New Technical Notes

Macintosh

Developer Support

Disk Driver Q&As

Devices

Revised by: Developer Support Center May 1993
Written by: Developer Support Center October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

Logical SCSI block numbers of a Mac partition on a SCSI drive
Date Written: 8/19/92
Last reviewed: 10/13/92

How can I find out the physical block location (SCSI ID, start/end block# ...) of a SCSI hard disk partition from volume information (VCB...)?

—

The SCSI ID of a drive can be determined by examining the driver reference number of the driver handling a volume. Any SCSI device driver's unit table number will be 32 + the SCSI ID. Because driver refNums are equal to -(unit table # + 1), the refNum will be equal to -33-SCSI ID. Thus, given a refNum, one can get the SCSI ID: SCSI ID = -(refNum + 33).

Now all that needs to be done is to determine where on the disk the partition resides. Unfortunately, this cannot be done from the VCB, because the translation from the position on the volume to the position on the device is done entirely by the disk's driver. To do this, you need to be able to read and parse the partition map on the SCSI device. This is described on and around page 576 of *Inside Macintosh* Volume V. You'll need to determine the location of all the volumes on the drive, and then examine each partition of the type APPLE_HFS to determine if it is the volume you are looking for. This can be done by

examining logical block #2 of the partition (beginning 1024 bytes in) and examining it as a Volume Information Block, as described on pages 165 and 166 of *Inside Macintosh* Volume IV. In this way, you should be able to get the volume's name and creation date. This should be enough to ensure you are examining the correct partition on the SCSI disk in question.

Macintosh RAM disk drive queue checksum flag
Date Written: 7/13/92
Last reviewed: 10/11/92

I noticed that the Macintosh Quadra's built-in RAM disk drive flags during startup are: 00480001. I understand all these bits (IM IV-181) except the 0001 part. Shouldn't these bits be 0? What do these bits do?

—

The low bit set in drive queue flag byte 3 indicates that the drive has checksumming turned on; this is a new feature introduced with the EDisk driver which indicates that blocks should be checksummed as they are read or written. For a reference to the EDisk checksumming scheme, see Designing Cards and Drivers, 3rd edition, page 481.

Macintosh PRAM and determining disk write-verify state
Date Written: 5/21/92
Last reviewed: 9/15/92

I want to have a control panel to turn write verify on or off for my disk driver. The problem I am experiencing is that if you start up using this disk, there's no way for the driver to access any of the stored information telling it if it is on or off. Can I store this bit in the PRAM? If not, got any other ideas on how I could do it?

—

Sorry, but there is no parameter RAM available for your driver; it's reserved for use by the system. You could store the bit within your driver image on the disk, perhaps just after the driver header. Your control panel could make a (private) control call to the driver to change the bit; the driver could perform the actual change, then write itself back out to the disk (this way, the location of the bit within the driver would have to be private to the driver).

Macintosh csCode 100
Date Written: 9/17/91
Last reviewed: 8/1/92

What is csCode 100? A control call with csCode 100 seems to come into my block device driver whenever the Macintosh system fails to recognize the file system on my media. Should my driver specify valid non-HFS formats such as ISO 9660?

—

The csCode 100 you're seeing is a ReadTOC command bound for what the operating system thinks is a CD-ROM drive. What's happening here is that after deciding your device doesn't contain an HFS partition, Foreign File Access (FFA) is attempting to identify the file system

residing on that device; looking at the Table of Contents on a CD is one of the ways that it attempts to go about that. This csCode is documented in the *AppleCD SC Developer's Guide*, which is available through APDA (#A7G0023/A).

If you remove the Audio CD File System Translator (FST) from your System Folder, you'll find that the csCode no longer gets issued. The reason is that the ReadTOC call returns information about audio CDs. Your driver doesn't even have to support this call; if FFA sees a

paramErr (which you should return upon seeing an unrecognized csCode) it knows it can't be looking at an audio CD.

You needn't worry about adding any support to your driver for ISO 9660 or High Sierra. FFA's FSTs will just request certain blocks from your driver which contain information that identifies the disk in question as ISO or HFS.

System 7 disk driver reentrancy requirements
Date Written: 8/15/91
Last reviewed: 8/1/92

What are the reentrancy requirements for System 7 disk device drivers?

—

If and only if multiple DCEs point to the same shared code in memory, it is possible for a disk driver to be reentered by VM. This will only happen when the driver is busy with one drive and VM wants to access another one controlled by the same driver as a result of a page fault at interrupt time. To prevent reentrancy problems, a disk driver should maintain a separate, independent set of variables for each drive it supports, and it should access those variables only through the DCtlPtr passed to the driver in A1 when it is called.

Many disk drivers don't even need to worry about this because a separate copy loads for each entry in the unit table.

X-Ref:
Macintosh Technical Note "Coping with VM and Memory Mappings"

Low-level Macintosh floppy disk access
Date Written: 8/8/91
Last reviewed: 8/1/92

I need to be able to modify the number of sectors per track and the interleave the disk drive looks for. There must be some way to do this through the driver. Do you have any example code of accessing the driver directly?

