ATA Device 0/1 Software Developer Guide

3/9/98
Technical Publications

Contents

Figures

About This Note

This developer note describes the Macintosh ATA Device 0/1 system software that provides support for two ATA devices on a single ATA bus. It also defines the changes that application and driver clients need to be aware of to take advantage of the ATA Device 0/1 architecture.

This note is intended to help experienced Macintosh hardware and software developers design Macintosh compatible drivers and applications that take advantage of the ATA Device 0/1 architecture.

If you are not already familiar with Macintosh computers or would simply like more technical information, you may wish to read the related technical documentation listed in the section “Supplemental Reference Documents.”

This note is published only in electronic form, as an Adobe™ Acrobat™ PDF (portable document file). The file is available from two sources:

- on the World Wide Web at <http://devworld.apple.com/dev/devnotes/dntable1.html>
- on the Reference Library Edition of the Developer CD Series, which is distributed as part of the monthly mailing to registered developers

To use the information in this developer note, you should already be familiar with the writing device drivers for the Macintosh computer.

Contents of This Note

This note includes one chapter:

- Chapter 1, “ATA Device 0/1 Software,” provides an overview of the Macintosh ATA Device 0/1 system software architecture and describes the changes developers need to make in their application and driver code to use the features provided by the ATA Device 0/1 architecture.

Because this note is so short, it does not include an index.

Supplemental Reference Documents

ATA Device Software Guide, available on the Apple DeveloperWorld web site.

AT Attachment Interface for Disk Drives, ANSI X3.221-1994, approved May 12, 1994

AT Attachment Interface with Extensions (ATA-2), ANSI ASC X3.279-1996, revision 3, proposed American National Standard 948D

AT Attachment-3 Interface (ATA-3), ANSI ASC X3.298-199x

AT Attachment-4 Interface (ATA-4), X3T13 draft

ATA Packet Interface for CD-ROMs, SFF-8020, revision 1.2, June 13, 1994

Western Digital Enhanced IDE Implementation Guide, by Western Digital Corporation, revision 5.0

Fast ATA Sourcebook, Quantum Corporation, November 1994

Enhanced Disk Drive Specification, by Phoenix Technologies Ltd., version 1.1, January 95

Developers should also refer to Apple developer technotes which are available on the Apple Developer World Wide web site. The following technotes are of particular interest to developers interested in working with ATA devices.

Technote 1094, Virtual Memory Application Compatibility, which describes virtual memory interaction with the ATA Manager.

Technote DV 22, CD-ROM Driver Calls, which discusses ATA CD-ROM driver calls.

ATA Demo sample code demonstrates how to call the ATA Manager and scan the ATA bus is also available on the Apple Developer World Wide web site.

Developers should also have copies of the relevant books of the *Inside Macintosh* series, available in technical bookstores and on the World Wide Web at

<http://gemma.apple.com/dev/insidemac.shtml>

You should also have *Designing PCI Cards and Drivers for Power Macintosh Computers*. These books are available in technical bookstores.

Conventions

This developer note uses the following typographical conventions.

Typographical Conventions

Computer-language text—any text that is literally the same as it appears in computer input or output—appears in `Letter Gothic` font.

Hexadecimal numbers are preceded by a dollar sign ($). For example, the hexadecimal equivalent of decimal 16 is written as $10.

Note
A note like this contains information that is interesting but not essential for an understanding of the text. ◆

IMPORTANT
A note like this contains important information that you should read before proceeding. ▲

C H A P T E R 1

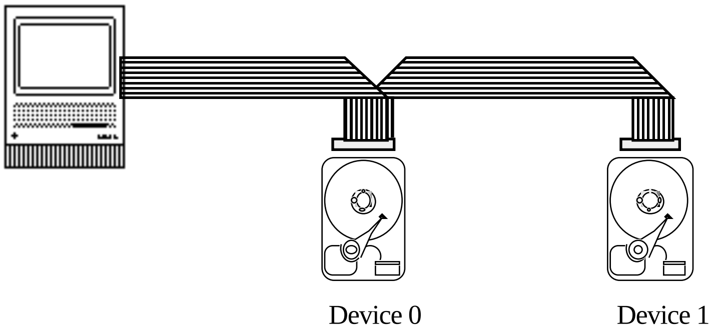
ATA Device 0/1 Software

This chapter describes the Macintosh ATA Device 0/1 system software that supports two ATA devices on a single ATA bus. It also defines the changes that application and driver clients need to be aware of to take advantage of the ATA Device 0/1 architecture.

