to check some of the other fields also. Byte zero is the *Peripheral Device Type*. This can be checked to ensure that this is the type of device that the driver wishes to converse with.

Byte one bit 7, if set to 1, indicates that the media is removable.

Bytes \$05 - \$25 can be used to verify a particular vendor or model of the device. The revision number can also be checked here. These could have meaning to a driver that is written for a specific vendor's product.

Starting at Byte \$26 is the start of the command bitmaps. These are arranged as 5 byte sets. The first byte is the group number and will be in the range of 0 - 7. An \$FF indicates that there are no more command groups to follow. The next 4 bytes are the bitmap itself. Byte 0 Bit 7 = command 0, bit 0 = command 7, Byte 1 bit 7 = command 8 and so on. These fields will need to be moved individually into the DIB extension to allow for holes in the bitmap. This is because the device may only return commands for group 0 and group 6 with nothing in between.

Bit	7	6	5	4	3	2	1	0
Byte								
0	Peripheral Device Type							
1	RMB	Device-Type Qualifier						
2	ISO Version		ECMA Version		ANSI-Approved Ver			
3	Reserved							
4	Additional Length							
<b>Vendor Unique Parameters</b>								
5	Vendor Unique							
6-7	Reserved							
8-F	Vendor ID in ASCII							
10-1F	Product ID in ASCII							
20-23	Revision Level in ASCII							
24-25	Number of Resrve/Release extents							
26	Group Number							
27-2A	Group Commands Implemented							
2B	Group Number							
2C-2F	Group Commands Implemented							
n	\$FF end of commands Implimented							

00h = Direct Access Device  
01h = Sequential Access Device  
02h = Printer Device  
03h = Processor Device  
04h = WORM Device  
05h = ROM Direct Access Device (CD)  
06h-7Eh = Reserved  
7Fh = Logical Unit not present  
80h-FFh = Vendor Unique

See SCSI Spec. Page 70

Figure 9

**MODE SENSE:** This command is covered in the ANSI® X3.131-1986 SCSI document starting on page 108. This is a 6 byte command from the group 0 list of commands. The parameter or command list is shown in **Figure 10**.

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation Code \$1A							
1	Logical Unit #							
2	Reserved							
3	Reserved							
4	Allocation Length							
5	Vendor Unique		Reserved				Flag	Link

**Figure 10**

In response to this call the data structure shown in **Figure 11** will be returned into your buffer. From this information you can determine whether or not the device is write protected and the size of each blocks in bytes, and the block count. Not all devices return this block data. Block device drivers should use the READ CAPACITY Call. That is all that we need from this call. It should be noted that even though this call does return the block count for this device, (if a block device) we will not be using this data.

Bit	7	6	5	4	3	2	1	0		
Byte										
0	Sense Data Length								Total Length of all Data Returned	
1	Medium Type									
2	WP	Reserved							WP = Write Protected Bit (1 = Protected)	
3	Block Descriptor Length									
Block Descriptor(s)										
0	Density Code								<div>\$00 Default (Only one density supported) \$01 Flexible disk, single density \$02 Flexible disk, double density \$03 - \$7F Reserved \$80 - \$FF Vendor Unique</div>	
1	Number of Blocks (MSB)									
2	Number of Blocks									
3	Number of Blocks (LSB)									Number of Blocks on the device
4	Reserved									
5	Block Length (MSB)								Block Size in Bytes	
6	Block Length									
7	Block Length (LSB)									
Vendor Unique Parameter(s)										
0 - N	Vendor Unique Parameter Byte(s)									

**Figure 11**

**READ CAPACITY:** This command is also covered in the ANSI® x3.131-1986 and SCSI-2 document. This is a 10 byte command from the group 1 list of commands. The parameter or command list is shown in **Figure 12**.

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation Code \$25							
1	Logical Unit #							
2	Logical Block Address (MSB)							
3	Logical Block Address							
4	Logical Block Address							
5	Logical Block Address (LSB)							
7	Reserved							
8	Allocation Length							
9	Reserved							
A	Vendor Unique		Reserved				Flag	Link

**Figure 12**

In response to this call the data structure shown in **Figure 13** will be returned into your buffer. The READ CAPACITY Call is used in the case of a block device to find the total size of the device in blocks and also the size of the block in bytes. The block number given is the last readable block on the disk. To get the total block count, the block number is incremented by one. This is to account for block zero being the first block. This information is also supplied by the MODE SENSE Call but not by all devices, so rather than take a chance on the values, we will use this call for this information.

Bit	7	6	5	4	3	2	1	0
Byte								
0	Logical Block Address (MSB)							
1	Logical Block Address							
2	Logical Block Address							
3	Logical Block Address (LSB)							
4	Block Length (MSB)							
5	Block Length							
7	Block Length							
8	Block Length (LSB)							

**Figure 13**

**Reading the Partitions:** Block devices can be partitioned into several smaller logical volumes. These volumes can vary in size and structure or Operating System. The Operating System type will be up to the FSTs to manage. The size is our responsibility. Using the READ CAPACITY Call, the driver knows the total size of this device. If partitioned, this is no more than the sum of all the partition sizes. The driver must therefore read the Partition Map and build a DIB for each of the partitions found except for those of the types 'Apple\_partition\_Map', 'Apple\_Scratch', and 'Apple\_Free' (these strings are not case sensitive).

The structure of the Partition Map is given in the section for STATUS Calls.

When partitioned media is encountered, the driver will need to ensure that the Head Link and Forward Device Link pointers are set correctly. There is also a bit in the device characteristics word of the DIB that must be set. The driver must also ensure that the block count matches the one from the Partition Map and not the READ CAPACITY block count.



## Driver Open Call

```
Call Parameters : Device Number      ≠      $0000
                  Call Number        =      $0001
                  DIB Pointer
```

Device Number:     *This word parameter specifies the target device.  
This parameter must be nonzero.*

Call Number:        *This word parameter specifies the type of call.*

DIB Pointer:        *This longword points to the DIB for the target device.*

The `Driver_Open` call is used to open a character device for data I/O. If this call is received by a driver written for a block device, it should take no action and return to the caller with no error.

In the case of character devices, this call should be handled on a device by device basis. Some devices would use the *LOAD/UNLOAD* commands. Other devices might use the *START/STOP Unit* calls. This can be any call or series of calls that ensure that the device is online and ready to respond to the users requests.

The driver should maintain a flag to indicate whether or not a previous open has been received without an accompanying close. If two opens are received the driver returns an **ALREADY OPEN** error.

## **Driver Read Call**

Call Parameters : Device Number        ≠        \$0000  
                  Call Number        =        \$0002  
                  Buffer Pointer  
                  Request Count  
                  Transfer Count  
                  Block Number  
                  Block Size        (Char. = \$0000, Block ≠ \$0000)  
                  FST Number        (Block device only)  
                  Cache Priority     (Block device only)  
                  Volume ID        (Block device only)  
                  DIB Pointer

Device Number:     *This word parameter specifies the target device.  
                     This parameter must be nonzero.*

Call Number:        *This word parameter specifies the type of call.*

Buffer Pointer:     *This longword points to where the data is to be  
                     returned.*

Request Count:     *This longword specifies the number of bytes requested  
                     from the target.*

Transfer Count:     *This longword indicates the number of bytes actually  
                     transferred.*

Block Number:      *This longword parameter is the logical block address  
                     from which data is to be read from. This parameter  
                     has no application in character device drivers.*

Block Size:        *This word parameter specifies the size of the block  
                     addressed by block number. This parameter must be a  
                     nonzero value for block devices or a zero for  
                     character devices.*

FST Number:        *This word parameter specifies the File System  
                     Translator that owns the volume from which the block  
                     is being read. When set, the most significant bit of  
                     the FST number will force device access during the  
                     read even if the block being accessed is in the  
                     cache. In this case no cache access will occur. If  
                     a zero, then the call came from a device call from  
                     the application.*

Cache Priority:     *This word parameter specifies whether or not caching  
                     is to be invoked for the block specified in the  
                     current I/O transaction. A value of \$0000 specifies  
                     that the block is not to be placed into the cache. A  
                     cache enable of \$0001 through \$7FFF implements  
                     caching using an LRU algorithm. If no space is*

available to cache the block, the least recently used purgable block will be purged to make room for caching the block in the current I/O request. A cache enable of \$8000 through \$FFFF also use the LRU algorithm but will be cached as deferred unpurgable blocks. Non-deferred cached blocks are cached by device number while deferred cached blocks are cached by volume ID. Read operations do not invoke deferred caching. The device dispatcher limits cache priorities to the range of \$0000 to \$7FFF during read operations. This parameter has no application in character device drivers.

Volume ID:           This word is used to identify deferred cached blocks belonging to a specific volume.

DIB Pointer:          This longword points to the DIB for the target device.

This call is supported by both character and block device drivers. It translates into a SCSI Read Command. If the driver is for a character device then it must have an active **Opened** device. If an I/O transaction is attempted with a device that has not been opened, the driver will return a 'NOTOPEN' error. Block devices do not have this requirement. In either case, the transfer count will be set to zero prior to any calls to the device and then updated by the driver as data is received. A character device should also verify that the block size is set to zeros. If the size is non-zero, a 'NOT A BLOCK DEVICE' error shall be returned.

Block devices also need to check that the block size is correct for the targeted device. If the block size is incorrect it will return a 'BAD CALL PARAMETER', or if the request count is not an integral multiple of block size, the driver should return an 'INVALID BYTE COUNT' error. If the block number (range if Request Count is greater than Block Size) is not within the legal range, a 'BAD BLOCK' error should be the response. As blocks are read from the device, the transfer count will be incremented by block\_size bytes until Request Count bytes have been transferred. If the blocks are read as part of a multi-block request, the count will be updated at the end of that request.

## Driver Write Call

Call Parameters :	Device Number	≠	\$0000
	Call Number	=	\$0003
	Buffer Pointer		
	Request Count		
	Transfer Count		
	Block Number		
	Block Size	(Char. = \$0000, Block ≠ \$0000)	
	FST Number	(Block devices only)	
	Cache Priority	(Block devices only)	
	Volume ID	(Block devices only)	
	DIB Pointer		

Device Number: *This word parameter specifies the target device. This parameter must be nonzero.*

Call Number:        *This word parameter specifies the type of call.*

Buffer Pointer: *This longword points to where the data is to be written from.*

Request Count:     *This longword specifies the number of bytes requested to send.*

Transfer Count: *This longword indicates the number of bytes sent to the target.*

Block Number: *This longword parameter is the logical block address to which data is to be written to. This parameter has no application in character device drivers.*

Block Size:        *This word parameter specifies the size of the block addressed by block number. This parameter must be a nonzero value for block devices or a zero for character devices.*

FST Number: This word parameter specifies the File System Translator that owns the volume to which the block is being written. The most significant bit of the FST number has no effect on a write call. If zero, then the call came from the application via a device call.

Cache Priority:     *This word parameter specifies whether or not caching is to be invoked for the block specified in the current I/O transaction. A value of \$0000 specifies that the block is not to be placed into the cache. A cache enable of \$0001 through \$7FFF implements caching using an LRU algorithm. If no space is available to cache the block, the least recently used purgable block will be purged to make room for caching the block in the current I/O request. A cache enable of \$8000 through \$FFFF also use the LRU algorithm but will be cached as deferred unpurgable blocks. Non-deferred cached blocks are cached by device number while deferred cached blocks are cached by volume ID. This parameter has no application in character device drivers.*

Volume ID:           *This word is used to identify deferred cached blocks belonging to a specific volume.*

DIB Pointer:          *This longword points to the DIB for the target device.*

This call is supported by both character and block device drivers. It transfers data from the buffer specified in the parameter block on direct page to the target device. Like the Driver Read Call, this will translate into a SCSI Write command regardless of whether the device is a block or character device.

If a character device is the target, the driver shall verify that the device has been and still is opened. If the device is not opened, the driver will return a '**NOT\_OPEN**' error. The block size should be set to zero or a '**NOT\_A\_BLOCK\_DEVICE**' error will be returned. As data is written to the device the transfer count will be incremented. This count should have been set to zero on entry to this call.

When the target device is a Block device the Block Size shall be validated. If it is incorrect it will return a 'BAD CALL PARAMETER', or if the request count is not an integral multiple of block size, the driver will return an 'INVALID BYTE COUNT' error. If the block number (range if Request Count is greater than Block Size) is not within the legal range, a 'BAD BLOCK' error will be the response. The driver resets the transfer count to zero and as data is written the count will be incremented by Block Size for every block read. If the driver reaches the end of the disk before request count blocks but after at least one block is read then no error shall be returned and the transfer count will reflect correctly the amount returned. If the blocks are read as part of a multi-block request, the count will be updated at the end of that request.

## Driver Close Call

```

Call Parameters : Device Number      ≠      $0000
                  Call Number        =      $0004
                  DIB Pointer

```

Device Number: *This word parameter specifies the target device. This parameter must be nonzero.*

Call Number:        *This word parameter specifies the type of call.*

DIB Pointer:        *This longword points to the DIB for the target device*

This call is the counter to the **Driver\_Open** Call and is used when talking to character devices. On receiving this call, the driver should return to the same state that was in effect when the **Driver\_Open** call was issued. All memory from the Memory Manager is released, the internal driver open flag is set to not open. If the driver was not in the open state when this call was issued a **NOT\_OPEN** Error is returned.

If the Driver is a Block Device Driver no action will be taken and the driver returns no error.

## Driver Status Call

```

Call Parameters : Device Number      ≠      $0000
                  Call Number        =      $0005
                  Status List Pointer
                  Request Count
                  Transfer Count
                  Status Code
                  DIB Pointer

```

Device Number:     *This word parameter specifies the target device.  
This parameter must be nonzero.*

Call Number:        *This word parameter specifies the type of call.*

Status List Pointer:     *This longword points to where the status list is to be returned.*

Request Count: *This longword indicates the number of bytes to be returned. If the request count is smaller than the minimum buffer size required by the call, an error is returned.*

Transfer Count: *This longword indicates the number of bytes actually transferred.*

Status Code: This word parameter specifies the type of status request. Status codes of \$0000 through \$7FFF are standard calls that must be supported by device drivers. SCSI specific status calls use status codes in the range of \$8000 through \$FFFF.

The list of standard status calls are as follows:

\$0000	Return Device Status
\$0001	Return Configuration Parameters
\$0002	Return Wait/No Wait Status
\$0003	Return Format Options
\$0004	Return Partition Map
\$0005	Return Last Command Result
\$0006 - \$7FFF	Reserved - These status codes to be assigned by Apple Computer, Inc.

The list of device specific status calls are as follows:

\$8000 - \$80FF Device specific SCSI commands.

*The following are the SCSI specific status calls.  
The values to the right are the Devices that use that  
code and are defined at the end of the list.*

#### Group 0 Commands

\$8000	Test Unit Ready	0,A,B,C,D,E
\$8003	Request Sense	0,A,B,C,D,E
\$8005	Read Block Limits	2
\$8006	Receive QIC-100 System Data	E
\$8008	Read	1,2,5,6,8,B
	Receive	4
	Get Message	A
\$800C	Reserved	
\$800D	Read SCSI Defect Data	E
\$800E	Read Controller Information	E
\$800F	Read Reverse	2
\$8011	Read Drive Lines	E
\$8012	Inquiry	0,A,B,C,D,E
\$8013	Read QIC-100 Information	E
\$8014	Recover Buffered Data	2,3
\$8019	Read QIC-100 Defect Data	E
\$801A	Mode	Sense 1,2,3,5,6,7,8,A,B,C, D,E
\$801C	Receive Diagnostic Results	0,A,B,E
\$801F	Read Log	2

#### Group 1 Commands

\$8020 - \$8023	Reserved	
\$8025	Read Capacity	1,5,6,8,B,E
	Get Window Parameters	7
\$8026 - \$8027	Reserved	
\$8028	Read	1,5,6,7,8,B,C,E
\$8029	Reserved	
\$802D	Read Update Block	8
\$8034	Read Position	2
	Get Data Status	7,C
\$8037	Read Defect Data	1,E
\$8038	Read Element Status	9
\$803C	Read Buffer	1,2,7,8,A,B,E
\$803E	Read Long	1

#### Group 2 Commands

\$8040 - \$8041	Reserved	
\$8042	Read Sub-Channel	
\$8043	Read TOC	
\$8044	Read Header	
\$8045 - \$804B	Reserved	
\$805A	Mode Sense	0



\$805B - \$805E	Reserved	
\$805F	Read Log	0

### Group 3 Commands

\$8060 - \$807F Reserved

### Group 4 Commands

\$8080 - \$809F Reserved

### Group 5 Commands

\$80A0 - \$80A7	Reserved	
\$80A8	Read	8
\$80A9 - \$80AC	Reserved	
\$80AD	Read Update Block	8
\$80B4 - \$80B6	Reserved	
\$80B7	Read Defect Data	8
\$80B8 - \$80BC	Reserved	
\$80BE - \$80BF	Reserved	

### Group 6 Commands

\$80C1	Read TOC	B
\$80C2	Read Q Subcode	B
\$80C3	Read Header	B
\$80C4 - \$80C7	Reserved	
\$80CC	Audio Status	B
\$80CE - \$80DF	Reserved	

## Group 7 Commands

\$80E0 - \$80FF Reserved

### Device Definitions

0	=	Devices '1' through '9'
1	=	Direct-Access Devices
2	=	Sequential-Access Devices
3	=	Printer Devices
4	=	Processor Devices
5	=	Write-Once Read-Multiple Devices
6	=	Read-Only Direct-Access Devices
7	=	Scanner Devices
8	=	Optical Memory Devices
9	=	Changer Devices
A	=	Communications Devices
B	=	Apple CD_ROM Drive
C	=	Apple SCSI Scanner
D	=	Apple LaserWriter SC
E	=	Apple Tape Drive

## Non-SCSI Commands

\$8100 - \$FFFF Reserved

DIB Pointer:        *This longword points to the DIB for the target device.*

This call is used to obtain current information from the device or driver itself and may be used to issue any extended SCSI Command that returns data from the device through the use of device specific control codes. The device driver is responsible for validating the status code prior to executing the requested command. The following 65816 code sample shows how this might be done.

```

validate_stat_code    lda     status_code           ;Get Status Code
                     bmi     device_specific        ;Is it Device Specific?
                     cmp     #max_command+1        ;No. Is it out of range?
                     blt     do_standard            ;No. Goto code segment
bad_code_exit         lda     #BAD_CODE             ;Central BAD CODE Exit
                     sec
                     rtl

device_specific        and     #$00ff               ;Is it ≤ the max SCSI
                     pha                     ;command for this
                     inc                     ;device?
                     ldy     #SCSI_maxcmd_offset
                     cmp     [DIB_PTR],y
                     blt     so_far_so_good         ;It's in a good range
                     pla                     ;Restore stack
                     bra     bad_code_exit          ;exit with error

```

```

so_far_so_good      lda    1,s                ;Get status code from stack
                    lsr                    ;Generate a group index
                    lsr                    ;to use to get to the
                    lsr                    ;correct group offset
                    and    #$000e
                    tay
                    lda    group_offset,y
                    sta    temp                ;Save offset for later

                    lda    1,s                ;Get status code from stack
                    and    #$0010            ;Upper or lower half
                    beq    first_16_bits      ;of group bitmap?
                    inc    temp              ;Upper half. Adjust
                    inc    temp              ;offset by two

first_16_bits        pla                    ;Last time. Get status code
                    and    #$000f            ;Retain low nibble
                    asl    a                ;Account for 16 bit
                    tay                    ;table of offsets.
                    lda    cmd_bm_mask,y
                    ldy    temp
                    and    [DIB_PTR],y      ;Is bit set in bitmap?
                    beq    bad_code_exit     ;No. Error exit.
                    .
                    .
                    .

max_command          equ    $0004            ;Max standard status code

SCSI_maxcmd_offset   equ    $0042

group_offset         dc    i2'GROUP0-DIB_start' ;Offsets from start
                    dc    i2'GROUP1-DIB_start' ;of DIB to Group
                    dc    i2'GROUP2-DIB_start' ;bitmaps
                    dc    i2'GROUP3-DIB_start'
                    dc    i2'GROUP4-DIB_start'
                    dc    i2'GROUP5-DIB_start'
                    dc    i2'GROUP6-DIB_start'
                    dc    i2'GROUP7-DIB_start'

temp                ds    2

cmd_bm_mask          dc    i2'%1000000000000000' ;Bitmap masks.
                    dc    i2'%0100000000000000'
                    dc    i2'%0010000000000000'
                    dc    i2'%0001000000000000'
                    dc    i2'%0000100000000000'
                    dc    i2'%0000010000000000'
                    dc    i2'%0000001000000000'
                    dc    i2'%0000000100000000'
                    dc    i2'%0000000010000000'
                    dc    i2'%0000000001000000'
                    dc    i2'%0000000000100000'
                    dc    i2'%0000000000010000'
                    dc    i2'%0000000000001000'
                    dc    i2'%0000000000000100'
                    dc    i2'%0000000000000010'
                    dc    i2'%0000000000000001'

```

If an invalid status code is passed to the driver, the driver will return a 'BAD CODE' error. The driver may also wish to identify where the call came from and shield certain calls from the application. Based on this, the driver could use a secondary command bitmap to validate codes allowed from the application. If an invalid control list length is passed to the driver, the driver should return a 'BAD PARAMETER' error. The device driver sets the transfer count to the number of bytes returned as a result of the status call.

**NOTE:** Both standard and device specific status calls may detect an OFFLINE or DISKSWITCH status. If either of these conditions are detected then a **SET\_DISKSW** call is issued to set the device dispatcher maintained disk switched error.

## \$0000 - Return Device Status

This call returns a general status followed by a longword specifying the target device in blocks. The block count for character devices will be returned as zero. The general status word indicates the status of the device.

The bit definition within the general status word for character devices is as follows:

Bit 15	Reserved (currently read as zero)
Bit 14	1 = Linked Device
Bit 6-13	Reserved (currently read as zero)
Bit 5	Buffer not empty
Bit 4	1 = Online, 0 = Offline
Bit 2-3	Reserved (currently read as zero)
Bit 1	1 = Device currently interrupting
Bit 0	1 = Device currently open

The bit definition within the general status word for block devices is as follows:

Bit 15	1 = Block count is uncertain for current block size
Bit 14	1 = Linked Device
Bit 5-13	Reserved (currently read as zero)
Bit 4	1 = Disk in drive, 0 = Disk not in drive
Bit 3	Reserved (currently read as zero)
Bit 2	1 = Write protected, 0 = Write enabled
Bit 1	1 = Device currently interrupting
Bit 0	1 = Disk has been switched

If either bit 0 or bit 4 has been set then the driver will call the SET\_DISKSW via the system service call table.

Disk switched status should be set on ejection for optimum performance. A status call should only return an error code if the status call fails. Error

codes should not be returned for conditions indicated with the general status word.

Status List:	Word	General status word
	Long	Number of blocks supported by device

### **\$0001 - Return Configuration Parameters**

This call returns a byte count as the first word in the status list which indicates the length of the configuration parameter list in bytes. The configuration parameters will be placed into the status list contiguous to the byte count. The structure of the configuration parameter list is device dependent.

Status List:	Word	Length of configuration parameter list
	Data	Returned configuration parameters

### **\$0002 - Return Wait/No Wait Status**

This call is used to determine if a device is in wait mode. If in wait mode, the device will wait for the number of characters specified in the request count of a read call before returning from the read. If in No Wait mode, a read call will return immediately with a transfer count indicating the number of characters returned. If a character was available the transfer count will return from a read with a non zero value. If a character was not available the transfer count will return from a read call with a value of zero. A word with a value of \$0000 returned in the status list indicates that the device is operating in Wait mode. A word with a value of \$8000 returned in the status list indicates that the device is operating in No Wait mode. Block devices always operate in Wait mode.

Status List:	Word	Wait/No Wait status
--------------	------	---------------------

### **\$0003 - Return Format Options**

This call returns a list of formatting options that may be selected using a Set\_Format\_Options call prior to issuing a format call to a block device. These parameters may include such variables as format environment, number of blocks, block size, and interleave. Devices that do not support media variables will return with a transfer count of zero and no error. The format of the status list on return from this call when a device does support media variables is as follows:

Returned List:	Word	Number of entries in list
	Word	Number of displayed entries in list
	Word	Recommended Default Option
	Word	Option that current online media is formatted with

## Apple IIgs® GS/OS® - SCSI Driver ERS 6.1 SCSI-2

Then each entry in the list consists of 16 bytes containing the following 5 fields:

Word	Media variables reference number
Word	Reference number of linked entry
Word	Flags
Long	Number of blocks supported by device
Word	Block Size
Word	Interleave Factor
Word	Media size (block count * block size)

Flags word definition is as follows:

Bits 0 - 1	Format Type
Bits 2 - 3	Size Multiplier
Bits 4 - 15	Reserved (must be zero)

Format Type definition is as follows:

00	Universal Format
01	Apple Format
10	Non-Apple Format
11	Not valid

Size Multiplier definition is as follows:

00	Size in bytes
01	Size in kilobytes
10	Size in megabytes
11	Size in gigabytes

A typical list returned from this call for a device supporting two possible interleaves intended to support Apple's file systems (ProDOS, GS/OS, MFS or HFS) might be as follows:

Transfer count	\$00000038	56 bytes returned in list
Returned List:	\$0003	Three entries in list
	\$0002	Only two display entries
	\$0001	Recommended default is option #1
	\$0003	Current media is formatted as specified by option #3
	\$0001	Refnum = Option #1
	\$0002	LinkRef = Option #2
	\$0008	Universal format / size in megabytes
	\$0000A2F9	Block count = 41721
	\$0200	Block size = 512 bytes
	\$0001	Interleave factor = 1:1
	\$0020	Media size = 20 megabytes
	\$0002	Refnum = Option #2
	\$0000	LinkRef = none
	\$0008	Universal format / size in megabytes
	\$0000A2F9	Block count = 41721
	\$0200	Block size = 512 bytes
	\$0002	Interleave factor = 2:1

\$0020	Media size = 20 megabytes
\$0003	Refnum = Option #3
\$0000	LinkRef = none
\$0008	Universal format / size in megabytes
\$00009C8C	Block count = 40076
\$0214	Block size = 532 bytes
\$0005	Interleave factor = 5:1
\$0019	Media size = 19 megabytes

Character devices should return no error.

