New Technical Notes

Macintosh

Developer Support

Apple Desktop Bus Q&As

Hardware

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.

New Q&As for this month:
System 7.1 and ADBReINIT extension

System 7.1 and ADBReINIT extension
Date Written: 11/18/92
Last reviewed: 3/1/93

Now that I've upgraded my Macintosh Quadra 950 to System 7.1, do I still need the ADBReInit bug fix that's posted in AppleLink or was this ADB problem fixed in System 7.1?

—

The ADBReInit bug wasn't fixed in 7.1, so you probably should continue to use the ADBReINIT extension.

The ADBReInit extension prevents memory from being moved at interrupt time, which could cause the PowerBook 140 and 170 models, as well as the Macintosh Quadra 700, 900, 950 models, to hang or crash if ADBReInit is called. Informal testing indicates that it's more likely to happen while AppleShare is active (file sharing is turned on).

ADBReInit normally is never called on a Macintosh Quadra because connecting ADB devices while the CPU power is on is discouraged. There is a specific instant that the PowerBook uses the call: When coming out of Sleep mode, ADBReInit is called to determine whether or not external ADB devices are attached. Also, the ADBReInit section of the Macintosh Technical Note “Space Aliens Ate My Mouse (ADB—The Untold Story)” suggests instances where this call might be used, as does the MacDTS sample code TbltDrvr, referenced in the ADBReInit

section of *Inside Macintosh*'s ADB chapter. So, though most developers won't be affected by the ADBReInit bug, it could affect some developers.

PowerBook 140 & 170 ADB low-level input voltage maximum
Date Written: 2/20/92
Last reviewed: 4/22/92

Our ADB (Apple Desktop Bus) device works on all Macintosh models except our PowerBook 170. Are there any PowerBook ADB changes?

—

On the early PowerBook 140 and 170 versions, the mouse was not properly grounded and as a result the low-level input voltage maximum was not within the 0.8 volt spec. Instead, it was closer to 1.5 volts. So if your device depends on the voltage being within the spec, it will have problems with these early PowerBooks. This problem has been fixed on the currently shipping machines, but if you or one of your users has a problem, an Apple Service rep will make a minor modification to the PowerBook to make it work correctly.

Look for multiple ADB device collisions
Date Written: 1/23/92
Last reviewed: 3/20/92

Our Apple Desktop Bus (ADB) device has problems with Apple mice in the serial number series APxxxxxxxxxx). A mouse in this series does not detect another mouse class device's talk register 3 under all conditions. To explain in detail: The mouse responds 220 microseconds after the stop bit of the talk register 3 command. Our device responds from 180 to 190 microseconds from the stop bit. If it responds to the talk register 3 command later than 183 microseconds (from 183 to 190), mouse doesn't detect that another ADB device has started a transmission. Consequently, the mouse will start its own response at 220 microseconds by setting the data line low. This has the effect of adding an extra 1 bit to the data stream and the mouse winds up with a Device Type of 0 to -128 as reported by the ADBLister application. What is the Apple approved workaround for this condition?

—

The best way for you to handle any problems which you have with this mouse is to always look for collisions (which you should do anyway). If you detect a collision, let go of the bus. This way, your product will in the end work better than the mouse and not get moved around, like the mouse will. Even if you are in the middle of a transaction, you should just let go of the bus.

PowerBook Control and Caps Lock keys can't be remapped
Date Written: 1/27/91

Last reviewed: 2/6/92

Is there any way to make the Caps Lock key behave as if it were the Control key on a PowerBook 170? I've tried playing with 'KCHR' resources, but I can't get the Caps Lock key to *not be* a sticky control key. The PowerBook 170 is great, except for the placement of the Control key.

—

For a long time the Macintosh ADB keyboards always had a mechanical Caps Lock key, so there's never been any special handling of switching between keycodes while the Caps Lock key is down or up.