Overview

Prior to the introduction of the ATA software described in this note, Macintosh products were restricted to a single device per ATA bus. Software support for two daisy-chained devices on an ATA bus is a new feature incorporated into the ATA Manager software. A two-device configuration on a single ATA bus is called an ATA Device 0/1 configuration, as defined in the ATA-3 specification. Figure 1-1 shows two daisy-chained ATA devices in an ATA Device 0/1 configuration.

Figure 1-1 Daisy-chained configuration for ATA Device 0/1



ATA Device 0/1 is an enhanced version of the old ATA-1 Master/Slave specification which enables up to two daisy-chained devices on the bus. The Master/Slave nomenclature was changed to ATA Device 0/1 in the ATA-2 specification and is still referred to as ATA Device 0/1 in the ATA-3 specification. The Macintosh ATA Device 0/1 implementation is based on the definition of ATA Device 0/1 in the ATA-3 specification.

The Master/Slave and the ATA Device 0/1 implementations are conceptually the same. A master device and a Device0 device are the same device, both have device ID 0, and a slave device and a Device1 device, both have device ID 1. The master device, Device0, must still sense and report the status of the slave device, Device1, at Reset and with the Execute Diagnostics command. Both devices receive commands, only the selected device, specified by the DRV bit in the Drive/Head register, responds. Where the two specifications differ is the Master/Slave specification permits only two configurations: a single device 0 configuration, or a device 0 and a device 1 configuration. ATA Device 0/1 however, supports both of the previously mentioned configurations and a device 1 only configuration. ATA Device 0/1 also improves the communication between the devices during Reset and Execute Diagnostics operations.

Supported Devices

The following section defines the device requirements for support of ATA Device 0/1 in Macintosh computers.

The devices supported by the ATA Device 0/1 configuration include ATA-3 compliant ATA and ATAPI hard disks, CD-ROM, DVD-ROM, and removable media devices. Each device should be able to operate as either a Device 0 or Device 1 as described in the ATA-3 specification.

In general, the Macintosh ATA architecture allows use of all devices that follow the ATA-3 specification. However, full support is only guaranteed for those devices qualified and installed in Macintosh systems built by Apple Computer.

IMPORTANT

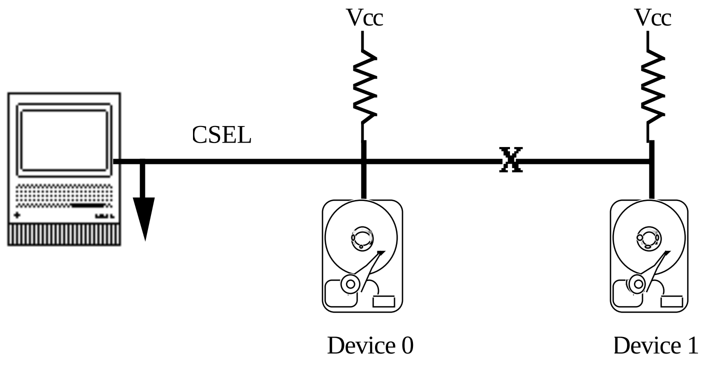
Devices on the ATA bus are assigned an identifier (ID) of either 0 or 1 using either the ID jumpers on the device, or by using the Cable Select method. Using the ID jumpers, a drive can be assigned a specific ID independent of its physical connection to the bus. At this time Macintosh computers that support ATA Device 0/1 require that the ID jumpers be set on the device. The Cable Select method is also supported in software, but not implemented with the proper cable at this time.

Making the Device 0/1 Cable Select Connection

This section provides some details about how a Cable Select hardware configuration would be set up. The information is for your edification only, since jumper ID select is the method currently used in Macintosh computers.

When devices are set up using the Cable Select method, the device senses the CSEL signal and responds to requests for ID 0 if the signal is grounded, or it responds to requests for ID 1 if the signal is high. The host computer grounds CSEL at its logic board connector, and each drive uses a pull-up resistor at the drive end. The ID numbers are assigned by either closing or opening the CSEL conductor at each device connector. Figure 1-2 illustrates two ATA devices in a typical Cable Select configuration.