### **\$0004 - Return Partition Map**

This call returns the partition map of the device specified which must be the first device of any linked list. If not an **Invalid Device Number** error is returned.

The size of the partition map can vary and the only way for an application to be sure that the entire partition map has been read is to validate the pmMapBlkCnt field in the map with the transfer count returned by the driver. If pmMapBlkCnt \* block size ≠ Transfer Count then the application must reissue the call with the appropriate request count. If Request Count is not an integral multiple of block size, then an **Invalid Byte Count** error is returned. All calls requesting the partition map will return the data starting with the first entry. There is no allowance for reading a specific entry in the partition map. If the application wishes to reassign an entry to a different type OS, the \$0006 control call Assign Partition Owner should be used.

The structure of the partition map is described in detail in '**Inside Macintosh™** Volume V' under the SCSI Manager section. Below (**Figure 14**) is a picture of what a partition map entry looks like along with definitions of a few of the fields. It must be noted that all fields, except strings, in the partition map are stored High Byte »» Low Byte. Example: Even though the partition signature is defined as \$504d, it will appear to the 65xxx family of processors as \$4d50.

byte 0	pmSig (word)	always \$504d
2	pmSigPad (word)	reserved for future use
4	pmMapBlkCnt (long word)	number of blocks in map
8	pmPyPartStart (long word)	first physical block of partition
C	pmPartBlkCnt (long word)	number of blocks in partition
10	pmPartName (32 bytes)	partition name
30	pmPartType (32 bytes)	partition type
50	pmLgDataStart (long word)	first logical block of data area
54	pmDataCnt (long word)	number of blocks in data are
58	pmPartStatus (long word)	partition status information
5C	pmLgBootStart (long word)	first logical block of boot code
60	pmBootSize (long word)	size in bytes of boot code
64	pmBootLoad (long word)	boot code load address
68	pmBootLoad2 (long word)	additional boot load information
6C	pmBootEntry (long word)	boot code entry point
70	pmBootEntry2 (long word)	additional boot entry information
74	pmBootCksum (long word)	boot code checksum
78	pmProcessor (16 bytes)	processor type
88	( 128 bytes )	boot specific arguments

**Figure 14**

The only fields that shall be defined here are the pmPartType (Partition Type) and the pmProcessor (Processor Type). These are ASCII strings of 1 to 32 or 16 bytes respectively in length; case is not significant. If either name is less than the max character length, it must be terminated with a NULL character (ASCII code 0). An empty name can be specified by setting the first byte to the NULL character.

Example TypesExample Processors

Apple_MFS	68000	8080	80186	6502
Apple_HFS	68008	8085	80286	65C02
Apple_Unix_SVR2	68010	Z80	80386	65802
Apple_partition_map	68012	8086	80486	65816
Apple_Driver	68020	8088		
Apple_PRODOS	68030	Z8000		
Apple_Free	68040	6800		
Apple_Scratch		6809		

**\$0005 - Return Last Command Result**



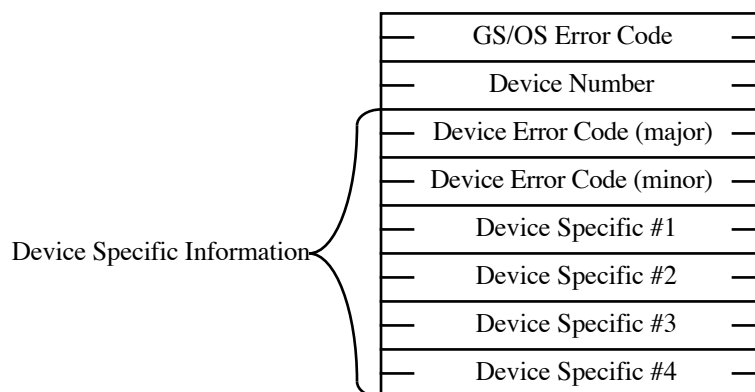
This command is used to query the driver for the results of the last command issued (because the Apple SCSI Drivers do Auto-Sensing, this is the only way to get accurate sense data from the device when using Device Specific Status and Control Calls). This is used in the case of an Async command to verify that it did perform the requested action. (Example. An Async Format call is issued to the device that may take several minutes. The command may have returned no error up front, but then have problems performing the actual Format of the media. In this way the application could be aware that the media is defective and warn the user).

This call would return request length bytes up to 16 bytes maximum. The returned information would be structured as below.

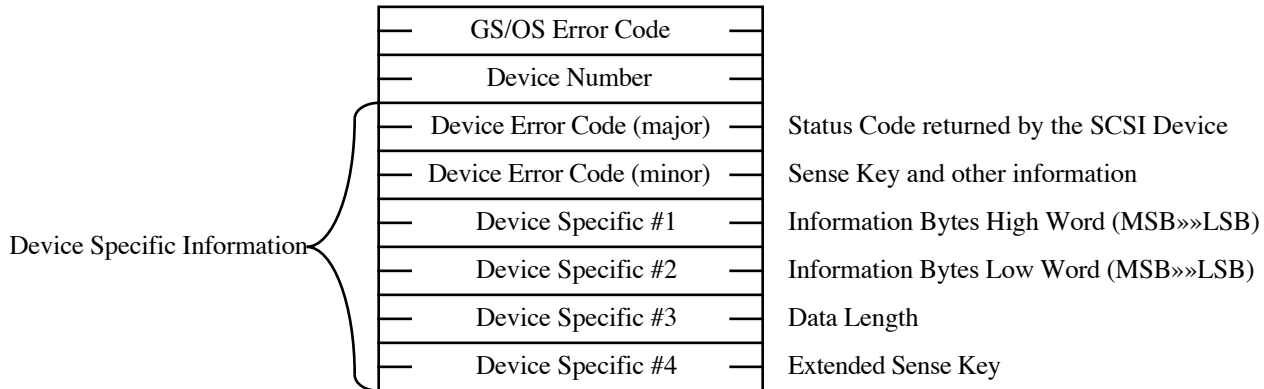
The first word would be from the set of GS/OS error codes and would reflect the problem encountered while performing the last requested transaction as closely as possible.

The second word would be the Device Number. This would aid in deciphering the device specific information that follows.

The next 12 bytes would be the device specific information pertaining to the device in question. (This information will be null if the request was successfully completed).



In the world of SCSI, the device specific information would be as pictured below and the 12 bytes of device specific information are as follows:



**Device Error Code (major)** - This is the status code returned from the target device during the Status Phase and are as defined in the following list.

rr00000r	=	Good. No Error. ('r' = Reserved)
rr00001r	=	Check Condition
rr00010r	=	Condition Met/Good
rr00100r	=	Busy
rr01000r	=	Intermediate/Good
rr01010r	=	Intermediate/Condition Met/Good
rr01100r	=	Reservation Conflict
rr10100r	=	Queue Full

All others are Reserved

**Device Error Code (minor)** - This is a combination of several bits and the Sense Key. These bits and key are defined as follows.

Bit 7	=	1 indicates that the command read a filemark. This is only used for sequential access devices.
Bit 6	=	1 indicates an end of medium condition. It is used in sequential access and printer devices. This includes End-of-tape, beginning-of-tape, out-of-paper, ect. Not used in direct access devices.
Bit 5	=	1 indicates that the requested logical block length did not match the logical block length of the data on the media. (Please refer to the ANSI SCSI Spec for a discussion of logical block lengths).
Bit 4	=	Reserved

Bit 3-0	=	Sense Key.
0	=	No Sense
1	=	Recovered Error
2	=	Not Ready
3	=	Medium Error
4	=	Hardware Error
5	=	Illegal Request
6	=	Unit Attention
7	=	Data Protect
8	=	Blank Check
9	=	Vendor Unique
A	=	Copy Aborted
B	=	Aborted Command
C	=	Equal
D	=	Volume Overflow
E	=	Miscompare
F	=	Reserved

**Device Specific #1 and #2** - These are used to form a Long result ordered from MSB to LSB and contain information about the failure. A few values will be discussed briefly and the reader is urged to read the ANSI SCSI Spec for further details concerning this information.

(1) The unsigned block address associated with the sense key.

(2) The difference (residue) of the requested length minus the actual length in either bytes or blocks, as determined by the command. (Negative values are indicated by two's complement notation).

**Device Specific #3** - The length of the data returned if the next command is a Request Sense Command requesting further data concerning the error.

**Device Specific #4** - This will contain the extended Sense Key that was returned by the device and defined in the ANSI SCSI Spec as well as the Device Vendors own documentation.

## Device Specific Status Calls

The remaining Status Code definitions pertain to the SCSI Device specific commands. To issue a single command, the caller will use the parameter block structure (**Figure 15**) below. The Buffer Pointer on direct page points to this structure and the REQUEST COUNT, also on direct page, is the number of bytes total requested by this command.

(NOTE: The request count should not be larger than what is allowed by the device for the call being issued. Also attention must be paid to the min. request count. Some calls, READ\_CAPACITY, return data regardless of the

count. The driver can not verify these. Space must be allocated for this result. Read Capacity always returns 8 bytes. Request Sense always returns at least 4 bytes. And so on.)

The first word is the call version number. To maintain compatibility with previous drivers written for the GS/OS Operating System, we need to distinguish between Versions or class of SCSI Device specific commands. The first release supported version \$0000 only. Under version \$0000, only the version number itself, the Command Specific Data and the buffer pointer were included. In this version of the driver, we will additionally support version \$0001.

The next field is the twelve byte command area and contains the information as outlined in the definition of the command that is being issued. This field is present in all versions.

The Buffer Pointer is where the data for the call is to be found at, returned to, depending on the call being sent. This field is present in all versions.

The remaining fields are supported by version \$0001, and are discussed in detail, along with the previous two fields and how they work together, in the text below.

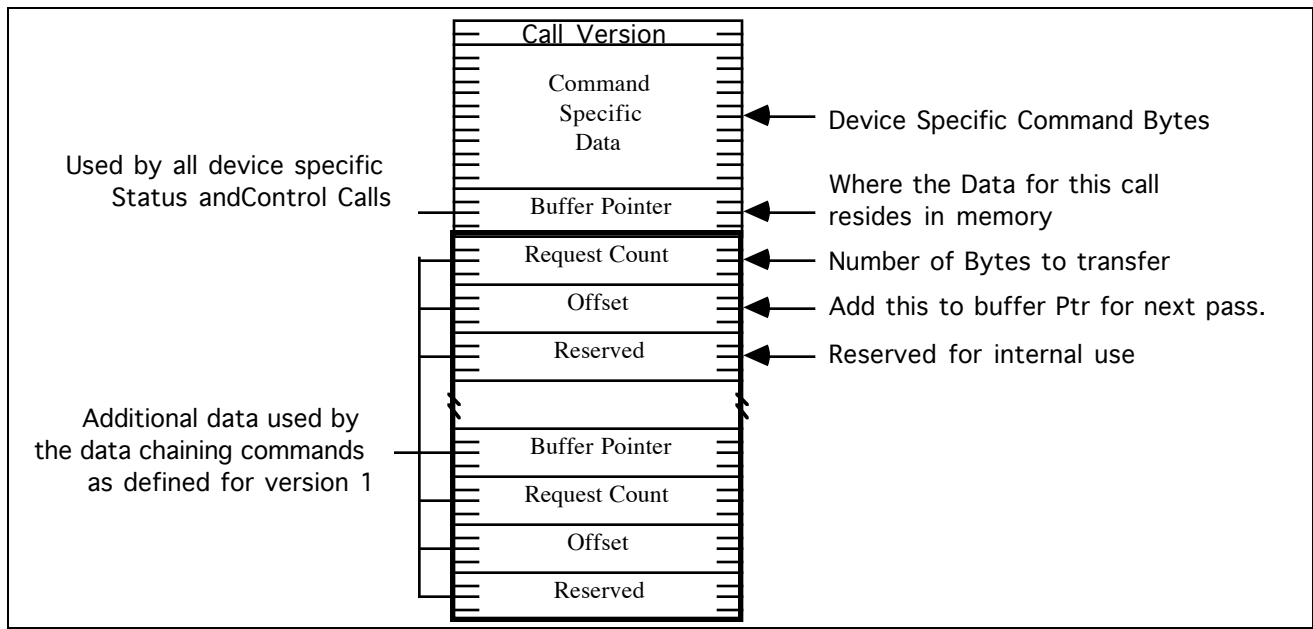


Figure 15

There are times when a transaction must be broken up into it's separate components. In these situations the Application might choose to use the data chaining capabilities of the SCSI Driver/Manager. These data chaining commands are each composed of four long word fields and are in the normal Low »» High format.

There are three commands that compose the data chaining structure. By using them together in various combinations data can be gathered or distributed depending on the direction of transfer. The three commands are listed below:

- The first is a value in the range of \$00000001 - \$fffffffe. Any value in this range will be treated as a buffer address. This is referred to as the DCMove instruction.
- The second command is \$ffffffff. This is the DCLoop Command.
- The third, \$00000000, is the DCSpecial Command.

Depending on what is in the Buffer field, the next field will be interpreted as given below.

- If a DCMove instruction, then this is a requested byte count.
- This is interpreted as a count for the number of times to loop through the buffer entries for the DCLoop Command.
- In the case of a DCSpecial Command, this field is used by the application to either stop the sequence of Data Chaining Commands (\$00000000) or an address of a routine to call if the field is non-zero (<>\$00000000). This can be used by the application to indicate the action to be taken such as a graphics page flip or some other task that would need to be performed mid transfer.

The third field also has three definitions depending on the contents of the buffer address. The are as follows.

- Offset to add to the buffer address for the next pass for DCMove.
- Number of entries to go back to restart the list when DCLoop.
- Reserved and should be null for the DCSpecial STOP instruction. If not a STOP Command, then this can contain flags to indicate where in the data chaining command structure the SCSI Manager is currently pointing.

The fourth field is for internal usage and must be null in all cases.

**(NOTE: The request count times the loop count plus any other request counts may not be larger than the REQUEST COUNT used when issuing the call. If it is larger, a time-out error will be returned, the data chaining structures will have been modified and the data returned is unreliable.)**

Two examples for using these commands are given below. One for receiving data and the other for sending. It should be noted that there is no support implied for the devices used in the following examples.

#### **Sending Data:**

A printer is connected to the SCSI interface that requires a header or declaration to be sent prior to sending the image to

be printed. The image resides in memory as contiguous pages and is rather large. Each page of data is the same size.

Rather than breaking up the data to be printed and shuffling it with copies of the header information, the Application could instead use the following data chaining commands.

DCMove	\$00000200	\$00000000	\$00000000
DCMove	\$00004000	\$00004000	\$00000000
DCLoop	\$00000019	\$fffffffe (-2)	\$00000000
DCStop	\$00000000	\$00000000	\$00000000

The first DCMove is the pointer to the header information that will be sent each time through the loop. It is 512 bytes in size and we want to sent the same data on each pass of the loop.

The second DCMove points to the first image to be sent. Each page is 16 KB in size and we want to send the next page which starts right after the current one on the next pass of the loop.

DCLoop tells us to loop \$19 times and to go back 2 instructions to the first DCMove each time. When the loop executes the \$19th pass it will drop through to the next command which is the DCStop. This will signal the end of the transmission and the call will return via the normal path to the Application.

#### Receiving Data:

An imaging device is connected to the SCSI interface that will be sending a large image to be displayed. The video mapping as it is in the Apple II family is not structured in a way that allows an image to be read directly in and displayed. The data must be moved around. By structuring the data chaining command, a HIRES Bitmap type Image can be read directly in video memory without having to adjust it first.

Rather than breaking up the Read calls to the device and spending a lot of time in overhead, the Application could instead use the following data chaining commands.

DCMove	\$00000028	\$00000028	\$00000000
DCMove	\$00000028	\$00000028	\$00000000
DCMove	\$00000028	\$00000028	\$00000000
DCMove	\$00000028	\$00000028	\$00000000
DCMove	\$00000028	\$00000028	\$00000000
DCMove	\$00000028	\$00000028	\$00000000
DCMove	\$00000028	\$00000028	\$00000000
DCMove	\$00000028	\$00000028	\$00000000
DCLoop	\$00000002	\$ffffff8 (-8)	\$00000000
DCStop	\$00000000	\$00000000	\$00000000

The first DCMove is the pointer to \$0400. This is the first line of video. This will increment by \$28 on each pass.

The second DCMove is the pointer to \$0480. This is the second line of video. This will increment by \$28 on each pass.

The third DCMove is the pointer to \$0500. This is the third line of video. This will increment by \$28 on each pass.

This continues on to the eighth DCMove which points to \$0780.

DCLoop tells us to loop 2 times and to go back 8 instructions to the first DCMove each time. When the loop executes the 3rd pass ( 1st sequence plus 2 loops = 3 passes) it will drop through to the next command which is the DCStop. This will signal the end of the transmission and the call will return via the normal path to the Application.

**Note: Commands are always checked for validity prior to issuance to the SCSI Manager. If any command is unacceptable, the driver returns the error in the Acc. the carry is set.**

The list of each Device Specific Status Code follows with each one giving a detailed description of what the Command Data looks like and what data is returned in the buffer as well as what data is needed in the buffer for a few of the calls.

**\$8000 - Test Unit Ready; ( All Devices )**  
**[M]**

This command must be accepted by all devices and is used to check if the target device is ready to accept media access commands. This command does not cause the device to do a self test. If no error is returned, the device is ready for access. If the device supports removable media and the media is not inserted, an error is returned.  
**!!!! Define error handling and error codes to be returned!!!!**

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$00 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused.

**Transfer Length:**

Unused.

**Buffer Data Structure:**

None.

**Errors:**

Good  
Not Ready



## **\$8003 - Request Sense; ( All Devices )**

**[M]**

The Request Sense command must be accepted by all devices and is used to retrieve Sense data from the target device. The sense data is valid for the last command to the target device until the target has received another command. That is this command if it is issued. The data returned is defined by the ANSI® X3.131-1986 and SCSI-2 Specs. **Must request at least 4 bytes.**

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$03 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000004 - \$000000ff (Bytes) No error if larger

### **Transfer Length:**

\$00000004 - \$000000ff

### **Buffer Data Structure:**

Sense Data as defined by the ANSI® X3.131-1986 and SCSI-2 Specs are returned in the specified buffer. See also Vendors Device documentation.

### **Errors:**

Good  
Check Condition.

## **\$8005 - Read Block Limits; ( Sequential Access )**

**[M]**

The Read Block Limits call is aimed at Sequential Access Devices and is mandatory for those devices. It returns the minimum and maximum block size for that device. See the ANSI® x3.131-1986 and SCSI-2 Spec for more details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$05 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00000006. A value less than 6 will return an error.  
A larger value will only return 6 bytes

### **Transfer Length:**

\$00000000 if an Error  
\$00000006 if successful

### **Buffer Data Structure:**

The data returned by this command is shown below. If the maximum length equals the minimum length, then only fixed length blocks of the size indicated are supported. Otherwise the block size can vary between the two ranges. If the max block size = 0 then no upper limit is specified.

\$00	Reserved
\$01	Maximum Block Len (MSB)
\$02	Maximum Block Len
\$03	Maximum Block Len (LSB)
\$04	Minimum Block Len (MSB)
\$05	Minimum Block Len (LSB)

### **Errors:**

Good  
Check Condition.

## \$8006 - Receive QIC-100 System Data; (Apple Tape Drive)

The Receive QIC-100 System Data call is aimed at the Apple Tape Backup Drive only and is mandatory for that device. This command causes the controller to transfer 128 bytes of QIC-100 System Data to the host.

The data that is sent is the contents of the System Data bytes in the buffer memory. If the last command that was executed was a Read command, then the contents of these bytes will be the System Data bytes for the last block read.

See the 3M MCD-40 SCSI Spec for more details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$06 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$0B	Reserved	

### Request Length:

\$00000000 - \$00000080. A value less than 80 will return an error. A larger value will only return 80 bytes

### Transfer Length:

\$00000000 if an Error  
\$00000080 if successful

### Buffer Data Structure:

The data returned by this command is shown below.

\$00	System Data read with the first
	fram of User Data
\$3f	
\$40	System Data read with the second
	fram of User Data
\$7f	

### Errors:

Good  
Check Condition.

**\$8008 - Read; ( Direct Access )**  
**[M]**

This is the standard SCSI Read Command used by most devices that return data to the caller. If a block address greater than \$ffff is needed, a command \$8028 should be used.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$08 )
	\$01	Reserved	
	\$02 - \$03	Logical Block Address	( MSB »» LSB )
	\$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Buffer Data Structure:**

None.

**Errors:**

Good  
Check Condition.  
Reservation Conflict

**NOTE:** See also **\$8008 - Read, Receive** and **Get Message** described below.

**\$8008 - Read; ( Sequential Access )**  
**[M]**

This is the standard SCSI Read Command used by most devices that return data to the caller. If a block address greater than \$ffff is needed, a command \$8028 should be used.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$08 )
	\$01	Reserved	( %xxxxxx00 )
		SILI	( %000000x0 )
		Fixed	( %0000000x )
	\$02 - \$03	Logical Block Address	( MSB »» LSB )
	\$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Buffer Data Structure:**

None.

**Errors:**

Good  
 Check Condition.  
 Reservation Conflict

**NOTE:** See also **\$8008 - Read** above, and **Receive** and **Get Message** described below.

**\$8008 - Receive; (Processor Devices)**  
**[0]**

This command is used in the same manner as the read command that shares the Status Code with this command. There are some minor differences though. For further details please refer to the vendors documentation for more details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$08 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

Data

**Errors:**

Good  
Not Ready

**NOTE:** See also **\$8008** - Both **Reads** above, and **Get Message** described below.

**\$8008 - Get Message; (Communication Devices)**  
**[M]**

This command is used in the same manner as the read command that shares the Status Code with these commands. There are some minor differences though. For further details please refer to the vendors documentation for more details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$08 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

None.

**Errors:**

Good  
Not Ready

**NOTE:** See also **\$8008** - Both **Reads** and **Receive** above.

**\$800D - Read SCSI Defect Data; (Apple Tape Drive)**

This command is supported by the Apple Tape Backup Drive but shall be unsupported by any SCSI Driver. The \$8037 command shall be the command of choice to receive this information.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0D )
	\$01	Reserved	( %xxxx0000 )
		CMPLST	( %0000x000 )
		Reserved	( %00000xxx )
	\$02 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

None.

**Errors:**

Not Supported



**\$800E - Read Controller Information; (Apple Tape Drive)**

This command is supported by the Apple Tape Backup Drive and shall be supported by the SCSI Driver. Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0E )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this structure.

**Errors:**

Check Condition

**\$800F - Read Reverse; ( Sequential Access )**  
**[0]**

The Read Reverse command is used to optimize for head movement when using a Sequential Access Device. If the head is past the desired block this call would cause the media to move backwards across the head while the data was read. This is more efficient than resetting the head to the beginning of the data and then reading forward again.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0F )
	\$01	Reserved	( %xxxxxx00 )
		SILI	( %000000x0 )
		Fixed	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (See flag definition)

**Transfer Length:**

\$00000000 - \$00ffffff (See flag definition)

**Buffer Data Structure:**

Data.

**Errors:**

Good  
Check Condition.

## **\$8011 - Read Drive Lines; (Apple Tape Drive)**

The Read Drive Lines command allows the host computer to read the input lines from the target.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$11 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00000001 (Bytes)

### **Transfer Length:**

\$00000000 - \$00000001 (Bytes)

### **Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

### **Errors:**

Good  
Check Condition.

## **\$8012 - Inquiry; ( All Devices )** **[M]**

This command retrieves information about the target device regarding various parameters and configuration of same.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$12 )
	\$01	Reserved	( \$00 )
	\$02	Reserved	( \$00 )
	\$03	VPD Identifier	( \$xx )
	\$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$000000ff (Bytes)

### **Transfer Length:**

\$00000000 - \$000000ff (Bytes)

### **Buffer Data Structure:**

The information returned by this call is displayed graphically below. Each field is defined as follows.

Byte 0	Peripheral Device Type.
00	= Direct-Access Device
01	= Sequential-Access Device
02	= Printer Device
03	= Processor Device
04	= Write-Once Read-Multiple Device
05	= Read-Only Direct-Access Device
06	= Scanner Device
07	= Optical Memory Device
08	= Changer Device
09	= Communications Device
0A - 0F	= Reserved
10	= Direct Access Magnetic Tape Device
11 - 7E	= Reserved
7F	= Logical Unit not present
80 - FF	= Vendor Unique
Byte 1, Bit 7	Removable Media Flag
Bit 0-6	Device Type Qualifier

# Apple IIgs® GS/OS® - SCSI Driver ERS 6.1 SCSI-2

Byte 2                      ANSI® Version

- 0        Uses SCSI prior to x3.131-1986 approval
- 1        SCSI-1 standard (ANSI® x3.131-1986).
- 2        SCSI-2 standard
- 3 - 7   Reserved

Byte 3                      Reserved

Byte 4                      Additional Sense Length

Bit	7	6	5	4	3	2	1	0
Byte								
0	Peripheral Device Type							
1	RMB	Device-Type Qualifier						
2	ISO Version		ECMA Version			ANSI-Approved Ver		
3	Reserved							
4	Additional Length							
Vendor Unique Parameters								
5	Vendor Unique							
6-7	Reserved							
8-F	Vendor ID in ASCII							
10-1F	Product ID in ASCII							
20-23	Revision Level in ASCII							
24-25	Number of Resrve/Release extents							
26	Group Number							
27-2A	Group Commands Implemented							
2B	Group Number							
2C-2F	Group Commands Implemented							
n	\$FF end of commands Implimented							

## Errors:

Good  
Check Condition.