With the introduction of the PowerBook keyboards with nonmechanical Caps Lock keys, the Power Manager chip itself is handling this remapping. Thus there is no keyup sequence that is sent from the keyboard, which makes it impossible to write an ADB handler that remaps the keys. Also, there's no way to change the behavior of the keyboard inside or outside the driver, because the keymap handling is hardwired to the Power Manager chip.

So it's not possible to remap the Control and Caps Lock keys with the capability of handling Caps Lock down keycodes. The mapping of keys themselves is trivial, but the handling of different key codes is not.

Multiple Mac ADB devices with same original address & ADBS
Date Written: 12/11/91
Last reviewed: 12/11/91

Our ADB device uses an 'ADBS' (7) resource, because we contacted Software Licensing and were assigned a default address of 7. If another developer's device at default address 7 has an 'ADBS' resource, how would the ADB Manager handle this?

—

The ADB Manager doesn't have much to say about this. The Resource Manager doesn't allow two resources of identical type and ID in the same resource file, so the service routines for two devices as described cannot be installed in the System file at the same time. This is a limitation of the Macintosh Apple Desktop Bus implementation which is described in the Macintosh Technical Note "Space Aliens Ate My Mouse (ADB—The Untold Story)."

There really is no way around this. Since all of the addresses have been assigned multiple times to various developers, just changing your device's address will not solve the general problem. We wish that there was a solution, but this situation is inherent in the design of the ADB Manager.

If your product has an address conflict with another third-party ADB device, a conflict which is impairing your customers' use of your device, DTS will try to help you find a suitable workaround. As few users have multiple custom ADB devices, this limitation is generally not a problem.

Programmer's Key 1.4.1 and PowerBooks 140 & 170
Date Written: 11/14/91
Last reviewed: 12/12/91

Is there a version of Programmer’s Key for the PowerBook 140 and 170? The physical switches are a real pain to use.

—

The Programmer’s Key 1.4.1 runs fine on the PowerBooks. There’s only one catch. Since you need the power (startup) button to activate the Programmer’s Key, you need to hook up a

keyboard to the rear of the PowerBook. The Programmer's Key 1.4.1 is on the latest *Developer CD Series* disc. Here's the path: Tools & Apps (Moof!): Intriguing INITs/cdevs/DAs: Programmer's Key 1.4.1.

Apple Desktop Bus (ADB) startup timing
Date Written: 11/6/91
Last reviewed: 11/6/91

I need exact Apple Desktop Bus (ADB) startup timing data for each Macintosh model. When does the ADB query to devices first go out and when does it receive confirmation of a device present? When does the File Manager start loading the control files? What type of signal is sent and received--a pulse of what parameters?

—

It's not possible to tell you when ADB devices are initialized after power-up with any accuracy, because it depends on factors such as how much RAM is in the machine and what kind of screens are connected to it. When the system begins loading control files, it's even more variable, depending on things like disk speeds and if the user is booting off a floppy. In general, timing issues are difficult with the Macintosh, as it doesn't even approach being a real-time machine.

Regarding the characteristics of the ADB startup sequence, the best thing to do is get the ADB Specification from Apple's Software Licensing; it contains all the information you need (along with *Inside Macintosh* and the Macintosh hardware reference) to build an ADB device.

ADBReinit clarification
Date Written: 6/7/91
Last reviewed: 10/15/91

ADBReinit doesn't seem to work exactly as described in the Apple Desktop Bus (ADB) documentation. Please clarify.

—

A SendReset command byte is never sent across the bus, as *Inside Macintosh* and other documentation seems to imply. In fact, when the ADB Manager is asked to SendReset, it actually instructs the ADB transceiver to assert an Global Reset pulse.

The Talk R3 sequence after a reset is as follows: each of the 16 device addresses is polled once, and then the ADB Manager returns to the addresses that did not time out (addresses which contain at least one device) and retries those addresses (moving devices between default and temporary addresses) multiple times (50 in the current implementation) until it is satisfied that devices have separated properly.