Figure 1-2 Cable Select configuration



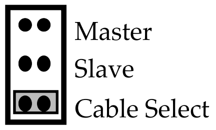
Of the two methods used for setting the device IDs, Cable Select is easier for customers to configure because it relieves them from determining how to set the ID jumpers correctly for each device. To support the Cable Select method both devices are configured the same, a jumper on the device is set to enable sensing the CSEL signal, which allows customer to simply connect either device to either connector on the Cable Select cable.

Note
The ID jumpers can be used to override the sensing of the CSEL signal, which would allow a customer to use jumper ID select in a system with a Cable Select cable.

Device ID Jumpers

Most ATA and ATAPI devices have three jumper connections which represent the ID options Master, Slave, and Cable Select. Only the Cable Select option should be set when the Cable Select method is desired. Figure 1-3 below shows an example of a device configured for Cable Select. If a device does not support Cable Select or if Cable Select is not desired, the appropriate jumper on Master or Slave should be set depending upon the ID of the connector (Master for 0, Slave for 1) the device will be plugged into.

Figure 1-3 Device configuration jumpers set for Cable Select



Always consult the manufacturer's documentation, because the labeling and terminology used for the jumper connectors may vary.

Cable Select Cables

A special cable is also required to use the Cable Select feature supported by ATA and ATAPI devices available today. The I/O cable connecting the two devices to the controller must conform to the Cable Select specification in the ATA-3 specification. The total length of cable should not exceed 18 inches (.46m) from the farthest connector to the input of the controller. The length includes trace runs on the motherboard.

All recommended termination described in section 4.3 of the ATA-3 specification should be implemented. This includes a 10K Ω pull-down resistor on the DD7 signal for the recognition of the absence of devices.

IMPORTANT

As of the writing of this document, Macintosh computers do not include cables that support the Cable Select method for device identification in an ATA Device 0/1 configuration. Devices must be jumpered as Master and Slave according to the position of the device on the cable. Jumper location and configuration information should be provided in the documentation available from the manufacturer of the device.

Software Architecture and Client Requirements

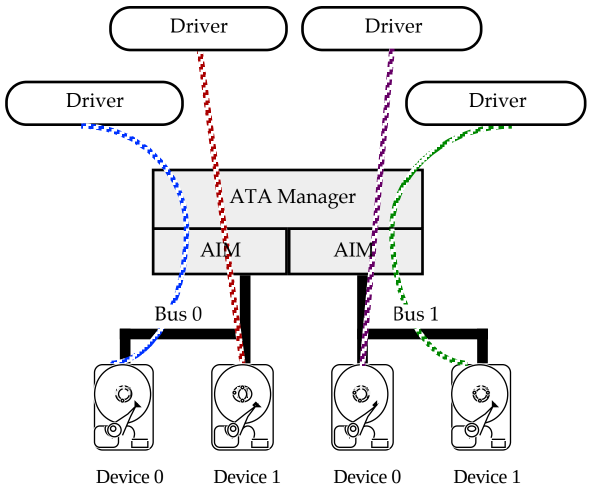
The following section describes the modifications to the ATA software architecture, the new features, and changes required from driver and application clients to support ATA Device 0/1 in Macintosh systems.

Software Architecture

The software architecture for ATA Device 0/1 is similar to the single device per bus configuration currently implemented for ATA devices, except that there is an additional I/O path for each bus. The ATA Manager has been changed to become more abstract with regard to devices rather than buses. However, drivers and application clients of the ATA Manager will still view each device independently. There should not be a valid reason for drivers to differentiate between a device being at ID 0 or ID 1.

Figure 1-4 shows a hierarchal diagram of the ATA Device 0/1 software architecture.

Figure 1-4 ATA Device 0/1 software architecture



There should be no changes required for software components above the driver level, except for applications, such as formatting utilities and test tools which bypass the driver and communicate directly with the ATA Manager.

ATA Device 0/1 Driver Compliance Signatures

The following section describes a set of signatures a driver must include to indicate full compliance with the ATA Device 0/1 architecture described in this document. Drivers that do not include the driver compliance signatures are assumed to behave improperly and are not used with device 1. However, a driver that does not have these signatures may still be used with device 0 in certain configurations.