## NOTE:

For addition information describing the required data returned by this call, please refer to the 'Apple SCSI Hard Disk Common Command Protocol' document. The information contained therein should apply to all devices that are to function in the Apple environment.

**Additional notes on the INQUIRY command.**

The INQUIRY command (see table 44) requests that information regarding parameters of the target and its attached peripheral device(s) be sent to the initiator. An option allows the initiator to request additional information about the target or logical unit.

Table 44 - INQUIRY command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (12h)							
1	Logical unit number			Reserved				EVPD
2	Page code							
3	Reserved							
4	Allocation length							
5	Control							

An enable vital product data (EVPD) bit of one specifies that the target shall return the optional vital product data specified by the page code field. If the target does not support vital product data and this bit is set to one, the target shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

An EVPD bit of zero specifies that the target shall return the standard INQUIRY data. If the page code field is not zero, the target shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

The page code field specifies which page of vital product data information the target shall return.

The INQUIRY command shall return CHECK CONDITION status only when the target cannot return the requested INQUIRY data.

NOTE 64 The INQUIRY data should be returned even though the peripheral device may not be ready for other commands.

If an INQUIRY command is received from an initiator with a pending unit attention condition (i.e. before the target reports CHECK CONDITION status), the target shall perform the INQUIRY command and shall not clear the unit attention condition.

**NOTES**

65 The INQUIRY command is typically used by the initiator after a reset or power-up condition to determine the device types for system configuration. To minimize delays after a reset or power-up condition, the standard INQUIRY data should be available without incurring any media access delays. If the target does store some of the INQUIRY data on the device, it may return zeros or ASCII spaces (20h) in those fields until the data is available from the device.

66 The INQUIRY data may change as the target executes its initialization sequence or in response to a CHANGE DEFINITION command. For example, the target may contain a minimum command set in its non-volatile memory and may load its final firmware from the device when it becomes ready. After it has loaded the firmware, it may support more options and therefore return different supported options information in the INQUIRY data.

### Standard INQUIRY data

The standard INQUIRY data (see table 45) contains 36 required bytes, followed by a variable number of vendor-specific parameters. Bytes 56 through 95, if returned, are reserved for future standardization.

Table 45 - Standard INQUIRY data format

Bit	7	6	5	4	3	2	1	0
Byte								
0	Peripheral qualifier			Peripheral device type				
1	RMB	Device-type modifier						
2	ISO version		ECMA version			ANSI-approved version		
3	AENC	TrmIOP	Reserved		Response data format			
4	Additional length (n-4)							
5	Reserved							
6	Reserved							
7	RelAdr	WBus32	WBus16	Sync	Linked	Reserved	CmdQue	SftRe
8	(MSB)							
15	Vendor identification							(LSB)
16	(MSB)							
31	Product identification							(LSB)
32	(MSB)							
35	Product revision level							(LSB)
36								
55	Vendor-specific							
56								
95	Reserved							
	Vendor-specific parameters							
96								
n	Vendor-specific							

The peripheral qualifier and peripheral device-type fields identify the device currently connected to the logical unit. If the target is not capable of supporting a device on this logical unit, this field shall be set to 7Fh (peripheral qualifier set to 011b and peripheral device type set to 1Fh). The peripheral qualifier is defined in table 46 and the peripheral device type is defined in table 47.

Table 46 - Peripheral qualifier

Qualifier	Description
000b	The specified peripheral device type is currently connected to this logical unit. If the target cannot determine whether or not a physical device is currently connected, it shall also use this peripheral qualifier when returning the INQUIRY data. This peripheral qualifier does not mean that the device is ready for access by the initiator.
001b	The target is capable of supporting the specified peripheral device type on this logical unit; however, the physical device is not currently connected to this logical unit.
010b	Reserved
011b	The target is not capable of supporting a physical device on this logical unit. For this peripheral qualifier the peripheral device type shall be set to 1Fh to provide compatibility with previous versions of SCSI. All other peripheral device type values are reserved for this peripheral qualifier.
1XXb	Vendor-specific

Table 47 - Peripheral device type

Code	Description
00h	Direct-access device (e.g. magnetic disk)
01h	Sequential-access device (e.g. magnetic tape)
02h	Printer device
03h	Processor device
04h	Write-once device (e.g. some optical disks)
05h	CD-ROM device
06h	Scanner device
07h	Optical memory device (e.g. some optical disks)
08h	Medium changer device (e.g. jukeboxes)
09h	Communications device
0Ah - 0Bh	Defined by ASC IT8 (Graphic arts pre-press devices)
0Ch - 1Eh	Reserved
1Fh	Unknown or no device type

A removable medium (RMB) bit of zero indicates that the medium is not removable. A RMB bit of one indicates that the medium is removable. The device-type modifier field was defined in SCSI-1 to permit vendor-specific qualification codes of the device type. This field is retained for compatibility with SCSI-1. Targets that do not support this field should return a value of zero.



The usage of non-zero code values in the ISO version and ECMA version fields are defined by the International Organization for Standardization and the European Computer Manufacturers Association, respectively. A zero code value in these fields shall indicate that the target does not claim compliance to the ISO version of SCSI (ISO 9316) or the ECMA version of SCSI (ECMA-111). It is possible to claim compliance to more than one of these SCSI standards.

The ANSI-approved version field indicates the implemented version of this International Standard and is defined in table 48.

Table 48 - ANSI-approved version

Code	Description
0h	The device might or might not comply to an ANSI-approved standard.
1h	The device complies to ANSI X3.131-1986 (SCSI-1).
2h	The device complies to this version of SCSI. This code is reserved to designate this standard upon approval by ANSI.
3h - 7h	Reserved

The asynchronous event notification capability (AENC) bit indicates that the device supports the asynchronous event notification capability.

a) Processor device-type definition: An AENC bit of one indicates that the processor device is capable of accepting asynchronous event notifications. An AENC bit of zero indicates that the processor device does not support asynchronous event notifications.

b) All other device-types: This bit is reserved.

A terminate I/O process (TrmIOP) bit of one indicates that the device supports the TERMINATE I/O PROCESS message. A value of zero indicates that the device does not support the TERMINATE I/O PROCESS message.

A response data format value of zero indicates the INQUIRY data format is as specified in SCSI-1. A response data format value of one indicates compatibility with some products that were designed prior to the development of this standard (i.e. CCS). A response data format value of two indicates that the data shall be in the format specified in this International Standard. Response data format values greater than two are reserved.

The additional length field shall specify the length in bytes of the parameters. If the allocation length of the command descriptor block is too small to transfer all of the parameters, the additional length shall not be adjusted to reflect the truncation.

A relative addressing (RelAdr) bit of one indicates that the device supports the relative addressing mode for this logical unit. If this bit is set to one, the linked command (Linked) bit shall also be set to one; since relative addressing can only be used with linked commands. A RelAdr bit of zero indicates the device does not support relative addressing for this logical unit.

A wide bus 32 (Wbus32) bit of one indicates that the device supports 32-bit wide data transfers. A value of zero indicates that the device does not support 32-bit wide data transfers.

A wide bus 16 (Wbus16) bit of one indicates that the device supports 16-bit wide data transfers. A value of zero indicates that the device does not support 16-bit wide data transfers.

NOTE 67 If the values of both the Wbus16 and Wbus32 bits are zero, the device only supports 8-bit wide data transfers.

A synchronous transfer (Sync) bit of one indicates that the device supports synchronous data transfer. A value of zero indicates the device does not support synchronous data transfer.

A linked command (Linked) bit of one indicates that the device supports linked commands for this logical unit. A value of zero indicates the device does not support linked commands for this logical unit.

A command queuing (CmdQue) bit of one indicates that the device supports tagged command queuing for this logical unit. A value of zero indicates the device does not support tagged command queuing for this logical unit.

A soft reset (SftRe) bit of zero indicates that the device responds to the RESET condition with the hard RESET alternative. A SftRe bit of one indicates that the device responds to the RESET condition with the soft RESET alternative.

ASCII data fields shall contain only graphic codes (i.e. code values 20h through 7Eh). Left-aligned fields shall place any unused bytes at the end of the field (highest offset) and the unused bytes shall be filled with space characters (20h). Right-aligned fields shall place any unused bytes at the start of the field (lowest offset) and the unused bytes shall be filled with space characters (20h).

The vendor identification field contains eight bytes of ASCII data identifying the vendor of the product. The data shall be left aligned within this field.

NOTE 68 It is intended that this field provide a unique vendor identification of the manufacturer of the SCSI device. In the absence of a formal registration procedure, X3T9.2 maintains a list of vendor identification codes in use. Vendors are requested to voluntarily submit their identification codes to X3T9.2 to prevent duplication of codes.

The product identification field contains sixteen bytes of ASCII data as defined by the vendor. The data shall be left-aligned within this field.

The product revision level field contains four bytes of ASCII data as defined by the vendor. The data shall be left-aligned within this field.

### **Vital product data**

Implementation of vital product data is optional. The information returned consists of configuration data (e.g. vendor identification, product identification, model, serial number), manufacturing data (e.g. plant and date of manufacture), field replaceable unit data and other vendor- or device-specific data.

The initiator requests the vital product data information by setting the EVPD bit to one and specifying the page code of the desired vital product data. If the target does not implement the requested page it shall return CHECK CONDITION status. The a sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

#### NOTES

69 The target should have the ability to execute the INQUIRY command even when a device error occurs that prohibits normal command execution. In such a case, CHECK CONDITION status should be returned for commands other than INQUIRY or REQUEST SENSE. The sense data returned may contain the field replaceable unit code. The vital product data should be obtained for the failing device using the INQUIRY command.

70 This International Standard defines a format that allows device-independent initiator software to display the vital product data returned by the INQUIRY command. For example, the initiator may display the data associated for the field replaceable unit returned in the sense data. The contents of the data may be vendor-specific; therefore, it may not be usable without detailed information about the device.

71 This International Standard does not define the location or method of storing the vital product data. The retrieval of the data may require completion of initialization operations within the device that may induce delays before the data is available to the initiator. Time-critical requirements are an implementation consideration and are not addressed in this International Standard.

### **\$8013 - Read QIC-100 Information; (Apple Tape Drive)**

The Read QIC-100 Information command allows the host computer to read QIC-100 related information from the target.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$13 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$000000ff (Bytes)

#### **Transfer Length:**

\$00000000 - \$000000ff (Bytes)

#### **Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

#### **Errors:**

Good  
Check Condition.

## **\$8014 - Recover Buffered Data; ( Sequential Access )**

### **[0]**

This call is similar in function to the Read Command. The only difference being that the data requested is read from the target devices buffer memory rather than the media. This call is used to recover data from a terminated write call that was unable to write the data to the media from the buffer. This situation could arise during an asynchronous write call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$14 )
	\$01	Reserved	( %00000000 )
		Fixed	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)  
See FIXED bit definition in ANSI® and Vendor documents, driver may need to translate to Blocks)

### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Buffer Data Structure:**

None.

### **Errors:**

Good  
Check Condition

**\$8014 - Recover Buffered Data;      ( Printer Devices )**  
**[0]**

This call is similar in function to the Read Command. The only difference being that the data requested is read from the target devices buffer memory rather than the media. This call is used to recover data from a terminated Print call that was unable to print the data to the media from the buffer. This situation could arise during an asynchronous print call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$14 )
	\$01	Reserved	( %00000000 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

None.

**Errors:**

Good  
Check Condition

### **\$8019 - Read QIC-100 Defect Data; (Apple Tape Drive)**

The Read QIC-100 Defect Data command allows the host computer to read defect data from the controller. If this command follows a Format or Verify Unit command, the data returned to the host are the defects found for that verify. If the defect data is not in memory at the time this command is received, the manufacturers' block is read

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$19 )
	\$01	Immed	( %x00000000 )
		CompLst	( %0000x000 )
	\$02 - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$000000ff (Bytes)

#### **Transfer Length:**

\$00000000 - \$000000ff (Bytes)

#### **Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

#### **Errors:**

Good  
Check Condition.

**\$801A - Mode Sense; ( All Devices )**  
**[0]**

This command is complimentary to the Mode Select Command. It is used to get the Medium, Logical Unit, or Peripheral Device parameters from the target device.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1A )
	\$01	Page Format (PF)	( %000x0000 )
		DBD Flag	( %0000x000 )
	\$02	Page Control	( %xx000000 )
		Page Code	( %00xxxxxx )
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$000000ff (Bytes)

**Transfer Length:**

\$00000000 - \$000000ff (Bytes)

**Buffer Data Structure:**

The information returned by this call is displayed graphically below. Each field is defined as follows.

Byte 0	Sense Data Length
Byte 1	Medium Type
Byte 2, Bit 7	Write Protected flag
Byte 3	Block Descriptor Length

This is then followed by the Block Descriptor(s). Each block descriptor contains a block count and block length. Following this is any vendor unique information that the target may attach.



Bit	7	6	5	4	3	2	1	0
Byte								
0	Sense Data Length							
1	Medium Type							
2	WP	Reserved						
3	Block Descriptor Length							
Block Descriptors								
0	Density Code							
1	Number of Blocks (MSB)							
2	Number of Blocks							
3	Number of Blocks (LSB)							
4	Reserved							
5	Block Length (MSB)							
6	Block Length							
7	Block Length (LSB)							
Vendor Unique Parameters								
0 - n	Vendor Unique Parameter Byte(s)							

**Errors:**

Good

**\$801C - Receive Diagnostic Results;      ( All Devices )**  
**[0]**

This command requests that the target device send it's analysis data after completing a Send Diagnostic Command. The data passed by this is vendor unique.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1C )
	\$01	Page Format (PF)	( %000x0000 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$0000ffff (Bytes)

**Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

**Buffer Data Structure:**

Entirely Vendor Unique

**Errors:**

Good  
Check Condition

**\$801F - Read Log;                      ( Sequential Access Devices )**  
**[0]                                      See \$805F command**

The Read Log command is issued to sequential access devices to request statistical information maintained by the device. This is an optional command and the data is returned in the same format as the Mode Select /Sense page codes.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1F )
	\$01	Page Format (PF)	( %000x0000 )
		No Log Save (NLS)	( %000000x0 )
		No Log Clear (NLC)	( %0000000x )
	\$02	Page Code	( %00xxxxxx )
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$000000ff (Bytes)

**Transfer Length:**

\$00000000 - \$000000ff (Bytes)

**Buffer Data Structure:**

The data is returned as vendor unique as well as Device or Medium specific

**Errors:**

Good  
 Check Condition

**\$8025 - Read Capacity; ( Direct Access )**  
**[M] ( WORM Devices )**  
**( Read-Only Direct Access )**

This mandatory group 2 command causes the target device to return information about the capacity of the logical unit. This command also allows the application to determine the free space left before the next mechanical delay takes place. The flags used in this command are described in the vendor and ANSI® SCSI Specs.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$25 )
	\$01	RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB »» LSB)
	\$06 - \$07	Reserved	
	\$08	PMI	( %0000000x )
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000008 All other values will be unpredictable or will return an error.

**Transfer Length:**

\$00000008

**Buffer Data Structure:**

Eight bytes are returned. The first four are the Logical block address for the last block on the device if the PMI bit is zero. If the PMI bit is 1, then this is the last block before a significant delay will take place before the next block can be read. The last four bytes is the length of the block in bytes.

Bit	7	6	5	4	3	2	1	0
Byte								
0	Logical Block Address (MSB)							
1	Logical Block Address							
2	Logical Block Address							
3	Logical Block Address (LSB)							
4	Block Length (MSB)							
5	Block Length							
6	Block Length							
7	Block Length (LSB)							

**Errors:**

Good

## **\$8025 - Get Window Parameters; ( Scanners )**

### **[0]**

The Get Window Parameters is used to pass information from the scanner detailing the task and how it is to be performed to the host. Before the scanner can scan a document or image, the application must provide certain details about the scan area. This information is provided in the form of parameters defining a scan window. These parameters include size, position, scanning resolution, scanning composition, as well as other parameters for each window.

The window parameters data shall consist of one or more Window Descriptor Blocks. Each window descriptor block specifies the location of a rectangular region and the mode in which the region is to be scanned.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$24 )
	\$01	Single	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Window Identifier	( \$xx )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$00ffffff (Length of window descriptor in bytes)

#### **Transfer Length:**

\$00000000 - \$00ffffff (Length of transferred data in bytes)

#### **Buffer Data Structure:**

Below is the structure for the parameters to be transferred by this call. These are given here because of the lack of available documentation detailing this call. The descriptions below are extracted from several documents to give an over all comprehensive description of the parameters listed.

The Request Length above is the length, in bytes, of a window descriptor set. This means that the Window Descriptor Length times the number of descriptors sent should be eight less than the Request Length. In other words Request Length = Window Descriptor Length \* number of descriptor blocks + 8.

All other bytes in the Descriptor header (the first 8 bytes) are Reserved

Bit Byte	7	6	5	4	3	2	1	0
0 - 5	Reserved							
6	Window Descriptor Block Length (MSB)							
7	Window Descriptor Block Length (LSB)							
	Window Desriptor Block(s)							
0	Window Identifier							
1	Reserved							Auto
2-3	X Resolution (MSB»»LSB)							
4-5	Y Resolution (MSB»»LSB)							
6-9	Upper Left X (MSB»»LSB)							
A-D	Upper Left Y (MSB»»LSB)							
E-11	Width (MSB»»LSB)							
12-15	Length (MSB»»LSB)							
16	Brightness							
17	Threshold							
18	Contrast							
19	Image Composition							
1A	Bits per Pixel							
1B-1C	Halftone Patern (MSB»»LSB)							
1D	RIF	Reserved				Padding Type		
1E-1F	Bit Ordering (MSB»»LSB)							
20	Compression Type							
21	Compression Argument							
22-N	Reserved							

Each window descriptor contains information about one window.

The **Window Identifier** field contains a number between 0 and 255, which uniquely identifies the window defined by the block descriptor. Use this unique identifier to indicate each window during data transfers and status requests.

The **X Resolution** specifies the horizontal resolution in pixels per inch. A value of zero indicates that the scanner should use it's default resolution.

The **Y Resolution** specifies the vertical resolution in lines per inch. A value of zero indicates that the scanner should use it's default resolution.

The **Upper Left X** specifies the location of the X-coordinate of the upper left corner of this rectangular window and is measured in pixels as defined by X Resolution. The point 0,0 is considered the most upper-left corner of the window.

The **Upper Left Y** specifies the location of the Y-coordinate of the upper left corner of this rectangular window and is measured in lines as defined by Y Resolution. The point 0,0 is considered the most upper-left corner of the window.

The **Width** specifies the window width in pixels from left to right.

The **Length** specifies the window width in lines from top to bottom.

The **Brightness** has a range of one (lowest setting) through 255 (highest setting) with zero specifying the default value.

The **Threshold** is just that, the threshold setting of the scanner. A zero indicates that the scanner should use it's default setting for this. One is the lowest and 255 is the highest with 128 being the nominal setting.

The **Contrast** has a range of one (lowest setting) through 255 (highest setting) with zero specifying the default value.

The **Image** specifies the type of image acquired and is defined by the following table.

<u>Code</u>	<u>Description</u>
00	Bi-level black and white
01	Dithered/halftone black and white
02	Multi-level black and white (gray scale)
03	Bi-level RGB Color
04	Dithered/halftone RGB Color
05	Multi-level RGB Color
06 - FF	Reserved

The **Bits per Pixel** specifies the number of bits to be used to define each pixel. The higher the Image setting, the greater the number of bits required for each pixel.

The **Halftone Pattern** specifies the halftone process by which multi-level data is converted to binary data. This field shall be used in conjunction with the Image Composition code specified above.

The **Reverse Image Format (RIF)** bit is applicable only for images represented by one bit per pixel. A RIF bit of zero indicates

that white pixels are to be indicated by zeros and black pixels are to be indicated by ones. A RIF bit of one indicates the opposite.

The **Padding Type** specifies what operation is to be done if the scanned data to be transmitted to the host is not an integral number of bytes. The padding type is defined below.

<u>Code</u>	<u>Description</u>
00	No padding
01	Pad with 0's to byte boundary
02	Pad with 1's to byte boundary
03	Truncate to byte boundary
04 - FF	Reserved

The **Bit Ordering** field defines the order in which data is transferred to the host from the window. Ordering will include direction of pixels in a scan line, direction of scan lines within a window and data packing within a byte.

The **Compression type** and **Argument** fields specify the compression technique to be applied to the scanned data prior to transmission to the host and are defined below.

<u>Code</u>	<u>Compression Type</u>	<u>Argument</u>
00	No compression	Reserved
01	CCITT Group III, 1 dimensional	Reserved
02	CCITT Group III, 2 dimensional	K factor
03	CCITT Group IV, 2 dimensional	Reserved
04 - 0F	Reserved	Reserved
10	Optical Character Recognition (OCR)	Vendor Unique
11 - 7F	Reserved	Reserved
80 - FF	Vendor Unique	Vendor Unique

#### **Errors:**

Good  
Check Condition



# **\$8028 - Read (Extended); ( Direct Access Devices )** **[M]**

The Read (Extended) Command allows more than 256 blocks of data to be read in from the target device. It also allows for larger media. All other facets are the same as the \$8008 Read Command. Please refer to the ANSI® Spec for details on the use of the flags.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$28 )
	\$01	DPO	( %000x0000 )
		FUA	( %0000x000 )
		RelAddr	( %00000000x )
	\$02 - \$05	Logical Block Address	(MSB »» LSB)
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

## **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

## **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

## **Buffer Data Structure:**

Data

## **Errors:**

Good  
 Check Condition  
 Reservation Conflict

**\$8028 - Read (Extended) ; ( Scanner Devices )**  
**[M]**

The Read (Extended) Command allows more than 256 blocks of data to be read in from the target device. It also allows for larger media. All other facets are the same as the \$8008 Read Command. Please refer to the ANSI® Spec for details on the use of the flags.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$28 )
	\$01	RelAddr	( %00000000x )
	\$02	Transfer Data Type	( \$xx )
	\$03	Reserved	
	\$04 - \$05	Transfer Identification	(MSB »» LSB)
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Buffer Data Structure:**

Data

**Errors:**

Good  
Check Condition  
Reservation Conflict

## **\$802D - Read Update Block; ( Optical Memory )**

**[0]**

This command requests that the target device transfer to the host data from the specified generation(s) and logical block(s). This command is more thoroughly discussed in the SCSI-2 ANSI® spec.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2D )
	\$01	DPO	( %000x0000 )
		FUA	( %0000x000 )
		RUBD	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB»»»LSB)
	\$06	Latest	( %x0000000 )
		Generation Addr (MSB)	( %0xxxxxxx )
	\$07	Generation Addr (LSB)	
	\$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Buffer Data Structure:**

Data

### **Errors:**

Good  
Check Condition  
Reservation Conflict

## \$8034 - Read Position; ( Sequential Access Devices ) [0]

Status Command \$8034 has two functions. The first function, Read Position, is an optional command for sequential access devices. It causes the target to return the current position of data blocks in the buffer and the position of the medium. When the buffer does not contain a whole block of data, or is empty, the two values are equal. No medium movement will result from this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$34 )
	\$01	Block Addr Type (BT)	( %0000000x )
	\$02 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

### Request Length:

\$00000014

### Transfer Length:

\$00000014

### Buffer Data Structure:

The following text comes from the draft spec ANSI® x3.131-198x and all references to partitions relate to the SCSI concept of partitions and not those as implemented by Apple Computer, Inc.

Bit	7	6	5	4	3	2	1	0
Byte								
0	BOP	LEP	Reserved			BPU	Reserved	
1	Partition Number							
2	Reserved							
3	Reserved							
4	Product Specific First Block Location (MSB)							
5	Product Specific First Block Location							
6	Product Specific First Block Location							
7	Product Specific First Block Location (LSB)							
8	Product Specific Last Block Location (MSB)							
9	Product Specific Last Block Location							
A	Product Specific Last Block Location							
B	Product Specific Last Block Location (LSB)							
C	Reserved							
D	Number of Blocks in Buffer (MSB)							
E	Number of Blocks in Buffer							
F	Number of Blocks in Buffer (LSB)							
10	Number of Bytes in Buffer (MSB)							
11	Number of Bytes in Buffer							
12	Number of Bytes in Buffer							
13	Number of Bytes in Buffer (LSB)							

*Beginning of partition (BOP) bit, when set to one, indicates that the device's current logical position is at the beginning-of-partition. When set to zero it indicates that the current logical position is not at the beginning-of-partition. The BOP indication is not necessarily a result of a physical tape marker (e.g. reflective marker). When the partition number field is zero BOP, set to one, indicates that the position is at beginning-of-tape.*

*Logical end of partition (LEP) bit, when set to one, indicates that the device's current logical position is between early-warning and physical end-of-partition within the current partition. When set to zero it indicates that the current logical position is not in this area of the current partition. The LEP indication is not necessarily a result of a physical tape marker (e.g. reflective marker).*

*Block position unknown (BPU) bit, when set to one, indicates that the device's logical position is not known or cannot be obtained.*

*The partition number field reports the partition number for the current logical position.*

*Product specific first block location bytes indicate the relative position of the first data block within a partition. The value shall be the position of the next data block to be transferred between the initiator and the target if the previous command was a READ or a WRITE. The value shall be the position of the last block position transferred to the initiator if the previous command was a READ REVERSE. The information is product specific and is provided to be used with the LOCATE command to place the device's logical position at the appropriate logical block on another medium in the case of unrecoverable errors on the first medium.*

*Product specific last block location bytes indicate the relative position of the last data block within a partition. The value shall be the position of the next data block to be transferred between the buffer and the medium if the previous command was a READ or a WRITE. The value shall be the position of the last data block read from the medium into the target's buffer if the previous command was a READ REVERSE. The information is product specific and is provided to be used with the LOCATE command to place the device's logical position at the appropriate logical block on another medium in the case of unrecoverable errors on the first medium.*

*Number of blocks in buffer field indicates the number of data blocks in the target's buffer that have not been transferred to the medium.*

*Number of bytes in buffer field indicates the total number of data bytes in the target's buffer that have not been transferred.*

**Errors:**

Good

## **\$8034 - Get Data Status;           ( Scanner Devices )**

### **[0]**

Status Command \$8034 has two functions. The second, Get Data Status, function is an optional command for scanner devices. It causes the target to return the scan data availability. The application program reports data availability for those windows that have been specified in the SCAN command. The format of the Get Data Status Command is shown below.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$34 )
	\$01	Wait	( %00000000x )
	\$02 - \$0B	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$02 - \$0B	Reserved	

The WAIT Flag indicates when the scanner will return data status to the host computer. A value of 1 indicates that the scanner is expected to wait for the quantity of data in the scanner's internal memory to exceed the limit set in the Mode Select command before responding with data. A value of 0 indicates that the target shall respond immediately with data whether or not data is available.