—

The only floppy access we support is through the driver, as documented in the Technical Note "What Your Sony Drives For You." The driver only handles block-level access, except that if you're using a Macintosh IIfx, the Raw Track Dump call gives direct read access to the data on the track. This may be just what you need; unfortunately, we don't have any example code that covers this call.

X-Ref:

Macintosh Technical Note “What Your Sony Drives for You”

Macintosh disk drivers and slot interrupts
Date Written: 8/30/91
Last reviewed: 8/1/92

Why are slot interrupts disabled while a floppy is being formatted?

—

In the present-day implementation, Macintosh slot interrupts are always disabled for floppy disk access. The floppy driver disables the interrupts using code like this:

```
OR.W #HiIntMask, SR
```

where

```
HiIntMask EQU $0300
```

For hard drives, interrupts are enabled with SCSI transfers. They are disabled for very short periods of time during SCSI reset and when things are being put into and removed from queues.

Note that some day Apple might implement SCSI chip interrupts, which might defer slot interrupts if a card wanted to interrupt while a SCSI interrupt was being serviced. To ensure future compatibility, please design variances in interrupt latency into your product.

VBL tasks under System 7 with virtual memory running

Date Written: 7/10/91
Last reviewed: 8/1/92

I use a VBL task to animate the cursor when my disk driver installer performs tasks such as media verification and low-level formatting. When a command such as Format Unit is sent to the drive, the Macintosh does not gain control until the drive completes the command, which ranges from a few minutes to close to an hour. Under System 7.0 with virtual memory (VM) running, the cursor is never even set, much less animated. I've tried HoldMemory, and I've tried allocating a system heap pointer and installing the VBL task there. Any help would be appreciated.

—

The reason your cursor is not animating when VM is running is fairly straightforward to explain.

VBL tasks are deferred until a time when paging is safe if the VBL interrupt occurs at a time when paging is not safe. Paging is not safe when the VM backing store drive's driver is busy, when a page fault is already being serviced, or when the SCSI bus is busy.

When you issue your Format Unit command, you call SCSIGet, which makes the SCSI bus busy. It probably takes several minutes before this command completes and you are able to call SCSIComplete, which makes the SCSI bus free. All during this SCSI bus tenure, VM paging is unsafe and no VBL tasks may execute. Holding memory will not help this situation.

It would be nice if the Macintosh SCSI Manager supported SCSI Disconnect/Reconnect, but this is not an option either at this point in time.

The beach ball in Apple HD SC Setup does continue to rotate during Apple HD SC Setup disk formatting, even under VM. Apple HD SC Setup uses the standard RotateCursor call in

the MPW CursorCtl.h interfaces rather than a VBL, and apparently never loses control to the SCSI

Manager. What Apple HD SC Setup does after SCSICmd returns is loop on SCSIStat until the SCSI bus enters status phase and then call SCSIComplete. This allows the cursor to rotate and a watchdog counter to time out the Format in case the drive dies. You're probably going to need to do something similar. The bottom line is that VBLs aren't going to do you much good during a long format.

Macintosh disk drivers and System 7 compatibility
Date Written: 5/9/91
Last reviewed: 8/1/92

Is there anything special that a Macintosh hard disk or a removable cartridge driver must do to be fully compatible with System 7?

—

One important thing you should be aware of regarding removable cartridges is that a cartridge can't be removable if VM is to use it for a backing store. Some removables allow you to fix this with a SCSI command to prohibit ejection and, just as important, the drive must be marked nonejectable in the drive queue.

Here are a couple of suggestions: Read the Macintosh Technical Note "Coping with VM and Memory Mappings." Also, take a look at the Memory Management chapter of *Inside Macintosh* Volume VI and the Virtual Memory paper (Goodies:VM Goodies:VM Paper) on the System 7 Golden Master CD.

Macintosh Finder eject on Unmount
Date Written: 12/20/90
Last reviewed: 8/1/92

Why doesn't dragging a mounted hard disk partition into the Trash generate an Eject call to the device driver?

—

Only ejectable media such as floppy disks generate an eject call. All others generate an Unmount call. If you were to set the ejectable bit (which is documented in *Inside Macintosh* Volume II, page 127-8), you will get an eject call for a hard disk which will then result in a continuous "Please insert <your hard disk>" alert.

Logical-to-physical sector information for 800K disks
Date Written: 11/8/90
Last reviewed: 8/1/92

Does the translation between logical and physical sectors on an 800K disk work in the same

manner as a 400K disk (as per *Inside Macintosh)?* If so, do track/sector numbers alternate between sides? Is this information available elsewhere, or should we be able to figure this out from data available from FEdit?

___

The translation between logical and physical sectors on an 800K disk follows in a manner similar to that for the 400K disk *(Inside Macintosh* Volume II, page 211), except that sectors alternate between sides before moving on to the next track, as you suggested.