Device1 Booting Signature

Older non-compliant drivers do not work correctly in configurations where device 1 is present, either with or without device 0. For example, if a non-compliant driver is loaded from device 1, the driver will most likely assume it is using device 0 instead. To avoid such problems, the ATA Driver Loader checks for the compliance signature `$D0D1` before opening the driver. See "Booting From Device 0/1" (page 1-20) for additional details about the ATA

Driver Loader. The presence of `$D0D1` at offset `$06` from the start of the driver indicates the driver is fully compliant with the ATA Device 0/1 specification. Shown below is sample 680x0 assembly code from a typical driver compliant with ATA Device 0/1.

```
$00     BRA.W       PrimaryBoot         ; primary boot entry
$04     DC.W        0                   ; reserved
$06     DC.W        $0D0D1              ; compliant with ATA Device 0/1
$08     BRA.W       SecondaryBoot       ; secondary boot entry
$0C     …
```

DriverGestalt Signature

Device drivers must also return the correct `DriverGestalt` response so that other software can determine if the driver is compliant with ATA Device 0/1. The `DriverGestalt` selector and response are as follows:

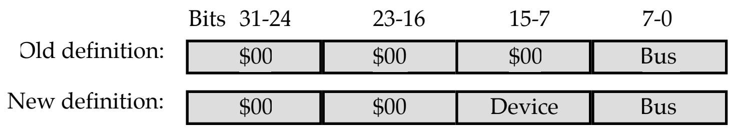
```
#define kdgATADev1      ‘dev1’      // Selector for Device 0/1 support
struct DriverGestaltATADev1Response
{
     unsigned long      dev1Support;    // 1 = supports devices 0 and 1
};

// This macro returns a pointer to the DriverGestaltATADev1Response.
// The input is a pointer to a DriverGestaltParam.

#define GetDriverGestaltATADev1Response(p)
    ((DriverGestaltATADev1Response *)(&((p)->driverGestaltResponse)))
```

Expanded Device ID Support

Access to ATA and ATAPI devices is provided by the ATA Manager using the `ataPB` data structure. Within the current structure is the device identifier `ataPBDevID`, a 32 bit value which represents a logical bus number. The device number is always 0, since only one device per bus is supported.

To support ATA Device 0/1, the structure of `ataPBDevID` has been modified to include a device number as well as the bus number (the device number has always been defined in the structure, but never truly used). The `ataPBDevID` field remains a 32-bit structure (`type UInt32`), but it changes internally. Older versions of the ATA Manager supported only one device per bus, which was device 0. The new `ataPBDevID` structure is backwards compatible with that limitation as shown in Figure 1-5.

Figure 1-5 Old and new ataPBDevID comparison



```
typedef struct          // ataDeviceID structure
{
    UInt16  Reserved;   // The upperword is reserved (0)
    UInt8   devNum;     // device number (0 or 1)
    UInt8   busNum;     // bus number
} ataDeviceID
```

Although `ataPBDevID` remains defined as a 32-bit number, drivers and applications can typecast it to the `ataDeviceID` structure to determine the bus number and device number for a specific ATA or ATAPI device. However, clients should not assume the number of buses, the number of possible devices per bus, or associate a particular ID with a fixed location. For example, the internal device may not always be at bus 0, device 0. It is also suggested that clients do not assume only device 0 is available.

Device Selection via Device ID

Even though previous versions of the ATA Manager supported only one device per bus (device 0) the client was still required to indicate the device to be selected by setting or clearing `DRV` (bit 4) in the Device/Head register value (`ataPBTaskFile.ataTFSDH`) passed in with the operation, see Figure 1-6. Clearing `DRV` selected device 0, setting `DRV` selected device 1. Setting and clearing this bit is still required to support the ATA Device 0/1 configuration.

Figure 1-6 Setting the DRV value in ataTFSDH

Bit	7	6	5	4	3	2	1	0
	1	LBA	1	**DRV**		HEAD		

To accommodate old drivers which are not aware of device 1 or do not set `DRV`, the ATA Manager now forces `DRV` to a value of one (`1`) if the `ataPBDevID` value indicates device 1 and `DRV` is zero (`0`). However, a `DRV` value of 1 with an `ataPBDevID` value indicating device 0 is considered a `paramErr` (`-50`). All other bits of the `ataTFSDH` value are passed to the device as set by the client.

Booting From Device 0/1

Current systems that support ATA devices assume that only device 0 is present, and only boot from device 0. To enable booting from device 1, the following Macintosh software components required change: the PRAM definition of StartUpDisk, the Start Manager code, the ATA Manager, the ATA Driver Loader, and the device driver.