#### **Request Length:**

\$00000000 - \$0000ffff (Bytes)

#### **Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

#### **Buffer Data Structure:**

The Get Data Status command returns the following structure providing a format for retrieving information about the availability of image data.

Bit	7	6	5	4	3	2	1	0
Byte								
0	Data Transfer Length (MSB)							
1	Data Transfer Length							
2	Data Transfer Length (LSB)							
3	Reserved							
4	Window Identifier							
5	Reserved							
6	Buffer Space Available (MSB)							
7	Buffer Space Available							
8	Buffer Space Available (LSB)							
9	Scan Data Available (MSB)							
A	Scan Data Available							
B	Scan Data Available (LSB)							

The Data Length specifies the length in bytes of the data status available to be transferred to the host computer during the Get Data Status command. The data length size does not include the data length field. The data transferred to the host computer consists of multiple structures, each one comprising of 8 bytes each as defined by bytes \$4 - \$B in the table above. As defined by the window identifier, each returned structure is associated with a window descriptor.

The Block bit specifies the buffering capabilities of the scanner. A value of 1 indicates that the scanner must transfer all available data to the host computer before the scanner can generate more scan data. The bit is also set when the scanner has reached the end of the scan, even though the buffer may not be full. A value of 0 indicates that the scanner is not currently blocked due to a lack of available buffer space.

The Window Identifier parameter identifies the window associated with the returned structure. This value matches the window identifier in the Window Descriptor of the Define Window Parameters command. After completing the scan, the scanner indicates the end of the transfer by setting the data length to 0 and sends on data status.

The Buffer Space Available field indicates the amount of storage in bytes available for data transfers to the target. This field is valid only in scanners with the ability to accept data from a host computer for processing.

The Scan Data Available field indicates the amount of data in bytes available to be transferred from the target.

#### Errors:

Good



## **\$8037 - Read Defect Data; ( Direct Access Devices )**

### **[0]**

The Read Defect Data command requests that the target transfer the media defect data to the host computer. This command is optional for Direct Access Devices and is discussed in detail in the ANSI® x3.131-198x Spec.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$37 )
	\$01	Reserved	( \$00 )
	\$02	PList	( %000x0000 )
		GList	( %0000x000 )
		Defect List Format	( %00000xxx )
	\$03 - \$06	Reserved	
	\$07 - \$08	Allocation Length	( MSB»»LSB )
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

#### **Request Length:**

\$00000000 - \$0000ffff (Bytes)

#### **Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

#### **Buffer Data Structure:**

Please refer to the vendor and ANSI® SCSI documents for further details of this call.

#### **Errors:**

Good  
Check Condition

**\$8038 - Read Element Status; ( Changer Devices )**  
**[0]**

The Read Element Status command will cause the target to report the status of it's internal elements. Refer to the Device and ANSI® documents for further details of this command and the flags associated with it.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$38 )
	\$01	Reserved	
	\$02 - \$03	Starting Element Address	( MSB»»»LSB )
	\$04 - \$05	Number of Elements	( MSB»»»LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

Please refer to the vendor and ANSI® documents for details of the returned Element Status data.

**Errors:**

Check Condition

## **\$803C - Read Buffer; ( All Devices )**

**[0]**

The Read Buffer command is used along with the Write Buffer command as a diagnostic tool for testing target memory and the SCSI Bus integrity. This command shall not alter the media. The reader should refer to the ANSI® x3.131-198x spec for more detailed information about this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$3C )
	\$01	Mode	( %00000xxx )

Mode is defined as follows.

000	Combined Header and Data	Optional
001	Vendor Unique	Vendor Unique
010	Data	Optional
011	Descriptor	Optional
1xx	Reserved	Reserved

\$02	Buffer ID	
\$03 - \$05	Buffer Offset	(MSB »» LSB)
\$06 - \$08	Reserved	
\$09	Vendor Unique	( %xx000000 )
	Reserved	( %00xxxxxx )
\$0A - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00ffffff (Bytes)

### **Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

### **Buffer Data Structure:**

See ANSI® and vendor documentation for details pertaining to this command

### **Errors:**

Good  
Check Condition

**\$803E - Read Long; ( Direct Access Devices )**  
**[0]**

The Read Long command request that the target transfer data to the host computer. This data is implementation specific, but shall include the data bytes and the ECC bytes. Any other data correctable by ECC should also be included.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$3E )
	\$01	CORRECT	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB »» LSB)
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000000 - \$0000ffff (Bytes)

**Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

**Buffer Data Structure:**

Data from the target including ECC and other information  
(Implementation specific).

**Errors:**

Good  
Check Condition

**\$8042 - Read Sub Channel; ( New for SCSI-2 CD-ROM drives )****[0]**

The READ SUB-CHANNEL command (see table 251) requests that the target return the requested sub-channel data plus the state of audio play operations.

Table 251 - READ SUB-CHANNEL command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (42h)							
1	Logical unit number			Reserved			MSF	Reserved
2	Reserved	SubQ	Reserved					
3	Sub-channel data format							
4	Reserved							
5	Reserved							
6	Track number							
7	(MSB)	Allocation length						(LSB)
8								
9	Control							

NOTE 178 Sub-channel data returned by this command may be from the last appropriate sector encountered by a current or previous media accessing operation. When there is no current audio play operation, the target may access the media to read the sub-channel data. The target is responsible that the data returned are current and consistent. For example with sub-channel data format 0, the International Standard Recording Code (ISRC) data reported must have been read from the same track as the reported current position data.

See table 237 for a description of the MSF bit.

Table 237 - MSF address format

Bit	7	6	5	4	3	2	1	0
Byte								
0	Reserved							
1	M field							
2	S field							
3	F field							

An MSF (Minute Second Frame) bit of zero requests that the logical block address format be used for the CD-ROM absolute address field or for the offset from the beginning of the current track expressed as a number of logical blocks in a CD-ROM track relative address field. This track relative logical block address (TRLBA) value is reported as a negative value in twos- complement notation for transition areas that have decreasing MSF encoded relative addresses.

An MSF bit of one requests that the MSF format be used for these fields. In certain transition areas, the relative MSF addresses are decreasing positive values. The absolute MSF addresses are always increasing positive values.

The M, S, and F fields are expressed as binary numbers. The values match those on the media, except for the encoding. The ratios of M field units to S field units and S field units to F field units are reported in the mode parameters page.

The sub Q bit set to one requests that the target return the Q sub- channel data. The sub Q bit set to zero requests that no sub-channel data be returned. This shall not be considered an error.

NOTE 179 The other bits in this byte are reserved for future standardization when they may be defined to request other sub-channel data.

The sub-channel data format field specifies the returned sub channel data (see table 252). If this field is 00h, sub-Q channel data is returned. If this field is 01h, 02h or 03h, the requested sub-Q data item is returned.

Table 252 - Sub-channel data format codes

Format Code	Returned data
00h	Sub-Q channel data
01h	CD-ROM current position
02h	Media catalogue number (UPC/bar code)
03h	Track international standard recording code (ISRC)
04h - EFh	Reserved
F0h - FFh	Vendor-specific

The track number field specifies the track from which ISRC data is read. This field must have a value between 01h and 63h (99bcd), and is valid only when the sub-channel data format field is 03h. In this case, the target returns ISRC data for this track.

## Sub-Q channel data format

The sub-CHANNEL command data formats consist of a four-byte header followed by a sub-channel data block. The header contains

the audio status byte and the sub-channel data length field. If the sub Q bit is zero, the target shall not return the sub-channel data block; in this case, the sub-channel data length is 0.

Table 253 defines the sub-Q channel data format.

Table 253 - Sub-Q channel data format

Bit	7	6	5	4	3	2	1	0
Byte								
	Sub-channel data header							
0	Reserved							
1	Audio status							
2	(MSB)	Sub-channel data length						(LSB)
3								
	Sub-Q channel data block							
4	Sub channel data format code (00h)							
5	ADR				Control			
6	Track number							
7	Index number							
8	(MSB)	Absolute CD-ROM address						(LSB)
11								
12	(MSB)	Track relative CD-ROM address						(LSB)
15								
16	MCVal	Reserved						
17	(MSB)	Media catalogue number (UPC/Bar code)						(LSB)
31								
32	TCVal	Reserved						
33	(MSB)	Track international standard recording code (ISRC)						(LSB)
47								

The audio status field indicates the status of audio play operations. The audio status values are defined in table 254. Audio status values greater than zero are returned only to the initiator that requested the last audio play operation. Audio status values 13h and 14h return information on previous audio operations; they are returned only once after the condition has occurred. If another audio play operation is not requested, the

audio status returned for subsequent READ SUB- CHANNEL commands is 15h.

Table 254 - Audio status codes

Status	Description
00h	Audio status byte not supported or not valid
11h	Audio play operation in progress
12h	Audio play operation paused
13h	Audio play operation successfully completed
14h	Audio play operation stopped due to error
15h	No current audio status to return

The sub-channel data length specifies the length in bytes of the following sub-channel data block. A sub-channel data length of zero indicates that no sub-channel data block is included in the returned data.

NOTE 180 Usual values for sub-channel data length are 0, 12, 20, 28 and 44 bytes. Sub-channel data length does not include the sub channel header.

The sub-Q channel data block consists of control data (bytes 4-5), current position data (bytes 6 - 15) and identification data (bytes 16 - 47). The control data and current position data is obtained from the Q sub-channel information of the current block. Identification data may be reported that was obtained from a previous block. If identification data is reported, the data shall be valid for the sector addressed by the current position data. a) If an audio play operation is proceeding in the background, position data for the last sector played shall be reported. b) In other cases, for instance after a READ command, the target may either report position data for the last sector processed for that operation or may report position data from the sector at the current read head position.

NOTE 181 When the type of information encoded in the Q sub-channel of the current sector is the media catalog number or ISRC; the track, index, and address fields should be extrapolated from the previous sector.

The ADR field gives the type of information encoded in the Q sub-channel of this block, as shown in table 255.

Table 255 - ADR sub-channel Q field

ADR code	Description
0h	Sub-channel Q mode information not supplied
1h	Sub-channel Q encodes current position data (i.e. track, index, absolute address, relative address)
2h	Sub-channel Q encodes media catalogue number
3h	Sub-channel Q encodes ISRC
4h - Fh	Reserved



The control bits are defined in table 256.

Table 256 - Sub-channel Q control bits

+-----+-----+-----+-----+			
Bit		Equals zero	Equals one
+-----+-----+-----+-----+			
0	Audio without pre-emphasis		Audio with pre-emphasis
1	Digital copy prohibited		Digital copy permitted
2	Audio track		Data track
3	Two-channel audio		Four-channel audio
+-----+-----+-----+-----+			

The track number specifies the current track number.

The index number specifies the index number in the current track.

The absolute CD-ROM address field gives the current location relative to the logical beginning of the media. If the MSF bit is zero, this field is a logical block address. If the MSF bit is one, this field is an absolute MSF address. (See table 237)

The track relative CD-ROM address field gives the current location relative to the logical beginning of the current track. If the MSF bit is zero, this field is a track relative logical block address. (If the current block is in the pre-gap area of a track, this will be a negative value, expressed as a two's-complement number. See 14.1.5). If the MSF bit is one, this field is the relative MSF address from the Q sub-channel.

A media catalogue valid (MCVal) bit of one indicates that the media catalogue number field is valid. A MCVal bit of zero indicates that the media catalogue number field is not valid.

The media catalogue number field contains the identifying number of this media according to the uniform product code values (UPC/EAN bar coding) expressed in ASCII. Non-zero values in this field are controlled by the Uniform Product Code Council 1) and the European Article Number Council 2). A value in this field of all ASCII zeros indicates that the media catalog number is not supplied.

The track code valid (TCVal) bit of one indicates that the track ISRC field is valid. A TCVal bit of zero indicates that the track international standard recording code (ISRC) field is not valid.

The track ISRC field contains the identifying number of this media according to the ISRC standards (DIN-31-621) expressed in ASCII. ---- 1) The Uniform Product Code Council is located at 8163 Old Yankee Road, Suite J, Dayton, Ohio 45459. 2) The European Article Number Council is located at Rue des Colonies, 54-BTE8, 1000 Brussels, Belgium.

## CD-ROM current position data format

Table 257 defines the CD-ROM current position data format.

Table 257 - CD-ROM current position data format

Bit	7	6	5	4	3	2	1	0
Byte								
Sub-channel data header								
0	Reserved							
1	Audio status							
2	(MSB)	Sub-channel data length						---
3								(LSB)
CD-ROM current position data block								
4	Sub channel data format code (01h)							
5	ADR				Control			
6	Track number							
7	Index number							
8	(MSB)	Absolute CD-ROM address						---
11								(LSB)
12	(MSB)	Track relative CD-ROM address						---
15								(LSB)

## Media catalogue number data format

Table 258 defines the media catalogue number data format.

Table 258 – Media catalogue number data format

Bit	7	6	5	4	3	2	1	0
Byte								
Sub-channel data header								
0	Reserved							
1	Audio status							
2	(MSB)	Sub-channel data length						---
3								(LSB)
Media catalogue number data block								
4	Sub channel data format code (02h)							
5	Reserved							
6	Reserved							
7	Reserved							
8	MCVal	Reserved						
9	(MSB)	Media catalogue number (UPC/Bar code)						---
23								(LSB)

If media catalogue number data is found, the MCVal bit is set to one. If MCN data is not detected, the MCVal bit is set to zero to indicate the media catalogue number field is invalid.

NOTE 182 Media catalogue number data returned by this command with sub- channel data format field code 02h may be from any block that has UPC bar code Q sub-channel data. (This code is constant anywhere in every applicable disc.)

## Track international standard recording code data format

Table 259 defines the track international standard recording code data format.

Table 259 - Track international standard recording code data format

Bit	7	6	5	4	3	2	1	0
Byte								
Sub-channel data header								
0	Reserved							
1	Audio status							
2	(MSB)	Sub-channel data length						---
3								(LSB)
Track ISRC data block								
4	Sub channel data format code (03h)							
5	ADR				Control			
6	Track number							
7	Reserved							
8	TCVal	Reserved						
9	(MSB)	Track international standard recording code (ISRC)						---
23								(LSB)

If ISRC data is detected, the TCVal bit is set to one. If ISRC data is not detected, the TCVal bit is set to zero to indicate the ISRC field is invalid.

NOTE 183 Track ISRC data returned by this command with sub-channel data format field 03h may be from any block in the specified track that has ISRC data.

**Request Length:**

\$00000000 - \$00000040 (Bytes)

**Transfer Length:**

\$00000000 - \$00000040 (Bytes)

**Buffer Data Structure:**

See tables 253, 257, 258, 259.

**Errors:**

Good  
Check Condition

**\$8043 - Read TOC;                    ( New for SCSI-2 CD-ROM drives )**  
**[0]**

The READ TOC command (see table 260) requests that the target transfers data from the table of contents to the initiator.

Table 260 - READ TOC command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (43h)							
1	Logical unit number			Reserved			MSF	Reserved
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Starting track							
7	(MSB)	Allocation length						---
8								(LSB)
9	Control							

See table 237 for a description of the MSF bit.

The starting track field specifies the starting track number for which the data shall be returned. If this value is zero, the table of contents data shall begin with the first track on the medium. The data are returned in contiguous ascending track number order.

If the starting track field is not valid for the currently installed medium, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

NOTE 184 The maximum TOC data length possible on currently available CD- ROM media is 804 bytes, or 100 TOC track descriptors.  
The format of the data returned is specified in table 261.

Table 261 - READ TOC data format

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB)	TOC data length						(LSB)
1		First track number						
2		Last track number						
3		TOC track descriptor(s)						
4		Reserved						
5		ADR			Control			
6		Track number						
7		Reserved						
8	(MSB)	Absolute CD-ROM address						(LSB)

The TOC data block contains a four-byte header followed by zero or more TOC track descriptors.

The TOC data length specifies the length in bytes of the following TOC data that is available to be transferred during the DATA IN phase. The TOC data length value does not include the TOC data length field itself.

The first track number field indicates the first track number in the table of contents.

The last track number field indicates the last track number in the table of contents before the lead-out track number.

NOTE 185 The first track number is not required to be one. A disc may start at any valid track number. The track numbers between the first track number and the last track number are required to be in contiguous ascending order, except for the lead-out track.

The ADR field gives the type of information encoded in the Q sub- channel of the block where this TOC entry was found. The possible ADR values are defined in table 255.

The control field indicates the attributes of the track. The possible control field values are defined in table 256.

The track number field indicates the track number for which the data in the TOC track descriptor is valid. A track number of 0AAh indicates that the track descriptor is for the start of the lead-out area.

The absolute CD-ROM address contains the address of the first block with user information for that track number as read from the table of contents. An MSF bit of zero indicates that the absolute CD-ROM address field contains a

logical block address. An MSF bit of one indicates the absolute CD-ROM address field contains an MSF address (see table 237).

NOTE 186 The starting logical block address value recovered from the TOC has a tolerance of zero for data tracks and plus or minus 75 CD sectors for audio tracks. This tolerance is multiplied by a factor dependent on the logical block length.

**Request Length:**

\$00000000 - \$00000400 (Bytes)

**Transfer Length:**

\$00000000 - \$00000400 (Bytes)

**Buffer Data Structure:**

See table 261.

**Errors:**

Good  
Check Condition

## \$8044 - Read Header; ( New for SCSI-2 CD-ROM drives )

### [0]

The READ HEADER command (see table 248) requests that the device return the CD-ROM data block address header of the requested logical block.

Table 248 - READ HEADER command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (44h)							
1	Logical unit number			Reserved			MSF	Reserved
2	(MSB)							
3	Logical block address							
4								
5	(LSB)							
6	Reserved							
7	(MSB)							
8	Allocation length							
9	(LSB)							
9	Control							

See table 237 for a description of the MSF bit.

The logical block address field specifies the logical block at which the read header operation shall begin.

See the READ command for exception handling. If the logical block size is other than the physical block size, it shall be mapped into the appropriate physical block from which the data would have been read.

The READ HEADER data format (see table 249) defines the CD-ROM data block address header of the requested logical block.

Table 249 - READ HEADER data format

Bit	7	6	5	4	3	2	1	0
Byte								
0	CD-ROM data mode							
1	Reserved							
2	Reserved							
3	Reserved							
4	(MSB)							
7	Absolute CD-ROM address							
	(LSB)							



The CD-ROM data mode field specifies the CD-ROM data mode of the logical blocks in this sector of data. The values in this field are defined in table 250.

Table 250 - CD-ROM data mode codes

+=====+			
CD-ROM mode	User data field contents (2 048 bytes)	Auxiliary field contents (288 bytes)	
+-----+			
00h	All bytes zero	All bytes zero	
01h	User data	L-EC symbols	
02h	User data	User data	
03h - FFh	Reserved	Reserved	
+=====+			

If the MSF bit is zero, the absolute address field gives the logical block address of the first logical block in the physical sector where the data for the requested logical block address is found. If the MSF bit is one, the absolute address field gives the MSF address of the sector where the data for the requested logical block address is found. (See 14.1.5.)

**Request Length:**

\$00000000 - \$00000008 (Bytes)

**Transfer Length:**

\$00000000 - \$00000008 (Bytes)

**Buffer Data Structure:**

See table 249.

**Errors:**

Good  
Check Condition

**\$805A - Mode Sense;            ( All Devices )**  
**[0]**

This command is complimentary to the Mode Select Command. It is used to get the Medium, Logical Unit, or Peripheral Device parameters from the target device.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$5A )
	\$01	Page Format (PF)	( %000x0000 )
	\$02	Page Control	( %xx000000 )
		Page Code	( %00xxxxxx )
	\$03 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000000 - \$0000ffff (Bytes)

**Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

**Buffer Data Structure:**

Refer to vendor and ANSI® document for further details and data descriptions.

**Errors:**

Good

**\$805F - Read Log;                      ( Sequential Access Devices )**  
**[0]                                      See \$801F command**

The Read Log command is issued to sequential access devices to request statistical information maintained by the device. This is an optional command and the data is returned in the same format as the Mode Select/Sense page codes.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$5F )
	\$01	Page Format (PF)	( %000x0000 )
		No Log Save (NLS)	( %000000x0 )
		No Log Clear (NLC)	( %0000000x )
	\$02	Page Code	( %00xxxxxx )
	\$03 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000000 - \$0000ffff (Bytes)

**Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

**Buffer Data Structure:**

The data is returned as vendor unique as well as Device or Medium specific

**Errors:**

Good  
 Check Condition

**\$80A8 - Read; (Optical Media)**  
**[M]**

This call is mostly a duplication of the \$8028 command except for larger transfer lengths. Please refer to the \$8028 command documentation.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$A8 )
	\$01	DPO	( %000x0000 )
		FUA	( %0000x000 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»»LSB )
	\$06 - \$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

**Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Buffer Data Structure:**

Data

**Errors:**

Good  
Check Condition  
Reservation Conflict.

## **\$80AD - Read Update Block; (Optical Media)**

**[0]**

This call is mostly a duplication of the \$802D command except for larger transfer lengths. Please refer to the \$802D command for field definitions.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$AD )
	\$01	DPO	( %000x0000 )
		FUA	( %0000x000 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06	Latest	( %x0000000 )
		Generation Addr (MSB)	( %0xxxxxxx )
	\$07	Generation Addr (LSB)	
	\$08 - \$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Buffer Data Structure:**

Data

### **Errors:**

Good  
Check Condition  
Reservation Conflict

**\$80B7 - Read Defect Data; (Optical Media)**  
**[0]**

This call is mostly a duplication of the \$8037 command except for larger transfer lengths. The Read Defect Data command requests that the target transfer the medium defect data to the initiator.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$B7 )
	\$01	PList	( %000x0000 )
		GList	( %0000x000 )
		Defect List Format	( %00000xxx )
	\$02 - \$05	Reserved	
	\$06 - \$09	Allocation Length	( MSB»»»LSB )
	\$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )

**Request Length:**

\$00000000 - \$ffffffff (Bytes)

**Transfer Length:**

\$00000000 - \$ffffffff (Bytes)

**Buffer Data Structure:**

Defect Data. This contains an eight byte header followed by zero or more defect descriptors. See ANSI® and vendor documentation for details of this data.

**Errors:**

Good  
Check Condition

## **\$80C1 - Read TOC; (Ruby Drive)**

This command requests that the target device transfer the the TOC (Table of Contents) to the host computer. Details of this call are given in the '**Apple CD ROM SCSI Command Set Rev. 1.2**'.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C1 )
	\$01	SCSI Command Flags	( \$00 )
	\$02	Track Number	(TNO in BCD)
	\$03 - \$04	Reserved	
	\$05	TOC Type	( %xx000000 )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$0000ffff (Bytes)

### **Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

### **Buffer Data Structure:**

See the '**Apple CD ROM SCSI Command Set Rev. 1.2**'

### **Errors:**

Good  
Check Condition

### **\$80C2 - Read Q Subcode; (Ruby Drive)**

This call causes the target device to send the Q Subcode data to the host computer. Details of this call are given in the '**Apple CD ROM SCSI Command Set Rev. 1.2**'.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C2 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$000000ff (Bytes)

#### **Transfer Length:**

\$00000000 - \$000000ff (Bytes)

#### **Buffer Data Structure:**

See the '**Apple CD ROM SCSI Command Set Rev. 1.2**'

#### **Errors:**

Good  
Check Condition



### **\$80C3 - Read Header; (Ruby Drive)**

This to the host computer four bytes of header information for the specified logical block address. This call is similar to the Read Extended (\$8028) command, but only the header bytes are returned by the target. Details of this call are given in the '**Apple CD ROM SCSI Command Set Rev. 1.2**'.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C3 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$05	Logical Block	( MSB »»» LSB )
	\$06 - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$000000ff (Bytes)

#### **Transfer Length:**

\$00000000 - \$000000ff (Bytes)

#### **Buffer Data Structure:**

See the '**Apple CD ROM SCSI Command Set Rev. 1.2**'

#### **Errors:**

Good  
Check Condition

**\$80CC - Audio Status; (Ruby Drive)**

Use of this command causes the target device to transmit the current audio play status and the starting Q Subcode Address of the next track. Details of this call are given in the '**Apple CD ROM SCSI Command Set Rev. 1.2**'.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$CC )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$000000ff (Bytes)

**Transfer Length:**

\$00000000 - \$000000ff (Bytes)

**Buffer Data Structure:**

See the '**Apple CD ROM SCSI Command Set Rev. 1.2**'

**Errors:**

Good  
Check Condition

## \$81C2 - Audio Status; (Ruby Drive)

Use of this command causes the target device to transmit the current audio play status and the starting Q Subcode Address of the next track. Details of this call are given in the '**Apple CDSC+ ROM Reference**'.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C2 )
	\$01	Status Mode	( %0000xxxx )

The Status Mode field is defined as follows:

Code	Returned Data
0h	return 6 bytes of audio status data as defined in current CDSC SCSI specification.
1h	return 4 bytes of audio volume control data.
2h-Fh	reserved

\$02 - \$0B Reserved.