This initialization task is patched out by System 6.0.5 and later, so every machine can use the new, improved ADB Manager which debuted with the Macintosh IIci ROM.

Minimizing ADB Manager address resolution collision potential
Date Written: 6/7/91
Last reviewed: 10/15/91

Apple offers two suggestions to minimize the potential for collisions during address resolution using the Apple Desktop Bus (ADB) Manager:

1. Use a somewhat random stop-to-start time in the range 140 µs ≤ Tlt ≤ 260 µs. Don't be too generous here, however. Tlt is one of the more sensitive ADB timing parameters and it's best to be near the middle of the window—try not to be too close to the endpoints or you may notice erratic behavior. Of course, there is a finite amount of time in the Tlt window and there are a finite number of random addresses a device may return, but in most cases devices should be able to resolve themselves properly. If they do not, it should be a rare coincidence which can be resolved by rebooting or simply using a utility program which executes the _ADBReinit trap.

2. Return a random number in the R3 address field. In fact, you're never required to respond with your actual address in response to a Talk R3. You only must keep track of what your current address is and readdress in accordance with Listen R3 commands from the host. By returning a random number in the address field, you minimize the possibility that two devices are transmitting exactly the same data on the bus at exactly the same time. This is the only instance where a well-designed algorithm would fail to detect a collision.

A well-designed algorithm detects collisions by sensing the ADB data line asserted when it is not expected to be. It must monitor for this condition before transmitting and any time the device is transmitting and expects the data line to be negated. One check per bit cell is typically not sufficient. Monitoring for collision should be a key, repetitive part of the transmission process.

ADB power requirements on Macintosh LC
Date Written: 3/19/91
Last reviewed: 4/29/91

What is the maximum power draw on the Apple Desktop Bus (ADB) on the Macintosh LC? The low-power mouse specifications state that the mouse shall consume no more than 10 mA. Why does Apple specify 80 mA for a mouse on the Macintosh LC? What does Apple recommend for maximum power consumption for a third-party ADB device?

—

The power budget for the Macintosh LC's ADB port changed after the information was compiled for its product information sheets. The Macintosh LC can supply the standard 500 mA that all ADB hosts are supposed to supply.

The standard keyboard and mouse are specified to draw no more than 100 mA. Because any of these devices may be attached to a Macintosh LC, it is necessary to specify the machine for significantly more than the 10 mA actually drawn by "low-power" ADB peripherals.

Apple has always recommended that ADB devices requiring 100 mA or more be designed to provide their own 5 V power source. The ADB engineering specification (062-0267), which you get when you license our ADB technology, has both the host and the device power loading information on page 4.

ADB name mapping

Date Written: 12/19/90
Last reviewed: 11/16/91

Where is the translation of ADB (Apple Desktop Bus) type from an unsigned integer (e.g., 1) to the “real” thing (e.g., “mouse”) documented? If so, what is it?

—

The only mappings of device IDs and addresses to names that you can rely on are the ones that appear in *Inside Macintosh.* Most devices appear like one of those devices. However, there isn’t any other ID to name mappings.

Reset and NMI on a Macintosh IIsi or LC
Date Written: 1/31/91
Last reviewed: 2/13/91

The Macintosh IIsi and LC computers don’t come with programmer’s switches. How do I get reset and NMI on these machines?

—

Both machines have an ADB (Apple Desktop Bus) I/O processor which incorporates the reset and NMI functions through the keyboard. The functions are accessed as follows:

- Use MacsBug 6.2 or later to enable the NMI on the Macintosh IIsi or LC. Both machines start up with NMI disabled because nontechnical users might get into trouble with it, so the ADB controller must be programmed to enable it. Once MacsBug is installed, you can activate NMI by holding down the Command key while pressing the Power button.

- Hold down the Command and Control keys while pressing the Power button, with or without MacsBug installed, to reset these Macintosh systems.