Changes To StartUpDisk PRAM Values

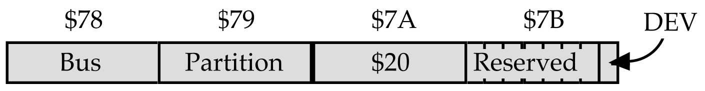
The StartUpDisk (PRAM $78-$7A) value for ATA devices is currently defined as shown in Figure 1-7. A single byte at $78 defines the device. Byte $7A (Arch/Tech) is a fixed value for ATA/ATAPI, and byte $7B is reserved.

Figure 1-7 Current StartUpDisk definition for ATA/ATAPI

$78	$79	$7A	$7B
Bus	Partition	$20	$00

To support booting from device 1 the device number has been added to the StartUpDisk definition at the previously unused byte $7B. The device number is defined in bit 0 (0 = device 0, 1 = device 1), and bits 1-7 are reserved (0), see Figure 1-8. The new StartUpDisk definition is backward compatible with current systems where $7B is reserved.

Figure 1-8 New StartUpDisk definition for ATA/ATAPI



The Start Manager code uses the new StartUpDisk definition to determine the device to boot from and when loading the drivers. The Start Manager calls the ATA Manager and ATA Driver Loader with the new device identifier format with the bus and device numbers to determine if the device is present and to load the appropriate driver.

ATA/ATAPI device drivers must use the new StartUpDisk definition to be able to boot properly in an ATA Device 0/1 configuration. When the Startup Disk control panel selects the startup disk, it makes a Status call (DriverGestalt ‘boot’) to the driver to obtain the StartUpDisk value. The DriverGestaltBootResponse value returned must match the new definition to enable booting from either device in an ATA Device 0/1 configuration. For most drivers, the only change is to return the device number in the DriverGestaltBootResponse.SIMsRSRC instead of 0 as is done today.

Driver Loading and Opening

When called to load and open a driver, the ATA Driver Loader component verifies whether the driver is compliant with the current device configuration before using the driver. Any ATA configuration with device 1, either with or without device 0, requires the drivers to be compliant with the ATA Device 0/1 specification. The driver must have the compliance signature $D0D1 to be loaded, see “ATA Device 0/1 Driver Compliance Signatures” (page 1-17). The compliance signature is not required for device 0 only configurations. Older drivers can be used in that case.

After the ATA Driver Loader loads a device driver, it opens that driver by calling the driver’s boot entry routine located at either offset 0 (Primary) or offset 8 (Secondary) of the start of the driver. Upon entry to the driver the D5 register contains the boot ID. The D5 definition is shown in Figure 1-9.

Figure 1-9 D5 register definition for ATA/ATAPI drivers

Primary (offset 0):

Bits 31-24	23-16	15-7	7-0
Flags	$00	$00	Bus

Secondary (offset 8):

Bits 31-24	23-16	15-7	7-0
Flags	Bus	Device	$00

The D5 definition differs depending upon whether the Primary or Secondary boot entry is called, as shown in Figure 1-9. The Primary boot entry is the old style method which has little information other than the bus number. The Primary boot entry is no longer used for ATA/ ATAPI devices in an ATA Device 0/1 configuration. The D5 definition for the Secondary boot, however, is complete in that it can support both the bus number and device number. Support for ATA Device 0/1 requires the correct bus and devices values be passed to the driver to permit booting from either device 0 or device 1.

The ATA/ATAPI device driver must now recognize both bus and device values in D5 and transform them into the new ataDeviceID structure for communication with the ATA Manager. If the driver does not recognize the Device value (for example a non-compliant ATA Device 0/1 driver is present) then it may try to boot from Device 0 when Device 1 is the intended device. This could be a significant problem, especially if Device 0 already has a driver loaded.

Independent Device Timing

For efficient performance on the bus, an independent set of bus mode and timing parameters are maintained for each device. If instead, a single mode and timing parameter were used for two different devices, one device could be limited to the fastest mode/timing common to both devices. For example, if one device runs only at PIO Mode 0 and the other can run at both PIO Mode 3 and Multiword DMA, then both devices must run at PIO Mode 0.

The ATA Manager/AIM is responsible for remembering the current bus mode/ timing parameters for each device and for programming the controller for each device's bus mode/timing. The AIM ensures the correct timing is enabled for any device before allowing access to the device. Note there may be an issue with mixing very fast and very slow devices where the slower device may not work correctly when programming the ATA registers in faster PIO modes. However, this can be corrected by always programming the registers in a slow PIO mode and switching to the faster mode for data transfers.