First we'll define side 0 and side 1. Similar to the 400K disk, there are 80 tracks on each side. Track 0 is outermost, track 79 is innermost. The sectors per track are grouped in a similar manner to the 400K disk.

The first 12 sectors for an 800K disk map to side 0, track 0, sectors 0-11, similar to the corresponding sectors on the 400K disk. The next 12 sectors for the 800K disk map to side 1 in a similar manner—side 1, track 0, sectors 0-11. Logical sectors 24-36 map to side 0, track 1, sectors 0-11, and so on. The following chart demonstrates this:

Side	Track	Sectors Per Track	Physical Sectors	Logical Sectors
0	0	12	0-11	0-11
1	0	12	0-11	12-23
0	1	12	0-11	24-35
1	1	12	0-11	36-47
...				
1	15	12	0-11	372-383
0	16	11	0-10	384-394
1	16	11	0-10	395-405
0	17	11	0-10	405-415
...				
1	31	11	0-10	725-735
0	32	10	0-9	736-745
1	32	10	0-9	746-755
0	33	10	0-9	756-765
...				
1	47	10	0-9	1046-1055
0	48	9	0-8	1056-1064
1	48	9	0-8	1065-1073
0	49	9	0-8	1074-1082
...				
1	63	9	0-8	1335-1343
0	64	8	0-7	1344-1351
1	64	8	0-7	1352-1359
0	65	8	0-7	1360-1367
...				
1	79	8	0-7	1592-1599

Macintosh RAM disk driver tricks
Date Written: 5/21/90
Last reviewed: 8/1/92

How are you able to keep the Macintosh Portable's RAM Disk from being destroyed after it has been dragged to the trash? Could we pull that trick with our RAM disk too? How does the RAM Disk's memory stay protected during system startup? Also, the Finder's "Get Info" window for the RAM disk shows "Where: Internal RAM disk" rather than "Where: AppleTalk." How does the driver convey this information to the operating system?

—

At the time that your driver receives a disk eject, post a disk insertion event to the OS event queue. Your icon will disappear and reappear like magic.

The Macintosh Portable has a special version of the disk driver that looks for special header blocks for a RAM disk and multiple ROM disks in the ROM expansion address space. If it finds the correct header, it tells the system initialization (again specially coded for the Macintosh Portable) to not check the RAM mapped out by the RAM disk header.

The 'Get Info' dialog gets the "Where" information from the standard control call to the disk driver. Your driver should respond to a Status call with a csCode =21 as documented in *Inside Macintosh* Volume V, page 470.

How to determine if a disk drive supports ejectable media

Date Written: 5/3/89
Last reviewed: 8/1/92

How can I determine if a disk drive supports ejectable media?

—

The disk driver in Macintosh System 6.0.2 (and later) supports control calls with cscode = 23; this call returns information about the drive's physical location, size, and other characteristics described as follows:

```
Return Drive Info (csCode = 23) csParam = long;
```

Most of the bits are not used at the moment and are being reserved for future expansion. The drive type field is in bits 0 through 3 and describes the type of drive that is connected. The supported types are as follows:

```
  0    no such drive
  1    unspecified drive
  2    400K
  3    800K
  4    High Density drive FDHD
  5    reserved
  6    reserved
  7    Hard Disk 20
 8-15  reserved
```

The attributes field occupies bits 8 through 11 and describes the following:

```
  bit  8    0 - Internal    1 - external
  bit  9    0 - IWM         1 - SCSI
  bit 10    0 - removable   1 - fixed media
  bit 11    0 - primary     1 - secondary
```

So, bit 10 in csParm tells whether the drive in question supports ejectable media or not.

Drivers from other manufacturers may not support this control call, in which case it becomes necessary to traverse the queue in order to secure the information. The following little assembly language routine shows how to do it the old-fashioned way, as mentioned in *Inside Macintosh* Volume II (page 128) and Volume IV (page 181):

```
;---------------------------------------------------------------
; FUNCTION DoItTheOldWay(ioDrvNum: INTEGER): integer;
;   returns:    0 = is ejectable
```

```
;               -1 = is not ejectable
;                1 = error occurred (could not find specified drive)
;--------------------------------------------------------------

DoItTheOldWay   PROC

            MOVE.L  (sp)+,A0            ; save the return address
            MOVE    (sp)+,D1            ; get target drive #
            MOVE.L  DrvQHdr+QHead, A1   ; Point to the drive queue

@0           CMP.W   DQDrive(A1), D1     ; is it the right drive #?
            BEQ.S   @1                  ; if so, skip out

            MOVE.L  QLink(A1), A1       ; Get the link to the next vol
            MOVE.L  A1, D0              ; is it NIL?
            BNE.S   @0                  ; if not, go back.
            MOVE.W  #1,(SP)
            JMP     (A0)
@1
            CMP.B   #8,-3(A1)           ; ejectable value??
@5           SGE     (SP)                ; record if it isn't ok
            JMP     (A0)

            ENDP
```

X-Refs:
“File Manager,” *Inside Macintosh* Volumes II and IV
“Device Manager,” *Inside Macintosh* Volumes II and IV