Under some circumstances you may have to hold the reset and NMI key combinations down for a little while (but no more than a second) to make sure the ADB processor sees them.

You’ll find the latest MacsBug on the Developer Essentials disc and in AppleLink’s Developer Support folder. The MacsBug reference manual is very helpful. A package containing both MacsBug 6.2 and the manual is available from APDA for $35.00 (#M7034/B).

“How to ADB” Apple Desktop Bus references
Date Written: 11/14/90
Last reviewed: 12/19/90

I understand that doing Apple Desktop Bus (ADB) development for mice (or is it mouses)

can be difficult. Are there many undocumented technical issues which I need to be aware of in order to effectively communicate with the ADB bus?

—

You should have the following bits o' info:

- *Inside Macintosh* Volume V, Chapter 20, "Apple Desktop Bus"
- Macintosh Technical Note "Space Aliens Ate My Mouse (ADB—The Untold Story)"

- Macintosh Technical Note “Absolute Pointing Device Memory Structure”
  Macintosh Technical Note “Key Mapping”
- Apple Desktop Bus specifications package. This is a package of official Apple Specifications covering the ADB protocol, and Apple-manufactured ADB Devices. This is available from Apple’s software licensing department upon execution of an ADB License Agreement. Contact the licensing department at:

Apple Software Licensing
Apple Computer, Inc.,
20525 Mariani Avenue, M/S 38-I
Cupertino, CA, 95014
(408) 974-4667
AppleLink: SW.LICENSE

Additional ADB references:

Inside Macintosh Volume V, The Apple Desktop Bus
Macintosh Family Hardware Reference
Apple IIGS Hardware Reference
Apple IIGS Firmware Reference

Note: In the case of a discrepancy between the ADB specification and a reference manual, the specification supersedes.

Baum, Peter.
“Boarding the Bus,” MacUser, July 1987, p. 142.
“An Overview of Apple Desktop Bus,” Call A.P.P.L.E., June 1987, p. 24.

In addition to the written word, there are some code samples that will probably prove useful:

- Sample Code #SC.017.TbltDrvr - An example pointing device driver for the Macintosh

- ADBLister - A simple program that uses the ADB manager to display information about devices connected to ADB (on the *Developer Series CD).*

ADB problems from timing extremes & undefined transaction sizes
Date Written: 10/22/90
Last reviewed: 2/20/91

Why do my ADB devices fail often after a restart on my Macintosh IIfx, but work fine on other Macintosh models? Why do they simply not work on the Macintosh IIsi and Macintosh LC?

—

With each new system release and each new machine we hear a few more complaints about broken ADB devices. This is true of system software versions 6.0.4 and 6.0.5, and the Macintosh IIci, IIfx, IIsi, and LC.

A little historical perspective is in order. The Apple Desktop Bus specification is pretty sloppy with tolerances as high as ±30% on some signal timings. This has led to many marginal implementations for ADB devices, both within Apple and from third parties. Unfortunately, the first generation of machines supporting ADB could have been even more lenient than the

specification, so that devices which were -35% might have worked just fine. In at least one instance, a minimum timing was not enforced at all! Our more recent products and system software, while remaining within specifications, are relatively stricter about allowable tolerances.

The most common manifestation of an ADB timing problem is during device initialization. Developers must adhere strictly to the minimum and maximum delay times for initialization transactions and they should aim for the middle of the tolerance window, not the edge. Of particular importance recently is the Tlt (stop to start time) parameter from the ADB Specification. Nominally it is 200 µs. The minimum delay is 140 µs. Many devices which do not initialize properly are found to respond more quickly than this minimum delay during the TalkR3 initialization sequence.

The problem with the Macintosh IIsi and LC is one of protocols. The ADB specification requires a transaction to consist of at least two bytes. Some device/driver combinations have been getting away with “single-byte” transactions. These will not necessarily slip by newer implementations.