The ATA/ATAPI driver is responsible for determining the optimal mode/timing for its associated device and for communicating this to the ATA Manager. The driver uses the ATA Manager's Set Drive Configuration (`kATAMgrSetDrvConfiguration`) and Get Drive Configuration (`kATAMgrGetDrvConfiguration`) functions to set and get the current mode/timing parameters.

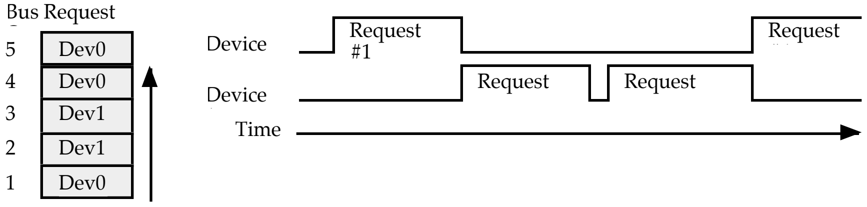
Bus Management

The ATA Manager/AIM is solely responsible for managing traffic on the ATA bus. Previous versions had it easy in handling only a single device per bus, but now with two devices more stringent traffic control is required.

For ATA Device 0/1 support the ATA Manager/AIM closely monitors the bus and allows only one device to be active at any given time. The ATA protocol does not allow concurrent device operation. Once any device is selected and begins a task, the other device cannot be selected until the first device completes the task. If a request from a client driver is received while the bus is busy the request is queued and deferred until the bus is free.

The ATA Manager/AIM contains a single request queue per bus so there is no competition for the bus between devices. Requests for both devices on the bus are queued and executed in the order received. The diagram shown in Figure 1-10 illustrates the order in which device requests are executed. The Device0 and Device1 signals shown represent a request being processed rather than the BSY state of the device.

Figure 1-10 Bus request execution order for ATA Device 0/1



The execution order shown in Figure 1-10 applies to a series of typical non-immediate requests. Immediate requests are put at the head of the queue and affect the execution order. It is expected that immediate requests will be used only in certain circumstances such as error handling or a page fault conditions where queued requests must be held off until some other action occurs first.

Reset Handling and Notification

With the single-device-per-bus implementation, the handling of a soft reset (SRST) is straightforward. Only one device is affected. The driver has the responsibility of issuing the reset request and reconfiguring the device before issuing further requests to the device. The ATA Manager/AIM simply initiates the reset (either an ATA SRST or an ATAPI Device Reset) and assumes the bus is free afterwards.

However, with two devices on the bus, bus reset becomes a bit more complex. First, both devices may be affected. Second, since not all devices retain their parameters after soft reset, it is assumed the device must be reconfigured. It is therefore necessary to make the drivers of all devices on the bus aware of the reset. In addition, critical handling of the reset is required since there may already be requests queued for both devices or new requests may come in before the devices have been reconfigured.

New Reset Event

Since version 2.0, the ATA Manager has had a method for notifying drivers of events affecting the bus. The method is called ATAEvents, these events include notification of a new device coming online (ready for use), going offline, and device removal. Also included, but not implemented, was a reset event. With ATA Device 0/1 the reset event is implemented. The following ATAEvents are supported.

```
kATANullEvent     = 0x00,     /* Just kidding -- nothing happened */
kATAOnlineEvent   = 0x01,     /* device has come online */
kATAOfflineEvent  = 0x02,     /* device has gone offline */
kATARemovedEvent  = 0x03,     /* device has been removed from the bus */
kATAResetEvent    = 0x04,     /* The device received an ATA reset */
kATAPIResetEvent  = 0x0A,     /* The device received an ATAPI reset */
```

A driver registered for a device must register for either the `kATAResetEvent` and/or the `kATAPIResetEvent` to be notified when its device has been reset. This is important since software other than a driver may request to reset any device. ATAPI drivers should register for both reset events since ATAPI devices may be affected by SRST and the notification allows the drivers the option of either taking action or disregarding the event depending upon the device's needs. ATA drivers should register for only kATAResetEvent since ATAPI reset does not affect ATA devices. Registration for a reset events is accomplished using the ATA Manager functions Register Driver (`kATAMgrDriveRegister`) and Modify Event Mask (`kATAMgrModifyEventMask`). Refer to the ATA Manager documentation for details on registering, modifying, and receiving ATAEvents.