### Request Length:

\$00000000 - \$00000006 (Bytes)

### Transfer Length:

\$00000000 - \$00000006 (Bytes)

### Buffer Data Structure:

Audio Volume Control Data (Status Mode =1h)									
Bit	7	6	5	4	3	2	1	0	
Byte									
0	L-Channel Volume Control								
1	R-Channel Volume Control								
2	Reserved								
3	Reserved								

See the '**Apple CDSC+ ROM Reference**' for the Description of Status Mode 0

### Errors:

Good  
Check Condition

## **\$80C3 - Read Header; (Ruby Drive)**

This transfers to the host computer four bytes of header information for the specified logical block address. This call is similar to the Read Extended (\$8028) command, but only the header bytes are returned by the target. Details of this call are given in the '**Apple CD ROM SCSI Command Set Rev. 1.2**'.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C3 )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$05	Logical Block	( MSB »» LSB )
	\$06 - \$0B	Reserved.	

### **Request Length:**

\$00000000 - \$000000ff (Bytes)

### **Transfer Length:**

\$00000000 - \$000000ff (Bytes)

### **Buffer Data Structure:**

See the '**Apple CD ROM SCSI Command Set Rev. 1.2**'

### **Errors:**

Good  
Check Condition

## **Control Call**

Call Parameters : Device Number        ≠        \$0000  
                  Call Number        =        \$0006  
                  Control List Pointer  
                  Request Count  
                  Transfer Count  
                  Control Code  
                  DIB Pointer

Device Number:        *This word parameter specifies the target device.  
                         This parameter must be nonzero.*

Call Number:         *This word parameter specifies the type of call.*

Control List Ptr: *This longword points to the control list location.*

Request Count:       *This longword parameter indicates the number of bytes  
                         to be transferred. If the request count is smaller  
                         than the minimum buffer size required by the call, an  
                         error is returned.*

Transfer Count:       *This longword returned by the call indicates the  
                         number of bytes actually transferred.*

Control Code:         *This word parameter specifies the type of control  
                         request. Control codes of \$0000 through \$7FFF are  
                         standard control calls that must be supported by  
                         device drivers. SCSI specific control calls use  
                         control codes in the range of \$8000 through \$FFFF. A  
                         list of standard control calls follows:*

\$0000	Reset Device
\$0001	Format Device
\$0002	Eject
\$0003	Set Configuration Parameters
\$0004	Set Wait/No Wait Mode
\$0005	Set Format Options
\$0006	Assign Partition Owner
\$0007	Arm Event
\$0008	Disarm Event
\$0009	Set Partition Map
\$000A - \$7FFF	Reserved - These control codes to be assigned by Apple Computer, Inc.

*The list of device specific status calls are as  
follows:*

\$8000 - \$80FF Device specific SCSI commands.

*The following are the SCSI specific calls. The values to the right are the Devices that use that code and are defined at the end of the list.*

**Group 0 Commands**

\$8001	ReZero	1, 5, 6, 8, B, E
	Rewind Unit	2
	Reset Printer	D
\$8002	Down Load Code	D
\$8004	Format Unit (Non-Printers)	1, 8
\$8004	Format Unit (Printers)	3, D, E
\$8005	Draw Bits	D
	Send QIC-100 System Data	E
\$8006	Clear Bits	D
\$8007	Reassign Blocks	1, 5, 8, E
\$8009	Verify Unit	E
\$800A	Write	1, 2, 5, 8, E
	Print	3, D
	Send	4
	Send Message	A
\$800B	Seek	1, 5, 6, 8, B, E
	Track Select	2
	Slew and Print	3
\$800C	Reserved	
\$800F	Write Controller Information	E
\$8010	Write File Marks	2
	Flush Buffer	3
	Drive Pass-Thru	E
\$8011	Space	2
\$8013	Verify	2
\$8014	Write QIC-100 Information	E
\$8015	Mode Select	1, 2, 3, 5, 6, 7, 8, A, B, C, D, E
\$8016	Reserve Unit	1, 2, 3, 5, 6, 7, 8, B, C, D, E
\$8017	Release Unit	1, 2, 3, 5, 6, 7, 8, B, C, D, E
\$8018	Copy	0
\$8019	Erase	2
\$801B	Start/Stop Unit	1, 5, 6, 8, B
	Load/Unload	2, E
	Stop Print	3
\$801B	Scan	7, C
\$801D	Send Diagnostic	0, A, B, C, E
\$801E	Prevent/Allow Medium Removal	1, 2, 5, 6, 8, B

### Group 1 Commands

\$8020 - \$8023	Reserved	
\$8024	Define Window Parameters	7,C
\$8026 - \$8027	Reserved	
\$802A	Write	1,5,8,E
	Send	7,C
\$802B	Seek	1,5,6,8,B,E
	Locate	2
\$802C	Erase	8
	???????Read Generation????????	8
\$802E	Write and Verify	1,5,8
\$802F	Verify	1,5,6,8,B
\$8030	Search Data High (Unsupported)	1,5,6,8
\$8031	Search Data Equal (Unsupported)	1,5,6,8
	Medium Position	7
\$8032	Search Data Low (Unsupported)	1,5,6,8
\$8033	Set Limits	1,5,6,8
\$8034	Pre_Fetch	1,8
\$8035	Synchronize Cache	1,8
\$8036	Lock/Unlock Cache	1,8
\$8038	Media Scan	8
\$8039	Compare	1,2,5,6,7,8
\$803A	Copy and Verify	1,2,5,6,7,8
\$803B	Write Buffer	1,2,7,8,A,B,E
\$803D	Update Block	8
\$803F	Write Long	1

### Group 2 Commands

\$8040	Change Definition (Unsupported)	0
\$8041 - \$8044	Reserved	
\$8045	Play Audio(10)	
\$8047	Play Audio MSF	
\$8048	Play Audio Track Index	
\$8049	Play Track Relative(10)	
\$804B	Pause/Resume	
\$804C - \$8054	Reserved	
\$8055	Mode Select	0
\$8056 - \$805F	Reserved	

### Group 3 Commands

\$8060 - \$807F Reserved

### Group 4 Commands

\$8080 - \$809F Reserved

#### Group 5 Commands

\$80A0 - \$80A4	Reserved	
\$80A5	Move Medium	9
\$80A5	Play Audio(12)	
\$80A6	Exchange Medium	9
\$80A7	Reserved	
\$80A9	Play Track Relative(12)	
\$80AA	Write	8
\$80AB	Reserved	
\$80AC	Erase	8
\$80AE	Write and Verify	8
\$80AF	Verify	8
\$80B0	Search Data High	(Unsupported) 8
\$80B1	Search Data Equal	(Unsupported) 8
\$80B2	Search Data Low	(Unsupported) 8
\$80B3	Set Limits	8
\$80B4 - \$80B6	Reserved	
\$80B8 - \$80BC	Reserved	
\$80BD	Update Blocks	8
\$80BE - \$80BF	Reserved	

#### Group 6 Commands

\$80C0	Eject Disk	B
\$80C4 - \$80C7	Reserved	
\$80C8	Audio Track Search	B
\$80C9	Audio Play	B
\$80CA	Audio Pause	B
\$80CB	Audio Stop	B
\$80CD	Audio Scan	B
\$80CE - \$80DF	Reserved	



## Group 7 Commands

\$80E0 - \$80FF Reserved

### Device Definitions

0	=	Devices '1' through '9'
1	=	Direct-Access Devices
2	=	Sequential-Access Devices
3	=	Printer Devices
4	=	Processor Devices
5	=	Write-Once Read-Multiple Devices
6	=	Read-Only Direct-Access Devices
7	=	Scanner Devices
8	=	Optical Memory Devices
9	=	Changer Devices
A	=	Communications Devices
B	=	Apple CD_ROM Drive
C	=	Apple SCSI Scanner
D	=	Apple LaserWriter SC
E	=	Apple Tape Drive

## Non-SCSI Commands

\$8100 - \$FFFF Reserved

DIB Pointer:        *This longword points to the DIB for the target device.*

This call is used to send control information to the device or device itself and may be used to issue any extended SCSI Command that sends data to the device through the use of device specific control codes. The device driver is responsible for validating the status code prior to executing the requested command. The following 65816 code sample shows how this might be done.

```

validate_ctrl_code    lda    control_code        ;Get control Code
                     bmi    device_specific      ;Is it Device Specific?
                     cmp    #max_command+1      ;No. Is it out of range?
                     blt    do_standard         ;No. Goto code segment
bad_code_exit         lda    #BAD_CODE          ;Central BAD CODE Exit
                     sec
                     rtl

device_specific        and    #$00ff             ;Is it ≤ the max SCSI
                     pha                     ;command for this
                     inc                     ;device?
                     ldy    #SCSI_maxcmd_offset
                     cmp    [DIB_PTR],y
                     blt    so_far_so_good      ;It's in a good range
                     pla                     ;Restore stack
                     bra    bad_code_exit       ;exit with error

```

```

so_far_so_good    lda    1,s                ;Get control code from stack
                  lsr                    ;Generate a group index
                  lsr                    ;to use to get to the
                  lsr                    ;correct group offset
                  and    #$000e
                  tay
                  lda    group_offset,y
                  sta    temp                ;Save offset for later

                  lda    1,s                ;Get control code from stack
                  and    #$0010            ;Upper or lower half
                  beq    first_16_bits      ;of group bitmap?
                  inc    temp              ;Upper half. Adjust
                  inc    temp              ;offset by two

first_16_bits     pla                    ;Last time. Get control code
                  and    #$000f            ;Retain low nibble
                  asl    a                ;Account for 16 bit
                  tay                    ;table of offsets.
                  lda    cmd_bm_mask,y
                  ldy    temp
                  and    [DIB_PTR],y      ;Is bit set in bitmap?
                  beq    bad_code_exit     ;No. Error exit.
                  .
                  .
                  .

max_command       equ    $0009            ;Max standard control code

SCSI_maxcmd_offset equ    $0042

group_offset      dc    i2'GROUP0-DIB_start' ;Offsets from start
                  dc    i2'GROUP1-DIB_start' ;of DIB to Group
                  dc    i2'GROUP2-DIB_start' ;bitmaps
                  dc    i2'GROUP3-DIB_start'
                  dc    i2'GROUP4-DIB_start'
                  dc    i2'GROUP5-DIB_start'
                  dc    i2'GROUP6-DIB_start'
                  dc    i2'GROUP7-DIB_start'

temp              ds    2

cmd_bm_mask       dc    i2'%1000000000000000' ;Bitmap masks.
                  dc    i2'%0100000000000000'
                  dc    i2'%0010000000000000'
                  dc    i2'%0001000000000000'
                  dc    i2'%0000100000000000'
                  dc    i2'%0000010000000000'
                  dc    i2'%0000001000000000'
                  dc    i2'%0000000100000000'
                  dc    i2'%0000000010000000'
                  dc    i2'%0000000001000000'
                  dc    i2'%0000000000100000'
                  dc    i2'%0000000000010000'
                  dc    i2'%0000000000001000'
                  dc    i2'%0000000000000100'
                  dc    i2'%0000000000000010'
                  dc    i2'%0000000000000001'

```

If an invalid control code is passed to the driver, the driver will return a 'BAD CODE' error. The driver may also wish to identify where the call came from and shield certain calls from the application. Based on this, the driver could use a secondary command bitmap to validate codes allowed from the application. If an invalid control list length is passed to the driver, the driver should return a 'BAD PARAMETER' error. The device driver sets the transfer count to the number of bytes processed as a result of the control call.

**NOTE:** Both standard and device specific status calls may detect an OFFLINE or DISKSWITCH status. If either of these conditions are detected then a **SET\_DISKSW** call is issued to set the device dispatcher maintained disk switched error.

### **\$0000 - Reset Device**

This control call is used to reset a particular device to it's default settings. A device driver should configure itself based on the contents of the driver's configuration parameter list (not to be confused with control list). This call should also return any media variables modified through a Set\_Format\_Options call to the default settings.

Control List:	Word	Length of control list (\$0000)
---------------	------	---------------------------------

### **\$0001 - Format Device**

This control call is used to format the media used by a block device. This call is not linked to any particular file system. It simply prepares all blocks on the media for reading and writing. After completion of formatting a block device, any media variables that may have been modified by a Set\_Format\_Options control call should be returned to their default value. Character devices do not support this function and will return with no error.

Due to Partitioning, this command will check the device for a valid partition map. If none exists, then a format call will be sent to the device. If on the other hand a partition map does exist then this call will result in no error with no action taken on the device. If a hard format is desired, then an \$8004 Format Unit command should be used to force the Format command being issued.

Control List:	Word	Length of control list (\$0000)
---------------	------	---------------------------------

## **\$0002 - Eject**

This control call is used to physically or logically eject the media from a block device. The targeted device will be sent an SCSI Unload/Unlock command to enable the ejection of the media and then physically ejected if the device supports the eject call. If not, then the device will be marked offline and it will be up to the user to then remove the media. If the device is linked to other devices, it will be marked offline and physical ejection will occur when all the linked DIBs for that device are marked offline. Character devices that do not support this control call, will return with no error.

Control List:	Word	Length of control list (\$0000)
---------------	------	---------------------------------

Note for SCSI-2 : the CD-ROM driver now sends two eject calls to the SCSI target to have the inserted media removed. The first one is the standard START STOP UNIT command (as used in the other Apple IIgs SCSI drivers). If the command fails, the EJECT DISK command (see the description of the \$80C0 control subcall later in the manual) is sent to the target. That way, all media are ejected whatever the SCSI version.

## **\$0003 - Set Configuration Parameters**

This control call is used to send device specific configuration parameters to a device. The first word in the control list indicates the length of the configuration parameter list in bytes. The configuration parameters should be placed into the configuration list contiguous to the byte count. The structure of the configuration parameter list is device dependent. The first word of the new configuration list must be equal to the request count. Additionally, the first word of the new configuration list must equal the first word of the existing configuration list. It is not legal to set a new configuration list of a different size of the existing list. If this call is made with an erroneous configuration list length a bad parameter error will be returned.

Control List:	Word	Length of configuration parameter list
	Data	Configuration Parameter List Data

## **\$0004 - Wait/No Wait Mode**

This call is used to set a character device to Wait or No Wait mode. If in wait mode, the device will wait for the number of characters specified in the request count of a read call before returning from the read. If in No Wait mode, a read call will return immediately with a transfer count indicating the number of characters returned. If a character was available the transfer count will return from a read with a non zero value. If a character was not available the transfer count will return from a read call with a value of zero. The control list will contain a word with a value of \$0000 to set

Wait Mode or a value of \$8000 to set No Wait mode. Block devices do not support this control call and will return with no error.

Control List:                      Word                      Wait/No Wait Mode

### **\$0005 - Set Format Options**

This call sets media specific parameters prior to executing a format call to a block device. This call does not imply a format. The control list consists of a word (Format\_RefNum) and a word (Interleave\_Factor). The format reference number specifies a group of variables used during a subsequent format call which includes Format Environment, Block Count, Block Size and Interleave Factor. If the Interleave\_Factor is set to NIL then the default interleave specified in the format variables list is used. In order to obtain a list of Format\_RefNum values and their corresponding variables, a Get\_Format\_Options status call is issued to the device. After the appropriate format variables are selected, a Set Format\_Options control call is issued followed by a format control call. This call is not supported by character devices and should return a 'BAD\_COMMAND' error.

Control List:              Word              Format\_RefNum  
                            Word              Interleave\_Factor if used.

**NOTE: This call must be issued to the first DIB in the link if the device is partitioned. If this is not the case an Invalid Device Number error will be returned.**

### **\$0006 - Assign Partition Owner**

This call is supported by block devices supporting partitioned media. This call is executed by an FST as a result of the 'ERASE\_DISK' system call. The control list consists of a class 1 string indicating the partition type. Partition types can be up to 32 bytes in length. If the string is less than 32 bytes in length, it must be terminated with a NUL. A partition type can be cleared by setting the first byte of the string to the NUL character. Upper and lower case characters are considered equivalent. The driver will then reassign the current partition to the new owner. This call does not reassign physical block allocation within a device partition descriptor. Block devices utilizing non-partitioned media and character devices should return with no error.

Control List:              String              Class 1 string specifying partition type

Example String

\$0d \$00 \$41 \$70 \$70 \$6c \$65 \$5f \$50 \$52 \$4f \$44 \$4f \$53 \$00

Example Types

Apple\_MFS  
Apple\_HFS  
Apple\_Unix\_SVR2  
Apple\_partition\_map  
Apple\_Driver  
Apple\_PRODOS  
Apple\_Free  
Apple\_Scratch

## \$0007 - Arm Signal

This call provides a means for a device driver to install a signal handler into the GS/OS signal manager's handler list. Signal code is an arbitrary value assigned by the driver to identify the signals that it generates. The driver will maintain a list of armed signals. When an interrupt associated with the driver occurs, the driver's interrupt handler will assess the priority of the interrupt and pass the appropriate signal handler's address to the signal mechanism. Each signal should have a unique signal code. If an attempt is made to arm a signal for which that signal code already exists in the drivers signal list, a driver\_bad\_parameter error will be returned and the signal will not be added to the drivers signal handler list. Priority is the signal priority, with \$0000 being the lowest priority and \$FFFF being the highest priority. Handler address is the address where the signal handler resides.

Control List:	Word	Signal Code
	Word	Priority
	Longword	Signal Handler Address

**NOTE:** This call is not supported by the drivers that follow the layout defined by this spec. The Application should use async and data chaining coding techniques to fully benefit from the designs speed enhancements.

## \$0008 - Disarm Signal

This call provides a means for a device driver to remove it's interrupt handler into the GS/OS signal manager's handler list. Signal code is an arbitrary value assigned by the driver when the signal was armed.

Control List:	Word	Signal Code
---------------	------	-------------

**NOTE:** This call is not supported by the drivers that follow the layout defined by this spec. The Application should use async and data

chaining coding techniques to fully benefit from the designs speed enhancements.

## \$0009 - Set Partition Map

This call writes the partition map to the device specified which must be the first device of any linked list. If not an **Invalid Device Number** error is returned.

The structure of the partition map entries are described in detail in '**Inside Macintosh™** Volume V' under the SCSI Manager section and is shown below. It must be noted that all fields, except strings, in the partition map are stored High Byte »» Low Byte. Example: Even though the partition signature is defined as \$504d, it will appear to the 65xxx family of processors as \$4d50.

byte 0	pmSig (word)	always \$504d
2	pmSigPad (word)	reserved for future use
4	pmMapBlkCnt (long word)	number of blocks in map
8	pmPyPartStart (long word)	first physical block of partition
C	pmPartBlkCnt (long word)	number of blocks in partition
10	pmPartName (32 bytes)	partition name
30	pmPartType (32 bytes)	partition type
50	pmLgDataStart (long word)	first logical block of data area
54	pmDataCnt (long word)	number of blocks in data area
58	pmPartStatus (long word)	partition status information
5C	pmLgBootStart (long word)	first logical block of boot code
60	pmBootSize (long word)	size in bytes of boot code
64	pmBootLoad (long word)	boot code load address
68	pmBootLoad2 (long word)	additional boot load information
6C	pmBootEntry (long word)	boot code entry point
70	pmBootEntry2 (long word)	additional boot entry information
74	pmBootCksum (long word)	boot code checksum
78	pmProcessor (16 bytes)	processor type
88	( 128 bytes )	boot specific arguments

The size of the partition map can vary in size<sup>3</sup> but must be written all at once. The driver validates the request count against the pmMapBlkCnt using the block size for that device. If pmMapBlkCnt \* block size ≠ Request Count then an **Invalid Byte Count** error is returned. If the application wishes to reassign an entry to a different type, the \$0006 control call Assign Partition Owner should be used.

<sup>3</sup> The partition map can contain as few as 2 (Partition map plus the partition itself) or as many as 32 entries maximum.

When the driver receives this call it must do several layers of verification before it can be considered successfully completed. First the driver attempts to ensure that there is enough space to allocate DIBs for these partitions.

**Example:** The device had four partitions, each with it's own DIB. The new map has nine entries. The driver must allocate memory for five additional DIBs. If this can not be accomplished then a **Resource Not Available** error will be returned. Next the driver issues a **Post Driver Install** to see if the device table has room for the additional devices. If not, the memory allocated will be released and the appropriate error will be returned. Finally, the driver will write the partition map to the device.

The driver treats the successful completion of this call as if the media is removable and will declare to the device that a disk switched condition exists. No error will be returned.

### **Device Specific Control Calls**

The following calls are the device specific control calls allowed by this driver. For a discussion on how to issue these calls, please refer to the Device Specific Status calls.



**\$8001 - ReZero Unit; ( Direct Access Devices )**  
**[0]**

The ReZero Unit command instructs the target device to place itself in a specific state. This is an optional command to several of the block device classes. Refer to the vendor specifications for details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$01 )
	\$01 - \$03	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

# **\$8001 - Rewind Unit; ( Sequential Access Devices )** **[M]**

This is the equivalent to the \$8001 ReZero Unit call only this is mandatory to Sequential access devices. The command requests that the target rewind the logical unit to the beginning-of-medium or load point. Prior to execution of this command, the target shall write any data currently in the buffer to the media.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$01 )
	\$01	Immed	( %0000000x )

If set to one, the status will be returned as soon as the operation has been initiated. Zero indicates that the status is returned after the operation is complete.

**Note:** Prior to issuing a Rewind command with the Immed bit set to one, it is suggested that the Write Filemarks command with the Immed bit set to zero be used to flush the devices buffer to the media.

\$02 - \$04	Reserved	
\$05	Vendor Unique	( %xx000000 )
	Reserved	( %00xxxxxxx )
\$06 - \$0B	Reserved	

## **Request Length:**

Unused

## **Transfer Length:**

Unused

## **Buffer Data Structure:**

None

## **Errors:**

Good  
 Check Condition

## **\$8002 - Down Load Code; ( Apple LaserWriter SC )**

This command is specific to the LaserWriter SC and provides a method for the host computer to download 68000 code to the LaserWriter SC in order to correct errors in the firmware or to expand it's functionality.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$02 )
	\$01 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00ffffff (Bytes)

This specifies the number of bytes to send. If the LaserWriter SC lacks sufficient available memory, the command is terminated with a Check Condition status, the ILI bit of the Sense Data set to one, and the Information Byte of the Sense Data set to the difference between the request length and the amount of available memory. The Request Length must be an EVEN number (the 68000 requires word alignment). The last word of the data sent must be a two's compliment checksum of all the previous words of data. If the checksum doesn't match, a Check Condition status is returned and the Sense Key set to Aborted Command. After the code is downloaded and checksum verified, execution begins at the first byte of the code.

**NOTE:** The download command is meant for advanced programmers with knowledge of the internals of the LaserWriter SC firmware.

### **Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

### **Buffer Data Structure:**

Code.

### **Errors:**

Good  
Check Condition

**\$8004 - Format Unit; ( Direct Access Devices )**  
**[M]**

The format command instructs the target device to format the currently mounted media so that all host addressable logical blocks can be accessed. The media may also be certified at this time and various levels of defects reported. There is no guarantee that the media has or has not been altered. For further details pertaining to the levels of defect management, please refer to the vendor specification as well as the ANSI® x3.131-1986 document.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$04 )
	\$01	FmtData (format data) bit	( %000x0000 )
		CmpLst (complete list) bit	( %0000x000 )
		Defect List Format	( %00000xxx )
	\$02	Vendor Unique	
	\$03 - \$04	Interleave	(MSB»»LSB)
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

See spec for defect data types and structures.

**Errors:**

Good  
 Check Condition

## **\$8004 - Format Unit; ( Printer devices )**

The format command to printers provides a means for the host computer to specify forms or fonts to printers that support programmable forms or fonts. The format information sent is vendor unique since it is peripheral-device specific, For further details please refer to the vendor specification as well as the ANSI® x3.131-1986 document.

**NOTE:** The LaserWriter SC printer does not follow the ANSI® Spec for this call. It is important that the programmer refer to the spec for this device. If not used correctly, this command can cause imaging to occur off the paper and will reduce the life of the Canon Engine.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$04 )
	\$01	FmtData (format data) bit	( %000000xx )
		00	= Set Form
		01	= Set Font
		10	= Vendor Unique
		11	= Reserved
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00ffffff (Bytes)

### **Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

### **Buffer Data Structure:**

See vendor spec for the details of this command.

### **Errors:**

Good  
Check Condition

## **\$8005 - Send QIC-100 System Data; ( Apple Tape Drive )**

The Send QIC-100 System Data command allows the host computer to send 128 bytes of QIC-100 System Data to the controller. Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$05 )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00000080 (Bytes)

### **Transfer Length:**

\$00000000 - \$00000080 (Bytes)

### **Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

### **Errors:**

Good  
Check Condition.

## **\$8005 - Draw Bits; ( Apple LaserWriter SC )**

This device specific command applies to only one device, the LaserWriter SC. This command transfers image data into the specified rectangular area of imaging memory using one of several transfer modes. This command is only appropriate for devices with at least one megabyte of RAM installed.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$05 )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00ffffff (Bytes)

**NOTE:** The LaserWriter SC spec states that this value must be \$A (10). The driver will handle this transparent to the caller. The length of the entire structure being sent must be correctly represented here.