ATA Manager Reset Handling and Notification

When an Reset Request (`kATAMgrBusReset`) is received by the ATA Manager, the ATA Manager initiates either a ATA soft reset (SRST) or an ATAPI Device Reset depending upon the protocol of the request (refer to definition of `ataPBFlags`). If there is another device on the bus and it is executing an operation, the reset is delayed until the operation completes. If delaying the reset results in a timeout, then the reset request aborts and a timeout error is returned. If the intended device is busy, the reset is still issued.

After the reset has been sent and the device becomes ready again, the ATA Manager resets the bus timing to the default timing, and issues a notification to all of the drivers of devices affected and registered for the type of reset that occurred. The notification occurs synchronously (nested) within processing of the reset request.

The ATA Manager resets the bus timing parameters for each device affected by the reset since the device may not retain the timing it was previously configured with. The default timing is PIO Mode 0.

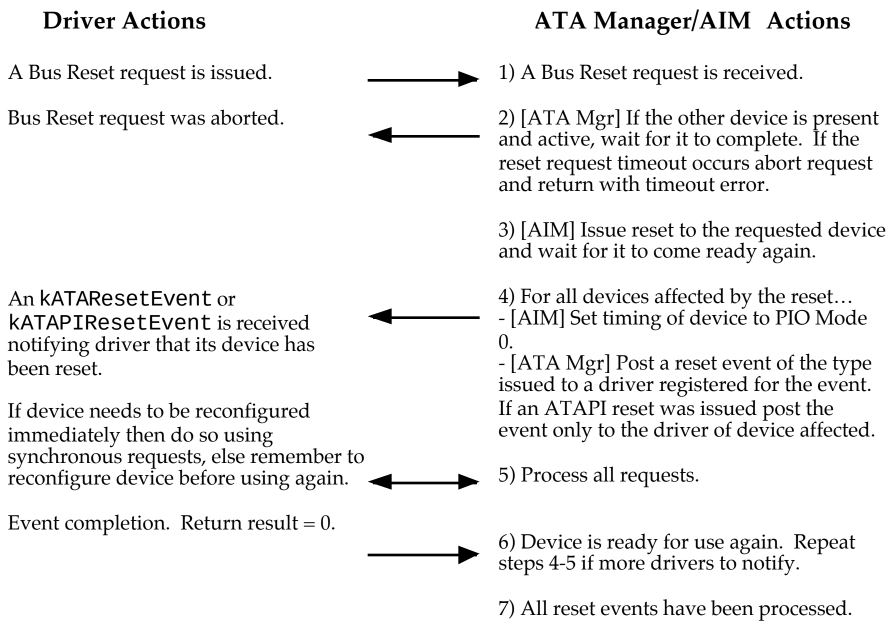
If the reset was an SRST all drivers registered for `kATAResetEvent` are notified. However, if the reset is ATAPI Device Reset the ATA Manager issues the `kATAPIResetEvent` notification instead only to the driver of the affected device. Reset events are generated only for soft resets and not for power on and hard reset1.

All notified drivers are expected to take appropriate action for the reset event and return a result of zero (0) to indicate both acknowledgment of the event and that the device is available for use again.

Note
Because some ATAPI devices do not respond correctly to an ATA soft reset (SRST), the ATA Manager issues an ATAPI Device Reset to the ATAPI device first, followed by an ATA soft reset. The ATAPI driver is notified of only the ATA soft reset (`kATAResetEvent`) and should treat both reset events the same.

Figure 1-11 illustrates the Reset handling sequence for both the driver and the ATA Manager.

Figure 1-11 ATA Manager and device driver reset handling sequence



Driver Actions

A Bus Reset request is issued.

Bus Reset request was aborted.

An `kATAResetEvent` or `kATAPIResetEvent` is received notifying driver that its device has been reset.

If device needs to be reconfigured immediately then do so using synchronous requests, else remember to reconfigure device before using again.

Event completion. Return result = 0.

ATA Manager/AIM Actions

1) A Bus Reset request is received.

2) [ATA Mgr] If the other device is present and active, wait for it to complete. If the reset request timeout occurs abort request and return with timeout error.

3) [AIM] Issue reset to the requested device and wait for it to come ready again.

4) For all devices affected by the reset…
- [AIM] Set timing of device to PIO Mode 0.
- [ATA Mgr] Post a reset event of the type issued to a driver registered for the event. If an ATAPI reset was issued post the event only to the driver of device affected.

5) Process all requests.

6) Device is ready for use again. Repeat steps 4-5 if more drivers to notify.

7) All reset events have been processed.

Driver's Handling of Reset Notification

When a driver receives an `kATAResetEvent` notification through the ATAEvent mechanism, it has the option of either immediately taking action (such as reconfiguring the device) or noting the event occurred and deferring its processing. It is recommended the driver defer processing of the reset unless it has outstanding requests to the device which have been queued or pending (of course, the reset request itself should not be considered). In either case, the driver must return a status of zero (0) for the event. Refer to the Driver Actions section in Figure 1-11 for the recommended processing of a reset event.

If a driver decides it needs immediate processing of the reset event because of queued requests, etc., it should perform the minimum tasks necessary to bring the drive back to a state suitable for use and defer the remaining tasks until later. This minimizes the delay associated with enabling the bus for use again.

Any request made to the ATA Manager by the driver for immediate processing of a reset event must be done synchronously and in the immediate mode (see definition of `ataPBFlags`) to avoid queuing a new request which could potentially cause deadlock.

Handling Bus Hangs

Another complex issue with Device0 /1 is how to manage the ATA bus when the bus is blocked due to a non-responsive (hung) device. This can occur when the device loses sync with the host (DMA underrun, for example) or if it has a firmware problem and is stuck busy. A hung device creates a problem because it prevents the other device on the same bus from being used. After the request which caused the device to hang times out, the ATA Manager executes the next request in the queue. If the bus is hung (the device has BSY set), that request can't proceed until the device can respond again. In this situation, the second request times out, and the next request, and so on until the device is free again (either by itself, or from a request such as reset which clears the hung device).

It may appear that the ATA Manager should initiate the reset, because it is responsible for bus management. However, the ATA Manager has no context with which to determine what the request made was and doesn't know if the device is hung or simply executing a lengthy task. Therefore, it is the responsibility of the device driver to clear a hung device, usually with a reset request.

When clearing a hung device it is recommended that the driver reset the device using an immediate call as soon as possible to avoid blocking access to all devices on the bus. A hint that the bus may be hung occurs when a timeout (`ATATransTimeOut`) error is returned.

Compatibility Issues

Issues that pertain to ATA/ATAPI device drivers, applications, and test tools used in an ATA Device 0/1 configuration are briefly defined in this section.

Older ATA/ATAPI Device Drivers

Older ATA/ATAPI device drivers that do not support ATA Device 0/1 (non-compliant drivers) may present compatibility problems. The possible problems are as follows:

- Non-compliant drivers should work with minimal problems on device 0, but they may not recognize and be able to work with device 1. A significant issue exists if an old driver is installed on the media for device 1 and loaded. This issue applies to removable media which may be initialized on device 0 and later used on device 1.
- The old driver may incorrectly assume device 0 is the target when opened and try to take control of it. Some protection is provided when the driver registers for control of the device (it will fail if another driver is in control of the same device). However, this assumes the driver registers properly and does not ignore the error from registration. There is no protection if device 0 does not have a driver registered, yet. The ATA Driver Loader protects against old drivers being used with device 1 by not loading these drivers from the media. However, the protection does not apply to drivers loaded from extensions.
- Non-compliant drivers may not be aware of their device being reset. This risk already exists in current Macintosh systems because reset notification is not implemented. But, the use of reset is a rare occurrence after the driver is loaded, so the risk of problems associated with undetected reset is minimal.

Older Applications and Test Tools

The risks for applications and test tools are essentially the same as those for device drivers with the following exceptions:

- Formatting utilities may install a non-compliant driver on the media. This is the same problem described in item 1 of the in the drivers section above if the media is removable or the device ID changes (from 0 to 1).
- The new definition of StartupDisk (PRAM) may be a problem with applications or diagnostics utilities that expect certain values. For additional information about the changes to StartupDisk PRAM, see "Changes To StartUpDisk PRAM Values" (page 1-20).

ATA Device 0/1 and Hot-Pluggable Devices

An ATA Device 0/1 configuration should not be used with hot-pluggable devices, for example in the PowerBook Media Bay. The ATA-3 specification does not provide proper isolation for hot-plugging with multiple devices on the bus. The PCMCIA standard does support hot-plugging, albeit with a single device per bus.

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Line art was created using Adobe™ Illustrator and Adobe Photoshop.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Adobe Letter Gothic.

WRITER
Steve Schwander

Special thanks to Kory Hansen, Ben Koning, Rich Kubota, and Rich Schnell