### **Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

### **Buffer Data Structure:**

See vendor spec for the details of this command.

### **Errors:**

Good  
Check Condition

## \$8006 - Clear Bits; ( Apple LaserWriter SC )

This is the compliment to the \$8005 Draw Bits command and it applies to only one device, the LaserWriter SC. This command clears image data from the specified rectangular area of imaging memory. This command is only appropriate for devices with at least one megabyte of RAM installed.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$06 )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### Request Length:

\$00000008 (Bytes)

### Transfer Length:

\$00000008 (Bytes)

### Buffer Data Structure:

All pixels within the bounds rectangle only will be cleared. If the bounds rectangle has it's bottom right point higher or more left than it's top left point, the rectangle is assumed to contain no bits and no image data will be cleared. A bounds rectangle outside of the imaging rectangle (Set by the format command) is an error.

Bit Byte	7	6	5	4	3	2	1	0
0	Bounds.topLeft.x (MSB)							
1	Bounds.topLeft.x (LSB)							
2	Bounds.topLeft.y (MSB)							
3	Bounds.topLeft.y (LSB)							
4	Bounds.botRight.x (MSB)							
5	Bounds.botRight.x (LSB)							
6	Bounds.botRight.y (MSB)							
7	Bounds.botRight.y (LSB)							



**Errors:**

Good  
Check Condition

**\$8007 - Reassign Blocks; ( Direct Access Devices )**  
**[0] ( WORM Devices )**

The Reassign Blocks command requests the target to reassign the defective blocks listed in the defect list to an area of the logical unit set aside for this purpose.

A defect list is transferred to the target with this command containing the logical blocks to be reassigned.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$07 )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$0000ffff (Bytes)

This value is to be equal to the number of defective blocks in the list times 4 plus 4. If eight blocks are to be reassigned, then the request length will be 8\*4+4 or 36 (\$24)

**Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

**Buffer Data Structure:**

The defect descriptor specifies a four byte defect logical block address that contains the defect. The defect descriptors are in ascending order.

If the target has insufficient capacity to reassign all of the defective logical blocks, the command shall terminate with a Check Condition status and the sense key shall be set to Medium Error. The logical block address of the first logical block not reassigned is returned in the information bytes of the sense data.

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved							
2	Reserved							
3	Reserved							
4	Defect Logical Block Address (MSB)							
5	Defect Logical Block Address							
6	Defect Logical Block Address							
7	Defect Logical Block Address (LSB)							
n - 3	Defect Logical Block Address (MSB)							
n - 2	Defect Logical Block Address							
n - 1	Defect Logical Block Address							
n	Defect Logical Block Address (LSB)							

**Errors:**

Good  
Check Condition

## **\$8009 - Verify Unit; ( Apple Tape Drive )**

The Verify Unit command causes the controller to perform a verify operation of the format of the tape. This is done by reading every frame on tape and determining whether both CRCs of that frame can be computed correctly. Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$09 )
	\$01	SCSI Command Flags	( %x0000000 )
		Defect List Format	( %000xxxxx )
	\$02	OneTrk	( %x0000000 )
		MfgBlk	( %0x000000 )
		ConCar	( %00x00000 )
		Track Number	( %000xxxxx )
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

### **Errors:**

Good  
Check Condition.

**\$800A - Write; ( Block Devices )**  
**[M]**

This is the standard SCSI Write Command used by most devices that receive data from the caller. If a block address greater than \$ffff is needed, a command \$802A should be used.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0A )
	\$01	Reserved	
	\$02 - \$03	Logical Block Address	( MSB »» LSB )
	\$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition  
 Reservation Conflict

**NOTE:** See also **\$800A - Write, Print, Send and Send Message** described below.

**\$800A - Write; ( Sequential Devices )**  
**[M]**

This is the standard SCSI Write Command used by most devices that receive data from the caller. If a byte count greater than \$ffffff is needed, a command \$802A should be used.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0A )
	\$01	Fixed	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

Data

**Errors:**

Good  
Check Condition  
Reservation Conflict

**NOTE:** See also **\$800A - Write** above and **Print, Send** and **Send Message** described below.

**\$800A - Print; ( Printer Device )**  
**[M]**

This command sends the specified number of bytes to the target device for printing.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0A )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

Data to be sent or printed.

**Errors:**

Good  
Check Condition

**NOTE:** The LaserWriter SC uses the Print command in a slightly different manner. When issued, no data is sent with the command. The LaserWriter SC will print the data that has been sent via the Draw Bits and Clear Bits command along with other data that resides in it's RAM.

**NOTE:** See also **\$800A - Write** and **Write** above and **Send** and **Send Message** described below.

**\$800A - Send; ( Processor Devices )**  
**[M]**

This command sends the requested number of bytes to the target Processor Device.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0A )
	\$01	Async Event (AEN)	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

Data to be sent or printed.

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$800A - Write, Write** and **Print** above and **Send Message** described below.



**\$800A - Send Message; ( Communication Devices )**  
**[M]**

This command transfers the requested number of bytes to the target communication device.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0A )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

Data to be sent or printed.

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$800A - Write, Write, Print, and Send** above.

**\$800B - Seek; ( Block Devices )**  
**[0]**

This command requests that the target position itself so as to be in position to read or write the Logical block specified. If a block address greater than \$ffff is required, then use the \$802B Seek (Extended) command.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0B )
	\$01	Reserved	
	\$02 - \$03	Logical Block Address	( MSB »» LSB )
	\$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

**NOTE:** See also **\$800B - Track Select** and **Slew and Print** described below.

**\$800B - Track Select; ( Sequential Access Devices )**  
**[0]**

The Track Select command instructs the target device to to select the track specified by the Track Value. This is an optional command.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0B )
	\$01 - \$03	Reserved	
	\$04	Track Value	( \$xx )
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$800B - Seek** above and **Slew and Print** described and below.

## **\$800B - Slew and Print; ( Printers )**

### **[0]**

The Slew and Print command transfers the requested number of bytes to the target device to be printed. The data sent is application dependent. This command is provided for printers that do not support forms control information imbedded within the print data.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0B )
	\$01	Channel	( %0000000x )
	\$02	Slew Value	
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

If the Channel bit is zero, the slew value specifies the number of lines that the form shall advance before printing the data. A slew value of \$FF that the form is advanced to the first line of the next form. If the Channel bit is one, the Slew Value indicates the forms control channel number to be advanced to.

#### **Request Length:**

\$00000000 - \$0000ffff (Bytes)

#### **Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

#### **Buffer Data Structure:**

Data to be printed.

#### **Errors:**

Good  
Check Condition

**NOTE:** See also **\$800B - Seek** and **Track Select** described above.

**\$800F - Write Controller Information; ( Apple Tape Drive )**

The Write Controller Information command allows the host to input. Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$0F )
	\$01 - \$0B	Reserved	

**Request Length:**

\$00000006

**Transfer Length:**

\$00000006

**Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

**Errors:**

Good  
Check Condition.

**\$8010 - Write File Marks; ( Sequential Access Devices )**  
**[M]**

This command causes the target device to write out the requested number of file marks to the media at the current position. By specifying zero filemarks it is possible to force the device to flush any data that might still be buffered to the media. See the ANSI® Spec for details about this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$10 )
	\$01	Immed	( %0000000x )
	\$02 - \$04	Number of file marks	(MSB»»LSB)
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$8010 - Flush Buffer** described below.

**\$8010 - Flush Buffer; ( Printers )**  
**[0]**

This command instructs the printer to finish printing any data that remains in it's buffers. This is useful for applications that may wish to ensure that certain housekeeping chores have been completed. See ANSI® Document for more details. This is an optional command for printer devices.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$10 )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

**NOTE:** See also **\$8010 - Write Filemarks** described above.

## **\$8010 - Drive Pass-Thru; ( Apple Tape Drive )**

The Drive Pass-Thru command allows the host to directly access the drive level interface. Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$10 )
	\$01	Reserved	
	\$02	Register 1	
	\$03	Register 2	
	\$04	Reserved	
	\$05	CmdMode	( %x00000000 )
	\$06 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

### **Errors:**

Good  
Check Condition.



## **\$8011 - Space; ( Sequential Access )**

**[M]**

The Space command is currently an optional command for Sequential Access Devices that is headed for the mandatory class under SCSI-2 currently in draft. This command provides several positioning functions that are determined by the code and count. Both forward and backward positioning are possible, although some target devices may not support all of the combinations. These targets will return a Check Condition status and an Illegal Request sense key for any illegal request. Please refer to the ANSI® x3.131-1986 spec for further details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$11 )
	\$01	Code	( %000000xx )
		00	= Blocks
		01	= Filemarks
		10	= Sequential Filemarks
		11	= Logical End-of-Data
	\$02 - \$04	Count	(MSB»»LSB)
		This count specifies the number of blocks or filemarks to skip over. If the number is positive, it will result in forward movement of the target media. A negative (2's compliment) value will cause the media to moved in the reverse direction over count blocks or filemarks.	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition

**\$8013 - Verify; ( Sequential Access )**  
**[M]**

The Verify command cause the target to verify one or more blocks from the current position. There are two flags associated with this call. For further information about this call and it's implementation please refer to the ANSI® x3.131-1986 spec.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$13 )
	\$01	Immed	( %00000x00 )
		BytCmp	( %000000x0 )
		Fixed	( %0000000x )
	\$02 - \$04	Verification Length	( MSB»»LSB )
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

## **\$8014 - Write QIC-100 Information; ( Apple Tape Drive )**

The Write QIC-100 Information command allows the host to write QIC-100 information to the controller, which will then be written on tape upon execution of the next Write or Format commands. Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$14 )
	\$01 - \$0B	Reserved	

### **Request Length:**

\$0000000b

### **Transfer Length:**

\$0000000b

### **Buffer Data Structure:**

Please refer to the MCD-40/SCSI & DM Reference Manual for the details of this call.

### **Errors:**

Good  
Check Condition.

**\$8015 - Mode Select;**                      **( Direct Access Devices )**  
     **[0]**                                      **( Sequential Access )**  
                                             **( Printer Devices )**  
                                             **( Scanner Devices )**

The Mode Select command allows the host system to specify various parameters (Medium, Logical Unit, or Peripheral device) to the target. The targets that implement this command must also implement the Mode Sense command. This call applies to many of the SCSI Device Types and the developer should refer to the device specifications as well as the correct section of the ANSI® x3.131-1986 document for further details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$15 )
	\$01	Page Format (PF)	( %000x0000 )
		Save Pages (SP)	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$000000ff    (Bytes)

**Transfer Length:**

\$00000000 - \$000000ff    (Bytes)

**Buffer Data Structure:**

The structure varies depending on the targeted device type. Please see the device documentation as well as the ANSI® spec for byte by byte details of the Mode Select Parameter List.

**Errors:**

Good  
 Check Condition

## **\$8015 - Mode Select; ( Changer Devices )**

**[0]**

The Mode Select command allows the host system to specify various parameters (Medium, Logical Unit, or Peripheral device) to the target. The targets that implement this command must also implement the Mode Sense command. This call applies to many of the SCSI Device Types and the developer should refer to the device specifications as well as the correct section of the ANSI® x3.131-1986 document for further details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$15 )
	\$01	Save Pages (SP)	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$000000ff (Bytes)

### **Transfer Length:**

\$00000000 - \$000000ff (Bytes)

### **Buffer Data Structure:**

The structure varies depending on the targeted device type. Please see the device documentation as well as the ANSI® spec for byte by byte details of the Mode Select Parameter List.

### **Errors:**

Good  
Check Condition

**\$8015 - Mode Select;                      ( Communications Devices )**  
**[0]**

The Mode Select command allows the host system to specify various parameters (Medium, Logical Unit, or Peripheral device) to the target. The targets that implement this command must also implement the Mode Sense command. This call applies to many of the SCSI Device Types and the developer should refer to the device specifications as well as the correct section of the ANSI® x3.131-1986 document for further details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$15 )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$000000ff    (Bytes)

**Transfer Length:**

\$00000000 - \$000000ff    (Bytes)

**Buffer Data Structure:**

The structure varies depending on the targeted device type. Please see the device documentation as well as the ANSI® spec for byte by byte details of the Mode Select Parameter List.

**Errors:**

Good  
Check Condition

**\$8016 - Reserve Unit;**                      **( Direct Access Devices )**  
**[M]**                                              **( WORM Devices )**  
                                                  **( Read-Only Direct Access )**

The Reserve Unit command is used to reserve a unit on the SCSI Bus for a particular host system and is used mostly in multi-host configurations. This command is supported to give the application full access to the target devices on the system. The reader should refer to the device and ANSI® Documentation to determine if this command offers a function that they need.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$16 )
	\$01	3rdPty	( %000x0000 )
		3rd Party Device ID	( %0000xxx0 )
		Extent	( %0000000x )
	\$02	Reservation Ident	
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$0000ffff (Bytes)

**Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

**Buffer Data Structure:**

Refer to documentation.

**Errors:**

Good  
 Check Condition  
 Reservation Conflict

**NOTE:** See also **\$8016 - Reserve Units** described below.

**\$8016 - Reserve Unit;**                      **( Sequential Access Devices )**  
    **[M]**                                      **( Printer Devices )**  
                                            **( Scanner Devices )**

The Reserve Unit command is used to reserve a unit on the SCSI Bus for a particular host system and is used mostly in multi-host configurations. This command is supported to give the application full access to the target devices on the system. The reader should refer to the device and ANSI® Documentation to determine if this command offers a function that they need.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$16 )
	\$01	3rdPty	( %000x0000 )
		3rd Party Device ID	( %0000xxx0 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

Refer to documentation.

**Errors:**

Good  
Check Condition  
Reservation Conflict

**NOTE:** See also **\$8016 - Reserve Units** described above and below.



## **\$8016 - Reserve Unit; ( Apple Tape Drive )**

**[M]**

The Reserve Unit command is used to reserve a unit on the SCSI Bus for a particular host system and is used mostly in multi-host configurations. This command is supported to give the application full access to the target devices on the system. The reader should refer to the 3M MCD-40 DM/SCSI Device as well as the ANSI® Documentation to determine if this command offers a function that they need.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$16 )
	\$01	3rdPty	( %000x0000 )
		3rd Party Device ID	( %0000xxx0 )
		Extent	( %0000000x )
	\$02	Reservation Identification	
	\$03 - \$04	Reserved	
	\$05	Reserved	( %00000000 )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$0000xxxx (Bytes)  
In multiples of 8

### **Transfer Length:**

\$00000000 - \$0000xxxx (Bytes)  
In multiples of 8

### **Buffer Data Structure:**

Refer to documentation.

### **Errors:**

Good  
Check Condition  
Reservation Conflict

**NOTE:** See also **\$8016 - Reserve Units** described above and below.

## **\$8016 - Reserve Unit; ( Changer Devices )**

**[M]**

The Reserve Unit command is used to reserve a unit on the SCSI Bus for a particular host system and is used mostly in multi-host configurations. This command is supported to give the application full access to the target devices on the system. The reader should refer to the device and ANSI® Documentation to determine if this command offers a function that they need.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$16 )
	\$01	3rdPty	( %000x0000 )
		3rd Party Device ID	( %0000xxx0 )
		Element	( %0000000x )
	\$02	Reservation Ident	
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$0000ffff (Bytes)

### **Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

### **Buffer Data Structure:**

Refer to documentation.

### **Errors:**

Good  
Check Condition  
Reservation Conflict

**NOTE:** See also **\$8016 - Reserve Units** described above.

**\$8017 - Release Unit; ( Direct Access Devices )**  
**[M] ( WORM Devices )**  
**( Read-Only Direct Access )**

This is the compliment command to Reserve Unit. This instructs the target to release the specified, previously reserved area.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$17 )
	\$01	3rdPty	( %000x0000 )
		3rd Party Device ID	( %0000xxx0 )
		Extent	( %0000000x )
	\$02	Reservation Ident	
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

**NOTE:** See also **\$8017 - Release Units** described below.

**\$8017 - Release Unit; ( Sequential Access Devices )**  
**[M] ( Printer Devices )**  
**( Scanner Devices )**

This is the compliment command to Reserve Unit. This instructs the target to release the specified, previously reserved area.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$17 )
	\$01	3rdPty	( %000x0000 )
		3rd Party Device ID	( %0000xxx0 )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$8017 - Reserve Units** described above and below.

**\$8017 - Release Unit; ( Changer Devices )**  
**[M]**

This is the compliment command to Reserve Unit. This instructs the target to release the specified, previously reserved area.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$17 )
	\$01	3rdPty	( %000x0000 )
		3rd Party Device ID	( %0000xxx0 )
		Element	( %0000000x )
	\$02	Reservation Ident	
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

**NOTE:** See also **\$8017 - Reserve Units** described above.

**\$8019 - Erase; ( Sequential Access Devices )**  
**[M]**

This command instructs the target device erase part or all of the remaining media beginning at the current media position. 'Erased' means either the media shall be erased or a pattern shall be written that appears as a gap to the target device. The distance is controlled by the long bit. A long bit of one indicates that all remaining media on the target device shall be erased. A long bit of zero indicates that a peripheral device specified portion of the media only will be erased.

Some targets may reject the Erase command with the long bit set to one if the media is not at the beginning of the media.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$19 )
	\$01	Immed	( %000000x0 )
		Long	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

**\$801B - Start/Stop Unit; ( Direct Access Devices )**  
**[0] ( WORM Devices )**  
**( Read-Only Direct Access )**

The Start/Stop command instructs the target to enable or disable the device for further operations. An Immed bit of one requests that the status be returned as soon as the operation is initiated. An Immed bit of zero indicates that the target is to wait until the request is completed before sending the status code.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1B )
	\$01	Immed	( %0000000x )
	\$02 - \$03	Reserved	
	\$04	Load/Eject	( %000000x0 )
		Start	( %0000000x )
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

**NOTE:** See also **\$801B - Load/Unload** and **Stop Print** described below.

## **\$801B - Load/Unload; (Sequential Access Devices)** **[0]**

The Load/Unload command instructs the target to prepare, unload, or retention the media. The load operation is in addition to the autoload performed when the media was inserted or powered up. An Immed bit of one requests that the status be returned as soon as the operation is initiated. An Immed bit of zero indicates that the target is to wait until the request is completed before sending the status code.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1B )
	\$01	Immed	( %0000000x )
	\$02 - \$03	Reserved	
	\$04	End of Tape (EOT)	( %00000x00 )
		Retention Re-Ten	( %000000x0 )
		Load	( %0000000x )
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition

**NOTE:** See also **\$801B - Stop Print** described below or **Start/Stop Unit** above.



## **\$801B - Stop Print; (Printers)**

The Stop Print command causes a buffered printer device to halt any printing in an orderly fashion.

A retain bit of zero indicates that any buffered data is to be discarded while a one indicates that the data is to be pReserved. The Recover Buffered Data command would be used to recover the data for a subsequent Print command, or a Slew and Print can be used to cause the remaining unrecovered data to be printed followed by any new data sent by the command. Where or at which point the printing is suspended is unspecified.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1B )
	\$01	Retain Bit	( %0000000x )
	\$02	Vendor Unique	
	\$03 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition

**NOTE:** See also **\$801B - Load/Unload** described below or **Start/Stop Unit** above.

**\$801B - Scan;                    ( Scanner Devices )**  
**[0]**

The Scan Command is used to inform the target that we are ready to receive data from it and how much data we have room for or have requested.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1B )
	\$01 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

\$00000000 - \$000000ff    (Bytes)

**Transfer Length:**

\$00000000 - \$000000ff    (Bytes)

**Buffer Data Structure:**

The buffer contains one or more window identifier bytes. The identifier tells the target which window descriptor to use for this command. These are the same that are defined by the Define Window Parameters call.

**Errors:**

Good

## **\$801D - Send Diagnostic; ( All Devices )**

**[M]**

The Send Diagnostic command instructs the target to perform diagnostic test on itself, or the attached peripheral devices, or both. This command is usually followed by a Receive Diagnostic Results command except when doing a self test.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1D )
	\$01	Page Format (PF)	( %000x0000 )
		SelfTest	( %00000x00 )
		DevOfL	( %000000x0 )
		UnitOfL	( %0000000x )
	\$02 - \$04	Reserved	
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$0000ffff (Bytes diagnostic list)

### **Transfer Length:**

\$00000000 - \$0000ffff (Bytes diagnostic list sent)

### **Buffer Data Structure:**

The diagnostic parameter list is device specific. Refer to the Device manuals for further information about this structure.

### **Errors:**

Good  
Check Condition

**\$801E - Prevent/Allow Removal; ( Direct Access )**  
**[0] ( WORM Devices )**  
**( Read-Only Direct Access )**  
**( Sequential Access )**

The Prevent/Allow Medium Removal command instructs the target to enable or disable the removal of the media from the unit.

A prevent bit of one inhibits mechanisms that normally allow removal of the media. A zero allows the media to be removed. The prevent condition shall be terminated after a Prevent/Allow Medium Removal command with the prevent bit set to zero, or Bus Device Reset condition from the host or a 'hard' Reset condition.

Targets that contain cache memory shall logically perform a Flush Cache command for the entire media prior to allowing media removal.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$1E )
	\$01 - \$03	Reserved	
	\$04	Prevent Flag	( %0000000x )
	\$05	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$06 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

## **\$8024 - Define Window Parameters; ( Scanners )**

### **[M]**

The Define Window Parameters is a mandatory command for all scanners. It is used to pass information to the scanner detailing the task and how it is to be performed. Before the scanner can scan a document or image, the application must provide certain details about the scan area. This information is provided in the form of parameters defining a scan window. These parameters include size, position, scanning resolution, scanning composition, as well as other parameters for each window.

If sent, the window parameters data shall consist of one or more Window Descriptor Blocks. Each window descriptor block specifies the location of a rectangular region and the mode in which the region is to be scanned.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$24 )
	\$01 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$00ffffff (Length of window descriptor in bytes)

#### **Transfer Length:**

\$00000000 - \$00ffffff (Length of transferred data in bytes)

#### **Buffer Data Structure:**

Below is the structure for the parameters to be transferred by this call. These are given here because of the lack of available documentation detailing this call. The descriptions below are extracted from several documents to give an over all comprehensive description of the parameters listed.

The Request Length above is the length, in bytes, of a window descriptor. This means that the Window Descriptor Length times the number of descriptors sent should be eight less than the Request Length. In other words Request Length = Window Descriptor Length \* number of descriptor blocks + 8.

All other bytes in the Descriptor header (the first 8 bytes) are Reserved

Bit Byte	7	6	5	4	3	2	1	0
0 - 5	Reserved							
6	Window Descriptor Block Length (MSB)							
7	Window Descriptor Block Length (LSB)							
	Window Desriptor Block(s)							
0	Window Identifier							
1	Reserved							Auto
2-3	X Resolution (MSB»»LSB)							
4-5	Y Resolution (MSB»»LSB)							
6-9	Upper Left X (MSB»»LSB)							
A-D	Upper Left Y (MSB»»LSB)							
E-11	Width (MSB»»LSB)							
12-15	Length (MSB»»LSB)							
16	Brightness							
17	Threshold							
18	Contrast							
19	Image Composition							
1A	Bits per Pixel							
1B-1C	Halftone Patern (MSB»»LSB)							
1D	RIF	Reserved				Padding Type		
1E-1F	Bit Ordering (MSB»»LSB)							
20	Compression Type							
21	Compression Argument							
22-N	Reserved							

Each window descriptor contains information about one window.

The **Window Identifier** field contains a number between 0 and 255, which uniquely identifies the window defined by the block descriptor. Use this unique identifier to indicate each window during data transfers and status requests.

The **X Resolution** specifies the horizontal resolution in pixels per inch. A value of zero indicates that the scanner should use it's default resolution.

The **Y Resolution** specifies the vertical resolution in lines per inch. A value of zero indicates that the scanner should use it's default resolution.

The **Upper Left X** specifies the location of the X-coordinate of the upper left corner of this rectangular window and is measured in pixels as defined by X Resolution. The point 0,0 is considered the most upper-left corner of the window.

The **Upper Left Y** specifies the location of the Y-coordinate of the upper left corner of this rectangular window and is measured in lines as defined by Y Resolution. The point 0,0 is considered the most upper-left corner of the window.

The **Width** specifies the window width in pixels from left to right.

The **Length** specifies the window width in lines from top to bottom.

The **Brightness** has a range of one (lowest setting) through 255 (highest setting) with zero specifying the default value.

The **Threshold** is just that, the threshold setting of the scanner. A zero indicates that the scanner should use it's default setting for this. One is the lowest and 255 is the highest with 128 being the nominal setting.

The **Contrast** has a range of one (lowest setting) through 255 (highest setting) with zero specifying the default value.

The **Image** specifies the type of image acquired and is defined by the following table.

<u>Code</u>	<u>Description</u>
00	Bi-level black and white
01	Dithered/halftone black and white
02	Multi-level black and white (gray scale)
03	Bi-level RGB Color
04	Dithered/halftone RGB Color
05	Multi-level RGB Color
06 - FF	Reserved

The **Bits per Pixel** specifies the number of bits to be used to define each pixel. The higher the Image setting, the greater the number of bits required for each pixel.

The **Halftone Pattern** specifies the halftone process by which multi-level data is converted to binary data. This field shall be used in conjunction with the Image Composition code specified above.

The **Reverse Image Format (RIF)** bit is applicable only for images represented by one bit per pixel. A RIF bit of zero indicates

that white pixels are to be indicated by zeros and black pixels are to be indicated by ones. A RIF bit of one indicates the opposite.

The **Padding Type** specifies what operation is to be done if the scanned data to be transmitted to the host is not an integral number of bytes. The padding type is defined below.

<u>Code</u>	<u>Description</u>
00	No padding
01	Pad with 0's to byte boundary
02	Pad with 1's to byte boundary
03	Truncate to byte boundary
04 - FF	Reserved

The **Bit Ordering** field defines the order in which data is transferred to the host from the window. Ordering will include direction of pixels in a scan line, direction of scan lines within a window and data packing within a byte.

The **Compression type** and **Argument** fields specify the compression technique to be applied to the scanned data prior to transmission to the host and are defined below.

<u>Code</u>	<u>Compression Type</u>	<u>Argument</u>
00	No compression	Reserved
01	CCITT Group III, 1 dimensional	Reserved
02	CCITT Group III, 2 dimensional	K factor
03	CCITT Group IV, 2 dimensional	Reserved
04 - 0F	Reserved	Reserved
10	Optical Character Recognition (OCR)	Vendor Unique
11 - 7F	Reserved	Reserved
80 - FF	Vendor Unique	Vendor Unique

#### **Errors:**

Good  
Check Condition



## **\$802A - Write (Extended); ( Direct Access Devices )** **[M]**

The \$802A Write command is similar to the \$800A Write command only the Logical Block Address and the Transfer capabilities have been expanded. There are also some additional flags and the reader should refer to the device documentation for details about these flags and their function.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2A )
	\$01	DPO	( %000x0000 )
		FUA	( %0000x000 )
		Write Same (WrtSme)	( %00000x00 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB »» LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Buffer Data Structure:**

Data

### **Errors:**

Good  
Check Condition  
Reservation Conflict

**NOTE:** See also **\$802A - Send** and **Write** described below.

**\$802A - Send (Extended); ( Scanner Devices )**  
**[0]**

This command is similar in function to the write command except parameter not storage data is sent to the scanner.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2A )
	\$01	RelAdr	( %0000000x )
	\$02	Transfer Data Type	
	\$03	Reserved	
	\$04 - \$05	Transfer Identification	( MSB»»»LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00ffffff (Bytes)

**Transfer Length:**

\$00000000 - \$00ffffff (Bytes)

**Buffer Data Structure:**

Refer to the device documents for a description of this as well as the Transfer Data Type and Transfer Identification values.

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$802A - Write** above and below.

## **\$802A - Write (Extended); ( Optical Memory Devices )**

### **[0]**

The \$802A Write command is similar to the \$800A Write command only the Logical Block Address and the Transfer capabilities have been expanded. There are also some additional flags and the reader should refer to the device documentation for details about these flags and their function.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2A )
	\$01	DPO	( %000x0000 )
		FUA	( %0000x000 )
		Erase By Pass (EBP)	( %00000x00 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB »» LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

#### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

#### **Buffer Data Structure:**

Data

#### **Errors:**

Good  
Check Condition  
Reservation Conflict

**NOTE:** See also **\$802A - Send** and **Write** described above.

**\$802B - Seek (Extended); ( Direct Access Devices )**  
**[0] ( WORM Devices )**  
**( Read-Only Direct Access )**

This is an extended seek command for devices that support more block address than the standard seek allows.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2B )
	\$01	Reserved	
	\$02 - \$05	Logical Block Address	(MSB»»LSB)
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$802B - Locate** below.

## **\$802B - Locate (Extended); (Sequential Access Devices)**

The Locate command instructs the target device to position the media so that the block address specified is positioned for the next write call. The data written will be sent to this block address. If any data resides in the targets buffer, it will be written before the Locate command is executed. Refer to the Device and ANSI® documents for details concerning the flags and partition values for this command.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2B )
	\$01	Blk Address Type (BT)	( %00000x00 )
		Change Partition (CP)	( %000000x0 )
		Immed	( %0000000x )
	\$02	Reserved	
	\$03 - \$06	Logical Block Address	( MSB»»»LSB )
	\$07	Reserved	
	\$08	Partition	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition

**NOTE:** See also **\$802B - Seek** above.

## \$802C - Erase; ( Optical Memory )

[0]

The Erase command instructs the target device to erase or fill with blank pattern part or all of the remaining media starting with the block specified. See device and ANSI® documents for flag definition and usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2C )
	\$01	Erase All (ERA)	( %00000x00 )
		RelAdr	( %0000000x )
	\$02 - \$05	Starting Logical Blk Addr	( MSB»»LSB )
	\$06	Reserved	
	\$07 - \$08	Number of Blocks	( MSB»»LSB )
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

### Request Length:

Unused

### Transfer Length:

Unused

### Buffer Data Structure:

None

### Errors:

Good  
Check Condition  
Reservation Conflict

**NOTE:** This command seems to conflict with the \$802C command **Read Generation**. The ANSI® spec is unclear. As the ANSI® document is updated, these will be modified to reflect those changes.

## \$802C - Read Generation; ( Optical Memory )

[0]

The Erase command instructs the target device to erase or fill with blank pattern part or all of the remaining media starting with the block specified. See device and ANSI® documents for flag definition and usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2C )
	\$01	RelAdr	( %0000000x )
	\$02 - \$05	Logical Blk Addr	( MSB»»»LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

### Request Length:

\$00000000 - \$000000ff (Bytes)

### Transfer Length:

\$00000000 - \$000000ff (Bytes)

### Buffer Data Structure:

Generation data. See vendor and ANSI® documentation.

### Errors:

Good

**NOTE:** This command seems to conflict with the \$802C command **Erase**. The ANSI® spec is unclear. As the ANSI® document is updated, these will be modified to reflect those changes.

## **\$802E - Write and Verify;      ( Direct Access Devices )**

### **[0]**

The Write and Verify command requests that the target device write the data send and then after the write, verify that what is on the media is what the host sent. Refer to the Device and ANSI® documents for information concerning the flags and their usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2E )
	\$01	DPO	( %000x0000 )
		WrtSme	( %00000x00 )
		BytChk	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

#### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

#### **Buffer Data Structure:**

Data to be written and verified

#### **Errors:**

Good  
Check Condition  
Reservation Conflict

**NOTE:** See also **\$802E - Write and Verify** below.



## **\$802E - Write and Verify; ( Optical Memory Devices )**

### **[0]**

The Write and Verify command requests that the target device write the data send and then after the write, verify that what is on the media is what the host sent. Refer to the Device and ANSI® documents for information concerning the flags and their usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2E )
	\$01	DPO	( %000x0000 )
		EBP	( %00000x00 )
		BytChk	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

#### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

#### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

#### **Buffer Data Structure:**

Data to be written and verified

#### **Errors:**

Good  
Check Condition  
Reservation Conflict

**NOTE:** See also **\$802E - Write and Verify** above.

```
$802F - Verify;                ( Direct Access Devices )
      [O]
```

The Verify command is almost identical to the Write and Verify command except that no data is send or written. the target verifies the data that is on the media. Refer to the Device and ANSI® documents for information concerning the flags and their usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2F )
	\$01	DPO	( %000x0000 )
		BytChk	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06	Reserved	
	\$07 - \$08	Number of blocks	( MSB»»LSB )
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxxx )
	\$0A - \$0B	Reserved	

## Request Length:

None.

**Transfer Length:**

None.

### Buffer Data Structure:

None.

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$802F - Verify** below.

```
$802F - Verify;                ( Optical Memory Devices )
[0]                ( WORM Devices )
```

The Verify command is almost identical to the Write and Verify command except that no data is send or written. the target verifies the data that is on the media. Refer to the Device and ANSI® documents for information concerning the flags and their usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$2F )
	\$01	DPO	( %000x0000 )
		Blank Verify (BlkVfy)	( %00000x00 )
		BytChk	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06	Reserved	
	\$07 - \$08	Number of blocks	( MSB»»LSB )
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxxx )
	\$0A - \$0B	Reserved	

## Request Length:

None.

**Transfer Length:**

None.

### Buffer Data Structure:

None. The request length specifies the number of blocks to be verified.

**Errors:**

Good  
Check Condition

**NOTE:** See also **\$802F - Verify** above.

**\$8031 - Medium Position; ( Scanner Devices )**  
**[0]**

The Medium Position command provides a variety of media positioning functions. Absolute as well as relative positioning of the medium is provided, although some SCSI devices may only support a subset of this command. Such SCSI devices shall return a Check Condition Status. Please refer to the Device and ANSI® documents for further details concerning this command and its use.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$31 )
	\$01	Position Type	( %00000xxx )
	\$02 - \$04	Count	
	\$05 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**\$8033 - Set Limits;                   ( Direct Access Devices )**  
**[U]                                   ( WORM Devices )**  
**( Read-Only Direct Access )**

The Set Limits command defines the range within which subsequent linked commands may operate. A second Set Limits command may not be linked to a chain of commands in which a Set Limits command has already been issued.

A read inhibit (RdInh) bit of one indicates that read operations within the range are inhibited. A write inhibit (WrInh) bit of one indicates that write operations within the range are inhibited.

The logical block address specifies the starting address for the range. The number of blocks specifies the number of blocks within the range. A number of zero indicates that the range shall extend to the last logical block on the logical unit.

Any attempt to access outside of the restricted range or any attempt to perform an inhibited operation within the restricted range shall not be performed.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$33 )
	\$01	Read Inhibit (RdInh)	( %000000x0 )
		Write Inhibit (WrInh)	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06	Reserved	
	\$07 - \$08	Number of Blocks	( MSB»»LSB )
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**\$8034 - Pre Fetch; ( Direct Access Devices )**  
**[0]**

The Pre Fetch command request the target to transfer the requested blocks to the targets cache memory. No data is transferred to the host computer.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$34 )
	\$01	Immed	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB»»»LSB)
	\$06	Reserved	
	\$07 - \$08	Number of blocks	( MSB»»»LSB )
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

None.

**Transfer Length:**

None.

**Buffer Data Structure:**

None

**Errors:**

Good  
Condition Met  
Check Condition

## **\$8035 - Synchronize Cache; ( Direct Access Devices )**

**[0]**

The Synchronize Cache command ensures that all logical blocks within the specified range have their most recent data values recorded on the physical media. For each logical block within the specified range, if a more recent data value exists in the cache memory than on the physical media, then the logical block will be written from the cache to the media. Blocks are not necessarily removed from the cache by this command.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$35 )
	\$01	Immed	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB»»»LSB)
	\$06	Reserved	
	\$07 - \$08	Number of Blocks	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition

# **\$8036 - Lock/Unlock Cache;      ( Direct Access Devices )** **[0]**

The Lock/Unlock Cache command requests that the target disallow or allow logical blocks within the specified range to be removed from the cache memory by the target's cache replacement algorithm. Locked logical blocks may be written to the physical media when modified, but a copy of the modified logical block shall remain in the cache.

A lock bit of one indicates that any logical block in the specified range that is currently present in the cache shall be locked into the cache. Only logical blocks that are already present in the cache are actually locked. A lock bit of zero indicates that all logical blocks in the specified range that are currently locked into the cache shall be unlocked, but not necessarily removed.

The logical block address specifies the first logical block of the range to be locked. The number of blocks specifies the total number of contiguous blocks to be considered for locking or unlocking. A number of zero indicates that all the remaining blocks shall be considered.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$36 )
	\$01	Lock	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB»»»LSB)
	\$06	Reserved	
	\$07 - \$08	Number of Blocks	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

## **Request Length:**

Unused

## **Transfer Length:**

Unused

## **Buffer Data Structure:**

None

## **Errors:**

Good  
Check Condition



**\$8038 - Media Scan;                   ( Optical Memory Devices )**  
**[0]                                   ( WORM Devices )**

The Media Scan command will cause the target to scan a defined range searching for a contiguous span of the media either empty or written. Results are posted in the Sense Data Block. Refer to the Device and ANSI® documents for further details of this command and the flags associated with it.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$38 )
	\$01	Writn Blk Search (WBS)	( %000x0000 )
		Adv Scan Alg (ASA)	( %0000x000 )
		Rev Search Dir (RSD)	( %00000x00 )
		Part Rslt Accept (PRA)	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Beg. Logical Blk Address	( MSB»»LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000000 or \$00000008 (Bytes)

**Transfer Length:**

\$00000000 or \$00000008 (Bytes)

**Buffer Data Structure:**

Number of Blocks to Scan and Number of Blocks to Verify are specified in the following optional parameter block. If this block is omitted by a Request Length of zero, the Number of Blocks to Scan shall default to zero (scan to end of media) and the Number of Blocks to Verify shall default

\$00	Number of Blocks to Verify (MSB)
\$01	Number of Blocks to Verify
\$02	Number of Blocks to Verify
\$03	Number of Blocks to Verify (LSB)
\$04	Number of Blocks to Scan (MSB)
\$05	Number of Blocks to Scan
\$06	Number of Blocks to Scan
\$07	Number of Blocks to Scan (LSB)

**Errors:**

Good  
Check Condition  
Condition Met

## \$803B - Write Buffer;

The Write Buffer command is used along with the Read Buffer command as a diagnostic tool for testing target memory and the SCSI Bus integrity. This command shall not alter the media. The reader should refer to the ANSI® x3.131-198x spec for more detailed information about this call.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$3B )
	\$01	Mode	( %00000xxx )
	000	Combined Header and Data	Optional
	001	Vendor Unique	Vendor Unique
	010	Data	Optional
	011	Descriptor	Optional
	100	Download Microcode	Optional
	101	Download Microcode and Save	Optional
	11x	Reserved	Reserved
	\$02	Buffer ID	
	\$03 - \$05	Buffer Offset	( MSB »» LSB )
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

### Request Length:

\$00000000 - \$00ffffff (Bytes)

### Transfer Length:

\$00000000 - \$00ffffff (Bytes)

### Buffer Data Structure:

See vendor and ANSI® x3.131-198x documents for details pertaining to this command

### Errors:

Good  
Check Condition

**\$803D - Update Block; ( Optical Memory Devices )**  
**[0] ( WORM Devices )**

The Update Block command logically replaces data on the media with new data. Please refer to the vendors device spec and the ANSI® documents for details about the implementation of this command.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$3D )
	\$01	RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB»»LSB)
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Buffer Data Structure:**

Data to be used in update.

**Errors:**

Good  
Check Condition

# **\$803F - Write Long;            ( Direct Access Devices )** **[0]**

The Write Long command request that the target transfer data from the host computer. This data is implementation specific, but shall include the data bytes and the ECC bytes. Any other data correctable by ECC should also be included.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$3F )
	\$01	RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB »» LSB)
	\$06 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

## **Request Length:**

\$00000000 - \$0000ffff (Bytes)

## **Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

## **Buffer Data Structure:**

Data to the target including ECC and other information (Implementation specific).

## **Errors:**

Good  
Check Condition

# **\$8045 - Play Audio(10);**                      **( New for SCSI-2 CD-ROM drives )** **[0]**

The PLAY AUDIO command (see table 240) requests that the target begin an audio playback operation. The command function (Immed and SOTC bits) and the output of audio signals shall be as specified by the settings of the mode parameters (see the CD-ROM Parameters Page section).

Table 240 - PLAY AUDIO(10) command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (45h)							
1	Logical unit number			Reserved				RelAdr
2	(MSB)							
3	Starting logical block address							
4								
5								
6	Reserved							
7	(MSB)							
8	Transfer length							(LSB)
9	Control							

If any commands related to audio operations are implemented then the PLAY AUDIO command shall be implemented to allow a method for the initiator to determine if audio operations are supported. A target responding to a PLAY AUDIO command that has a transfer length of zero with CHECK CONDITION status and setting the sense key to ILLEGAL REQUEST does not support audio play operations.

The logical block address field specifies the logical block at which the audio playback operation shall begin.

The transfer length field specifies the number of contiguous logical blocks that shall be played. A transfer length field of zero indicates that no audio operation shall occur. This condition shall not be considered an error.

If the logical block length is not equal to the sector size, the target may adjust the starting logical block address and the transfer length. In such case, it is recommended that the target start the audio play operation with the beginning of a sector whenever the starting logical address falls within that sector (MSF unit). If the requested transfer length causes the end of an audio play operation to fall within a sector, the target may continue the play operation through the end of that sector.

If the starting address is not found, if the address is not within an audio track, or if a not ready condition exists, the command shall be terminated with CHECK CONDITION status.

If the CD-ROM information type (data vs. audio) changes, the sense key shall be set to BLANK CHECK and the additional sense code set to END OF USER AREA ENCOUNTERED ON THIS TRACK.

If the logical block address requested is not within an audio track, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to BLANK CHECK and the additional sense code set to ILLEGAL MODE FOR THIS TRACK.

**Request Length:**

\$00000000 - \$0000ffff (Blocks)

**Transfer Length:**

\$00000000 - \$0000ffff (Blocks)

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

## \$8047 - Play Audio MSF; ( New for SCSI-2 CD-ROM drives )

### [0]

The PLAY AUDIO MSF command (see table 242) requests that the target to begin an audio playback operation. The command function (Immed and SOTC bits) and the output of audio signals shall be as specified by the settings of the mode parameters (see the CD-ROM Parameters Page section).

Table 242 - PLAY AUDIO MSF command

Bit	6	5	4	3	2	1	0
Byte							
0	Operation code (47h)						
1	Logical unit number			Reserved			
2	Reserved						
3	Starting M field						
4	Starting S field						
5	Starting F field						
6	Ending M field						
7	Ending S field						
8	Ending F field						
9	Control						

The starting M field, the starting S field, and the starting F field specify the absolute MSF address at which the audio play operation shall begin. The ending M field, the ending S field, and the ending F field specify the absolute MSF address where the audio play operation shall end. All contiguous audio sectors between the starting and the ending MSF address shall be played.

A starting MSF address equal to an ending MSF address causes no audio play operation to occur. This shall not be considered an error. If the starting MSF address is less than the ending MSF address, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST.

If the starting address is not found, if the address is not within an audio track, or if a not ready condition exists, the command shall be terminated with CHECK CONDITION status.



**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**\$8048 - Play Audio Track Index(10); ( New for SCSI-2 CD-ROM drives )**

[0]

The PLAY AUDIO TRACK INDEX command (see table 243) requests the target to begin an audio play operation. The command function (Immed and SOTC bits) and the output of audio signals shall be as specified by the settings of the mode parameters (see the CD-ROM Parameters Page section).

Table 243 - PLAY AUDIO TRACK INDEX command

+=====+																				
Bit	7		6		5		4		3		2		1		0					
Byte																				
+=====+																				
0	Operation code (48h)																			
+-----+																				
1	Logical unit number						Reserved													
+-----+																				
2	Reserved																			
+-----+																				
3	Reserved																			
+-----+																				
4	Starting track																			
+-----+																				
5	Starting index																			
+-----+																				
6	Reserved																			
+-----+																				
7	Ending track																			
+-----+																				
8	Ending index																			
+-----+																				
9	Control																			
+=====+																				

The starting track field specifies the track number of the starting audio track. The starting index field specifies the index number within the track at which the audio play operation shall begin. The ending track field specifies the track number of the ending audio track. The ending index field specifies the index number within the track after which the audio play operation shall stop. The audio play operation shall terminate at the last block with an index number equal to the ending index. All contiguous audio sectors between the starting and the ending address shall be played.

If the starting address is less than the ending address, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST.

If the starting address is not found, or if the address is not within an audio track, or if a not ready condition exists, the command shall be terminated with CHECK CONDITION status. See 14.1.7 for a description of error reporting information.

NOTE 174 Valid values for the track and index fields are 1 to 99. A starting index value of one specifies that playback is to start with the first audio

sector of the track following the (optional) pause. A last index value of 99 specifies that playback continues through the last sector of the track.

If the ending track is greater than the last information track on the media, the playback shall continue until the last track is complete. If the ending index is greater than the largest index value on the ending track, the playback shall continue until this track is complete; then terminate. These conditions shall not be considered errors.

If the starting index is greater than the largest index value on the starting track, and the stop on track crossing (SOTC) bit of the audio control MODE SELECT parameters page (see the CD-ROM Parameters Page section) is zero, the playback operation shall start at the beginning of the next track. This situation is not an error.

If the starting index is greater than the largest index value on the starting track, and the stop on track crossing (SOTC) bit of the audio control MODE SELECT parameters page (see the CD-ROM Parameters Page section) is one, the playback shall not begin. The target shall return CHECK CONDITION, and the sense key shall be set to ILLEGAL REQUEST.

NOTE 175 The operation of the SOTC bit described above comes about because the user may not be able to determine the largest index value on a track, either from the table of contents or by other means. The SOTC bit one case allows the user to determine the largest index. The SOTC bit zero case allows the user to set up play operations without complete knowledge of the media layout.

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good

Check Condition

**\$8049 - Play Track Relative(10); ( New for SCSI-2 CD-ROM drives )****[0]**

The PLAY AUDIO TRACK RELATIVE(10) command (see table 244) requests that the device begin an audio playback operation. The starting address is specified as a track relative logical block address within the specified starting track. The command function (Immed and SOTC bits) and the output of audio signals shall be as specified by the settings of the mode parameters (see the CD-ROM Parameters Page section).

Table 244 - PLAY AUDIO TRACK RELATIVE(10) command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (49h)							
1	Logical unit number				Reserved			
2	(MSB)							
3	Track relative logical block address							
4								
5								
6	(LSB)							
6	Starting track							
7	(MSB)							
8	Transfer length							
9	(LSB)							
9	Control							

The starting track field specifies the track number of the starting audio track.

The track relative logical block address (TRLBA) field specifies the two's complement starting logical block address relative to the beginning of the first sector on the track with an index value of one. Negative values indicate a starting location within the audio pause area at the beginning of the requested track.

The transfer length field specifies the number of contiguous logical blocks that shall be output as audio data. A transfer length field of zero indicates that no audio playback operation shall occur. This condition shall not be considered an error. Any other value indicates the number of logical blocks that shall be output.

If the logical block length is not equal to the sector size the target may adjust the starting logical block address and the transfer length. In such case, it is recommended that the target start the audio play operation with the beginning of a sector whenever the starting logical address falls within that sector (MSF unit). If the requested transfer length causes the end of an

audio play operation to fall within a sector, the target may continue the play operation through the end of that sector.

If the starting address is not found, or if the address is not within an audio track, or if a not ready condition exists, the command is terminated with CHECK CONDITION status.

**Request Length:**

\$00000000 - \$0000ffff (Blocks)

**Transfer Length:**

\$00000000 - \$0000ffff (Blocks)

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**\$804B - Pause/Resume;**                      **( New for SCSI-2 CD-ROM drives )**  
    **[0]**

The PAUSE RESUME command (see table 239) requests that the device stop or start an audio play operation. This command is used with PLAY AUDIO commands issued while the immediate bit is set to one.

Table 239 - PAUSE RESUME command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (4Bh)							
1	Logical unit number			Reserved				
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							Resume
9	Control							

A resume bit of zero causes the drive to enter the hold track state with the audio output muted after the current block is played. A resume bit of one causes the drive to release the pause and begin play at the block following the last block played.

If an audio play operation cannot be resumed and the resume bit is one, the command is terminated with CHECK CONDITION status. If the resume bit is zero and an audio play operation cannot be paused, (no audio play operation has been requested, or the requested audio play operation has been completed), the command is terminated with CHECK CONDITION status.

It shall not be considered an error to request a pause when a pause is already in effect, or to request a resume when a play operation is in progress.

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**\$8055 - Mode Select; ( All Devices )**  
**[0]**

The Mode Select command allows the host system to specify various parameters (Medium, Logical Unit, or Peripheral device) to the target. The targets that implement this command must also implement the Mode Sense command. This call applies to many of the SCSI Device Types and the developer should refer to the device specifications as well as the correct section of the ANSI® document for further details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$55 )
	\$01	Page Format (PF)	( %000x0000 )
		Disable Blk Desript (DBD)	( %0000x000 )
	\$02	Page Control (PC)	( %xx000000 )
		Page Code	( %00xxxxxx )
	\$03 - \$08	Reserved	
	\$09	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )
	\$0A - \$0B	Reserved	

**Request Length:**

\$00000000 - \$0000ffff (Bytes)

**Transfer Length:**

\$00000000 - \$0000ffff (Bytes)

**Buffer Data Structure:**

The structure varies depending on the targeted device type. Please see the device documentation as well as the ANSI® spec for byte by byte details of the Mode Select Parameter List.

**Errors:**

Good  
 Check Condition



**\$80A5 - Move Medium; ( Changer Devices )**  
**[M]**

This command requests that the target device move a unit of media between two elements of the automatic media changer.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$A5 )
	\$01	Reserved	
	\$02 - \$03	Transport Element Addr	( MSB»»»LSB )
	\$04 - \$05	Source Address	( MSB»»»LSB )
	\$06 - \$07	Destination Address	( MSB»»»LSB )
	\$08 - \$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition

**\$80A5 - Play Audio(12); ( New for SCSI-2 CD-ROM drives )**  
**[0]**

The PLAY AUDIO(12) command (see table 241) requests that the target to begin an audio playback operation. The command function (Immed and SOTC bits) and the output of audio signals shall be as specified by the settings of the mode parameters (see the CD-ROM Parameters Page section). See the PLAY AUDIO(10) command for a description of the fields in this command.

Table 241 - PLAY AUDIO(12) command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (A5h)							
1	Logical unit number			Reserved				RelAdr
2	(MSB)							
3	Logical block address							
4								
5								
6	(MSB)							
7	Transfer length							
8								
9								
10	(LSB)							
11	Reserved							
12	Control							

**Request Length:**

\$00000000 - \$ffffffff (Blocks)

**Transfer Length:**

\$00000000 - \$ffffffff (Blocks)

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

# **\$80A6 - Exchange Medium; ( Changer Devices )** **[0]**

This call allows the host system to request the target device to exchange a piece of media from the transport element with one at a full element. Please refer to the vendor and ANSI® documents for details.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$A6 )
	\$01	Reserved	
	\$02 - \$03	Transport Element Addr	( MSB»»LSB )
	\$04 - \$05	Source Address	( MSB»»LSB )
	\$06 - \$07	First Destination Address	( MSB»»LSB )
	\$08 - \$09	Second Destination Addr	( MSB»»LSB )
	\$0A	Inv1	( %000000x0 )
		Inv2	( %0000000x )
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

## **Request Length:**

Unused

## **Transfer Length:**

Unused

## **Buffer Data Structure:**

None.

## **Errors:**

Good

**\$809A - Play Audio Track Relative(12); ( New for SCSI-2 CD-ROM drives )**

[0]

The PLAY AUDIO TRACK RELATIVE(12) command (see table 245) requests that the device begin an audio playback operation. The command function (Immed and SOTC bits) and the output of audio signals shall be as specified by the settings of the mode parameters (see the CD-ROM Parameters Page section). See the PLAY AUDIO TRACK RELATIVE(10) command for a description of the fields in this command.

Table 245 - PLAY AUDIO TRACK RELATIVE(12) command

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (A9h)							
1	Logical unit number			Reserved				
2	(MSB)							
3	Track relative logical block address							
4								
5								
6	(MSB)							
7	Transfer length							
8								
9								
10	Starting track							
11	Control							

**Request Length:**

\$00000000 - \$ffffffff (Blocks)

**Transfer Length:**

\$00000000 - \$ffffffff (Blocks)

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

# **\$80AA - Write; ( Optical Memory )** **[M]**

This call is mostly a duplication of the \$802A command except for larger transfer lengths.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$AA )
	\$01	DPO	( %000x0000 )
		FUA	( %0000x000 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»»LSB )
	\$06 - \$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

## **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

## **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

## **Buffer Data Structure:**

Data

## **Errors:**

Good

**\$80AC - Erase; ( Optical Memory )**  
**[0]**

The Erase command instructs the target device to erase or fill with blank pattern part or all of the remaining media starting with the block specified. See device and ANSI® documents for flag definition and usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$AC )
	\$01	Erase All (ERA)	( %00000x00 )
		RelAdr	( %00000000x )
	\$02 - \$05	Starting Logical Blk Addr	( MSB»»»LSB )
	\$06 - \$09	Number of Blocks	( MSB»»»LSB )
	\$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
 Check Condition  
 Reservation Conflict

## **\$80AE - Write and Verify; ( Optical Memory Devices )** **[0]**

The Write and Verify command requests that the target device write the data send and then after the write, verify that what is on the media is what the host sent. Refer to the Device and ANSI® documents for information concerning the flags and their usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$AE )
	\$01	DPO	( %000x0000 )
		EBP	( %00000x00 )
		BytChk	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06 - \$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

### **Buffer Data Structure:**

Data to be written and verified

### **Errors:**

Good  
Check Condition  
Reservation Conflict



## **\$80AF - Verify; ( Optical Memory Devices )**

### **[0]**

The Verify command is almost identical to the Write and Verify command except that no data is send or written. the target verifies the data that is on the media. Refer to the Device and ANSI® documents for information concerning the flags and their usage.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$AF )
	\$01	DPO	( %000x0000 )
		Blank Verify (BlkVfy)	( %00000x00 )
		BytChk	( %000000x0 )
		RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06 - \$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

#### **Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

#### **Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

#### **Buffer Data Structure:**

None. The request length specifies the number of blocks to be verified.

#### **Errors:**

Good  
Check Condition

**\$80B3 - Set Limits;                   ( Direct Access Devices )**  
**[U]                                   ( WORM Devices )**  
**( Read-Only Direct Access )**

The Set Limits command defines the range within which subsequent linked commands may operate. A second Set Limits command may not be linked to a chain of commands in which a Set Limits command has already been issued.

A read inhibit (RdInh) bit of one indicates that read operations within the range are inhibited. A write inhibit (WrInh) bit of one indicates that write operations within the range are inhibited.

The logical block address specifies the starting address for the range. The number of blocks specifies the number of blocks within the range. A number of zero indicates that the range shall extend to the last logical block on the logical unit.

Any attempt to access outside of the restricted range or any attempt to perform an inhibited operation within the restricted range shall not be performed.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$B3 )
	\$01	Read Inhibit (RdInh)	( %000000x0 )
		Write Inhibit (WrInh)	( %0000000x )
	\$02 - \$05	Logical Block Address	( MSB»»LSB )
	\$06 - \$09	Number of Blocks	( MSB»»LSB )
	\$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

**\$80BD - Update Block; ( Optical Memory Devices )**  
**[0] ( WORM Devices )**

The Update Block command logically replaces data on the media with new data. Please refer to the vendors device spec and the ANSI® documents for details about the implementation of this command.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$BD )
	\$01	RelAdr	( %0000000x )
	\$02 - \$05	Logical Block Address	(MSB»»LSB)
	\$06 - \$0A	Reserved	
	\$0B	Vendor Unique	( %xx000000 )
		Reserved	( %00xxxxxx )

**Request Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Transfer Length:**

\$00000000 - \$00xxxxxx (Bytes)

**Buffer Data Structure:**

Data to be used in update.

**Errors:**

Good  
 Check Condition

## **\$80C0 - Eject Disk;            ( Ruby Drive )**

The Eject Disk command instructs the target device to eject the media if removal is allowed.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C0 )
	\$01	Immed	( %0000000x )
	\$02 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition

## **\$80C8 - Audio Track Search; ( Ruby Drive )**

The Audio Track Search command provides a means for positioning the optical pickup at address specified by the Search Address parameter. This command returns the status byte when the requested search address is found.

A Play bit of zero indicates the target will enter the hold track state (i.e. pause) when Search Address is found. A Play bit of one indicates the target will output the audio channels in the specified Play Mode when the address is found.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C8 )
	\$01	SCSI Command Flags	( \$00 )
	\$02	Play Flag	( %000x0000 )
	\$03	Play Mode	( \$00 - \$0F )
	\$04 - \$07	Search Address	(MSB»»LSB)
	\$08	Address Type	( %xx000000 )
	\$09 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition

## **\$80C9 - Audio Play; (Ruby Drive)**

The Audio Play command request that the target position the optical pickup at the Playback Address specified and output the audio channels in the specified play mode when and if the Playback Address is located. The status is returned when the Playback Address has been found or if it is not found or if the target is not yet ready to accept the Audio Play command.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C9 )
	\$01	SCSI Command Flags	( \$00 )
	\$02	StopAddr	( %000x0000 )
	\$03	Play Mode	( \$00 - \$0F )
	\$04 - \$07	Playback Address	(MSB»»LSB)
	\$08	Address Type	( %xx000000 )
	\$09 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition

**\$80CA - Audio Pause; (Ruby Drive)**

The Audio Pause command temporarily stops the audio play operation and enters the hold track state (i.e. keeps the media at the same Q Subcode address).

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$CA )
	\$01	SCSI Command Flags	( \$00 )
	\$02	Pause Bit	( %000x0000 )
	\$03 - \$0B	Reserved	

**Request Length:**

Unused

**Transfer Length:**

Unused

**Buffer Data Structure:**

None

**Errors:**

Good  
Check Condition

## **\$80CB - Audio Stop; (Ruby Drive)**

The Audio Stop command causes the target to stop the audio play operation at the Stop Address. The media will spin down and the optical pickup will be held near the area of the stop address.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$CB )
	\$01	SCSI Command Flags	( \$00 )
	\$02 - \$05	Stop Address	(MSB»»LSB)
	\$06	Address Type	( %xx000000 )
	\$07 - \$0B	Reserved	

### **Request Length:**

Unused

### **Transfer Length:**

Unused

### **Buffer Data Structure:**

None

### **Errors:**

Good  
Check Condition



## \$80CD - Audio Scan; (Ruby Drive)

The Audio Scan command requests that the target device transfer the current audio play status and the starting Q Subcode address of the next track to the host computer.

The **Command Data** structure is defined as:

Byte	\$00	SCSI Command Number	( \$C9 )
	\$01	SCSI Command Flags	( \$00 )
	\$02	Scan Direction	( %000x0000 )
	\$03	Reserved	( \$00 )
	\$04 - \$07	Scan Address	(MSB»»LSB)
	\$08	Address Type	( %xx000000 )
	\$09 - \$0B	Reserved	

### Request Length:

\$00000000 (Bytes)

### Transfer Length:

\$00000000 (Bytes)

### Buffer Data Structure:

See Device and ANSI® Documents for parameter block descriptions.

### Errors:

Good  
Check Condition

**\$80CE - Audio Control; (Ruby Drive)**

The Audio Control command requests that the target device set the internal Volume levels for the Left and Right channels using the values supplied by this call.

The **Command Data** structure is defined as:

Byte \$00 SCSI Command Number ( \$CE )  
\$01 - \$0B Reserved.

**Request Length:**

\$00000004 (Bytes)

**Transfer Length:**

\$00000004 (Bytes)

**Buffer Data Structure:**

Audio Volume Control Data							
Bit	7	6	5	4	3	2	0
Byte							
0	L-Channel Volume Control						
1	R-Channel Volume Control						
2	Reserved						
3	Reserved						

**Errors:**

Good  
Check Condition

### **Driver Flush**

Call Parameters : Device Number        ≠        \$0000  
                  Call Number         =        \$0007  
                  DIB Pointer

Device Number:        *This word parameter specifies the target device.  
                         This parameter must be nonzero.*

Call Number:         *This word parameter specifies the type of call.*

DIB Pointer:         *This longword points to the DIB for the target  
                         device.*

This call is issued only in preparation for a close or shutdown call. A character device which maintains it's own buffer will output any portion of the contents of that buffer which has not already been output to the device. A driver that does not maintain it's own data buffers will take no action. This call is not supported by block device drivers and should return a '**BAD COMMAND**' error.



Apple\_Driver partitions even though they are not mounted volumes. In addition, it will be possible to tie a Driver entry to a partition, making it the boot partition even though it is not the first partition on a drive.

## Calls

The calls will be sent to the Driver in the form of a Device Specific Status or a Device Specific Control call. The Call Number supplied by the caller will determine whether the call is intended to modify the Disk Structure or only one of the many Partitions.

## Configuration Status Parameters

### Disk Calls

This call is used to get the current configuration of the device (Hard Disk) in question. These two calls are used to get information about the setup of the drive or the current status of a Partition. We will first describe the Call to the Disk.

The call to get the Disk Info is Device Specific Status Call and the Device Specific Control Call to set the data are both Code \$F000, **must be issued to the Head Device of partitioned media**, and have the following Parameter List (Figure 1):

\$00	Driver Number
\$02	DDM Buff Length
\$04	DDM Trans Count
\$06	DDM Buffer Pointer
\$0A	Driver Buff Length
\$0C	Driver Trans Count
\$0E	Driver Buffer Pointer
\$12	Drvr Data Buff Len
\$14	Drvr Data Trans Cnt
\$16	Drvr Data Buff Pointer

Figure 1

**Driver Number** This is the driver number being referenced. This number must be in the range of \$1 - \$F and is used as an index into the DDM Driver List to determine which Apple\_Driver is being referenced.

The structure of the DDM and Driver Info data can be found in Inside Macintosh Volume 5, pages 577-579

**DDM Buff Length** This is a word value indicating the size of the DDM Buffer. It must be \$0200

**DDM Trans Count** This is a result word returned by the driver. If data is transfered then this will be \$0200 on exit from the driver

**DDM Buff Pointer** This is a long word pointer to the buffer allocated by the caller to contain the DDM Data

**Driver Buff Len** This is a word value indicating the size of the Driver Buffer. It must be \$0200

**Driver Trans Cnt** This is a result word returned by the driver. If data is transfered then this will be \$0200 on exit from the driver

**Driver Buff Ptr** This is a long word pointer to the buffer allocated by the caller to contain the Driver Info

**Drvr Data Buff Len** This is a word value indicating the size of the Driver Data Buffer. It must be large enough to contain all the Driver Data and is a multiple of \$0200

**Drvr Data Tr Cnt** This is a result word returned by the driver. If data is transfered then this will reflect the amount of data actually read on exit from the driver

**Drvr Data Buff Ptr** This is a long word pointer to the buffer allocated by the caller to contain the Driver Data. This is different from the Driver Info above in that the Driver Info is data from the Partition Map Entry for the Driver and the Driver Data is the actual Driver.

**NOTE: All three fields are to be null if no action is required for that segment of information. If the pointer is non-zero and the Buffer Length is not large enough for the data, then the Resulting transfer count will be the minimum sized buffer needed for that data and no other information is returned.**

## Partition Calls

This call is used to get or set the current Status of the volume (Partition) in question. This includes Read Enable, Write Enable, Remount the Volume as well as other items.

The call to get the Volume Status is Device Specific Status Call and the Device Specific Control Call to set the info or remount the volume are both Code \$F001 and have the following Parameter List (Figure 2):



Figure 2

Bitmap      This bitmap indicates what the current/new Read/Write Enable Status is for this partition.

Bit 14 is used to indicate that we want this volume to be remounted. Bit 14 being clear for this call will result in no error.

Bits 4 and 5 are used for the access level for this drive.

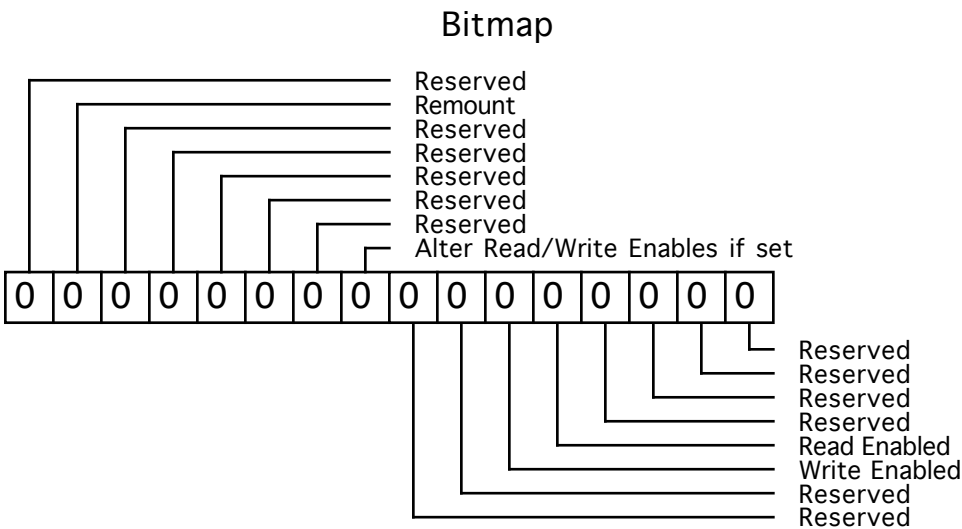


Figure 6

On the return from the status call, bits 4 and 5 will indicate if this volume is Read and/or Write Enabled.

**Note:** Bits 4 and 5 are used to set the Access Status to this volume even if they are both null. It is possible to Read Inhibit a volume making it inaccessible to GS/OS except through a DControl/DStatus Call. **Because of this, I have added bit 8 to indicate that the caller wishes to alter the access for this drive. If the caller is only trying to remount the volume, then having bit 8 clear will keep the driver from trying to modify the access for the volume.**

## **Device Driver Error Codes**

All error codes listed below must be supported by device drivers wherever applicable. All block device drivers must support disk switched errors without exception. Please take note that the error codes are returned from a device driver must have the high byte cleared. The device dispatcher maintains certain error codes under certain conditions. Device dispatcher error codes are passed in the upper byte of the accumulator.

Error Code	Description	Mnemonic
\$0000	No error occurred	NO_ERROR
\$0010	Device not found	DEV_NOT_FOUND
\$0011	Invalid Device Number	INVALID_DEV_NUM
\$0020	Invalid request	DRVR_BAD_REQ
\$0021	Invalid control or status code	DRVR_BAD_CODE
\$0022	Invalid parameter	DRVR_BAD_PARM
\$0023	Device not open (character driver)	DRVR_NOT_OPEN
\$0024	Device already open (character driver)	DRVR_PRIOR_OPEN
\$0026	Resource not available	DRVR_NO_RESRC
\$0027	I/O error	DRVR_IO_ERROR
\$0028	Device not connected	DRVR_NO_DEV
\$0029	Device is busy	DRVR_BUSY
\$002B	Write Protected (block driver only)	DRVR_WR_PROT
\$002C	Invalid Byte Count	DRVR_BAD_COUNT
\$002D	Invalid Block Number (block driver only)	DRVR_BAD_BLOCK
\$002E	Disk Switched (block driver only)	DRVR_DISK_SW
\$002F	Device Off Line or No Media Present	DRVR_OFF_LINE
\$004E	Invalid access or access not allowed	INVALID_ACCESS
\$0058	Not a block device	NOT_BLOCK_DEV
\$0060	Data is unavailable	DATA_UNAVAIL



**Miscellaneous CD-ROM tables**

The following tables are extracted from the SCSI-2 documentation and may be interested to Apple IIgs developers.

**Table 264 - CD-ROM medium type codes**

The medium-type code field is contained in the mode parameter header (see the Mode Sense section). Table 264 defines the medium type values for CD-ROM devices.

Code	Medium type description
00h	Default (only one type supported)
01h	120 mm CD-ROM data only
02h	120 mm CD-DA audio only
03h	120 mm CD-ROM data and audio combined
04h	Reserved
05h	80 mm CD-ROM data only
06h	80 mm CD-DA audio only
07h	80 mm CD-ROM data and audio combined
08h - 7Fh	Reserved
80h - FFh	Vendor-specific

**Table 265 - CD-ROM device-specific parameter**

The device-specific parameter field is contained in the mode parameter header (see the CD-ROM Parameters Page section). Table 265 defines the device-specific parameter field for CD-ROM devices.

Bit	7	6	5	4	3	2	1	0
	Reserved			DPOFUA	Reserved			EBC

When used with the MODE SELECT command, the DPOFUA bit is not used and the field is reserved.

When used with the MODE SENSE command, a DPOFUA bit of one indicates that the target supports the DPO and FUA bits.

A disable page out (DPO) bit of one indicates that the target shall assign the logical blocks accessed by this command the lowest priority for being fetched into or retained by the cache. A DPO bit of one overrides any retention priority specified in the cache page (see 9.3.3.1). A DPO bit of zero indicates the priority shall be determined by the retention priority fields in the cache page. All other aspects

of the algorithm implementing the cache memory replacement strategy are not defined by this International Standard.

**NOTE 107** The DPO bit is used to control replacement of logical blocks in the cache memory when the host has information on the future usage of the logical blocks. If the DPO bit is set to one, the host knows the logical blocks accessed by the command are not likely to be accessed again in the near future and should not be put in the cache memory nor retained by the cache memory. If the DPO bit is zero, the host expects that logical blocks accessed by this command are likely to be accessed again in the near future.

A force unit access (FUA) bit of one indicates that the target shall access the media in performing the command prior to returning GOOD status. Read commands shall access the specified logical blocks from the media (i.e. the data is not directly retrieved from the cache). In the case where the cache contains a more recent version of a logical block than the media, the logical block shall first be written to the media. Write commands shall not return GOOD status until the logical blocks have actually been written on the media (i.e. the data is not write cached).

An FUA bit of zero indicates that the target may satisfy the command by accessing the cache memory. For read operations, any logical blocks that are contained in the cache memory may be transferred to the initiator directly from the cache memory. For write operations, logical blocks may be transferred directly to the cache memory. GOOD status may be returned to the initiator prior to writing the logical blocks to the medium. Any error that occurs after the GOOD status is returned is a deferred error, and information regarding the error is not reported until a subsequent command.

The enable blank check (EBC) bit is reserved.

**Table 266 - CD-ROM density codes**

The density code field is contained in the mode parameter block descriptor (see the Mode Sense section). Table 266 defines the density code values for CD-ROM devices.

Code	Data types to be transferred
00h	Default density code
01h	User data only (2 048 bytes per physical sector)
02h	User data plus auxiliary data field (2 336 bytes per sector)
03h	4-byte tag field, user data plus auxiliary data (2 340 bytes per sector)
04h	Audio information only (1/75th of a second per logical block)
05h - 7Fh	Reserved
80h - FFh	Vendor-specific

**NOTE 187** The number of bytes per sector specified by this parameter is used with the block length to map CD-ROM sectors to logical block addresses.

**CD-ROM Parameters Page**

Or known as the CD-ROM audio control parameters page, the following section is important to understand if you want to control the output port and volume of your CD-ROM disk drive. The values are sent to the initiator with the MODE SENSE command, and received by the initiator by calling MODE SELECT.

The parameters page for CD-ROM audio control has page code 0Eh.

The CD-ROM audio control parameters page (see table 268) sets the playback modes and output controls for subsequent PLAY AUDIO commands and any current audio playback operation.

Table 268 - CD-ROM audio control parameters page

Bit	7	6	5	4	3	2	1	0
Byte								
0	PS	Reserved	Page code (0Eh)					
1	Parameter length (0Eh)							
2	Reserved					Immed	SOTC	Reserved
3	Reserved							
4	Reserved							
5	APRVal	Reserved			Format of LBAs / Sec.			
6	(MSB)	Logical blocks per second of audio playback						--
7								(LSB)
8	Reserved				Output port 0 channel selection			
9	Output port 0 volume							
10	Reserved				Output port 1 channel selection			
11	Output port 1 volume							
12	Reserved				Output port 2 channel selection			
13	Output port 2 volume							
14	Reserved				Output port 3 channel selection			
15	Output port 3 volume							

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit of one indicates that the target is capable of saving the page in a non-volatile vendor-specific location.

An immediate (Immed) bit of zero indicates the target shall not send completion status until the audio playback operation is terminated.

An Immed bit of one indicates the target shall send completion status as soon as the playback operation has been started.

NOTE 188 It is recommended that a Logical Unit type RESERVE be issued prior to starting audio play operations with an Immed bit of one in any multiple initiator environment.

A stop on track crossing (SOTC) bit of zero indicates the target shall terminate the audio playback operation when the transfer length is satisfied. Multiple tracks shall be played as necessary. Periods of time encoded as audio pause/silence at the beginning of tracks, (index 0) shall also be played.

A stop on track crossing (SOTC) bit of one indicates the target shall terminate the audio playback operation when the beginning of a following track is encountered.

The audio playback rate valid (APRVal) bit value of one indicates that the format of logical blocks per second field and the logical blocks per second of audio playback field are valid.

The format of logical blocks addresses per second field gives the multiplier to be used with the logical blocks per second of audio playback. This is defined in table 269.

Table 269 - Multiplier for LBAs

Format of LBAs / Sec value	Multiplier for LBAs / Sec field
0h	1
1h - 7h	Reserved
8h	1/256
9h - Fh	Reserved

NOTE 189 This field is provided as a means to return fractional (i.e. non-integral) values in the logical block addresses per second of audio playback. This shall occur when logical block sizes that are not even multiples or divisions of the physical block size are used.

The logical blocks per second of audio playback field gives the relationship between time and the duration of play per logical block address. The value in this field is to be multiplied by the value in format of LBAs per second field.

NOTE 190 The logical blocks per second of audio playback field and its companion format of LBAs per second field may not be supported by most current CD-ROM devices as a modifiable mode select parameter.

The output port channel selection specifies the audio channels from the disc to which this output port should be connected (see table 270). More than one output port may be connected to an audio channel. More than one audio channel may be connected to an output port.

Table 270 - Output port channel selection

+=====+		
Code	Description	
+=====+		
0000b	output port muted	
0001b	connect audio channel 0 to this output port	
0010b	connect audio channel 1 to this output port	
0100b	connect audio channel 2 to this output port	
1000b	connect audio channel 3 to this output port	
+=====+		

The channel volume control indicates the relative volume level for this audio output port. A value of zero indicates the output is muted, and a value of FFh indicates maximum volume level.

NOTE 191 If volume controls are implemented, the default volume level should be no more than 25 % of the maximum level as a personal safety consideration.

### **Extracting digital audio through the SCSI-2 bus**

The following CD-ROM drives from Apple allow you to extract audio through the SCSI bus :

- Apple 300 (Sony CDU-8003)
- Apple 300e (Matsushita CR-8004)

The list is non-exhaustive.

The data sent back by the target is a left/right pairs of 16-bit digital audio samples. There are 2352 bytes per CD-ROM block.

The two following commands are not implemented in the current GS/OS® SCSI-2 CD-ROM driver. The reason is that the two Group 6 commands of 12 bytes and Apple has implemented Group 6 10 bytes commands in its driver for communicating with the CD-SC drives. We would have broken compatibility by adding the commands.

Last note from this ERS : one can use them through the GENERIC SCSI call \$2B supported by the Apple High Speed DMA SCSI card. The commands run fine, trust me ;-)

# **\$D8 - Read CD-DA (for CD-Audio only)**

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (D8h)							
1	Logical unit number				Reserved			
2	(MSB)							
3	Track relative logical block address							
4								
5								
6								
7	(MSB)							
8	Transfer length							
9								
10								
11	Starting track							
12	Control							
13								

The values for the Subcode selector byte are :

Code	Description
0000b	standard : 2352 bytes
0001b	reserved
0010b	reserved
0011b	reserved

# **\$D9 - Read CD-DA MSF (for CD-Audio only)**

Bit	7	6	5	4	3	2	1	0
Byte								
0	Operation code (D9h)							
1	Logical unit number			Reserved				
2	Reserved							
3	Starting M field							
4	Starting S field							
5	Starting F field							
6	Reserved							
7	Ending M field							
8	Ending S field							
9	Ending F field							
10	Subcode selector							
11	Control							

The values for the Subcode selector byte are :

Code	Description
0000b	standard : 2352 bytes
0001b	reserved
0010b	reserved
0011b	reserved