Macintosh and multiple keyboard input
Date Written: 11/27/91
Last reviewed: 11/27/91

Is it possible to connect more than one keyboard into an ADB chain on a Macintosh II family computer?

—

Connecting multiple keyboards in parallel should work; however, a bug in many existing keyboards prevents them from distinguishing themselves at boot time. The result is that an application has no way of telling which keyboard has sent input. Parallel input from multiple keyboards doesn’t present a problem, but differentiating input from multiple keyboards often won’t work. This situation has improved with the Extended Keyboard II.

If you need to differentiate multiple keyboard input, use the Extended Keyboard II exclusively or mix your keyboards. Alternatively, the Macintosh can always distinguish a standard ADB keyboard from nonalike keyboards such as the Apple IIGS keyboard.

You can’t go wrong by trying, so hook them up and see if your needs are adequately served. If you don’t notice any problems, then great!

_ADBOp call service routines and data areas
Date Written: 1/.1/91
Last reviewed: 1/1/91

I am confused about the service routines and data areas passed in the _ADBOp call. What does it all mean?

—

That’s a good question.

```
FUNCTION ADBOp (data:Ptr; compRout:ProcPtr; buffer:Ptr;
```

```
commandNum:INTEGER) : oserr;
```

data is a pointer to the “optional data area.” This area is provided for the use of the service routine (if needed).

compRout is a pointer to the completion or service routine to be called when the _ADBOp command has been completed. It has the same meaning as the service routine passed to the _SetADBInfo call.

buffer is a pointer to a Pascal string, which may contain 0 to 8 bytes of information. These are the 2 to 8 bytes that a particular register of an ADB device is capable of sending and receiving.

commandNum is an integer that describes the command to be sent over the bus.

There is some confusion over the way that the completion routines are called from _ADBOp. This calling may be done in one of the following three ways:

1. You do not wish to have a completion routine called, as in a Listen command. Pass a NIL pointer to _ADBOp.
2. You wish to call the routine already in use by the system for that address (as installed by _SetADBInfo). Call _GetADBInfo before calling _ADBOp, and pass the routine pointer returned by _GetADBInfo to _ADBOp.
3. You wish to provide your own completion routine and data area for the _ADBOp call. In this case, simply pass your own pointers to the _ADBOp call.

Remember, there rarely should be a reason to call _ADBOp. Most cases are handled by the system’s polling and service request mechanism. In the cases where it is necessary to call _ADBOp, it should not be done in a polling fashion, but as a mechanism of telling the device something (for example, change modes, or in the case of our extended keyboard, turn on or off an LED).

Apple Desktop Bus (ADB) licensing information

Date Written: 11/22/89
Last reviewed: 11/7/90

To license the Apple Desktop Bus and ADB Device Specifications, contact:

Apple Software Licensing
Apple Computer, Inc.
20525 Mariani Ave., M/S 38-I
Cupertino, CA 95014
1-408-974-4667
AppleLink: SW.LICENSE

Using the LEDs on the Apple Extended Keyboard

Date Written: 5/3/89
Last reviewed: 12/17/90

How can I use the LEDs on the Apple Extended Keyboard?

—

Using the LEDs on the extended keyboard involves the _ADBOp call. Once you determine that you have an extended keyboard (with _CountADBs and _GetIndADB), then register 2 of the extended keyboard has the LED toggles in the low 3 bits.

Therefore, you would do a Talk to register 2 to have the device send you the contents of register 2, manipulate the low 3 bits to set the LEDs, and then pass the modified register 2 back to the device with a Listen to register 2 command.

The Apple Extended Keyboard has an ID of 02 and a device handler ID of 02, while the Apple Standard Keyboard has an ID of 02 and a device handler ID of 01.

Note: At this point it is not clear what Apple has in mind for these LEDs, so you are using them at your own risk.

Use of Apple's ADB Transceiver IC for complex signal timing
Date Written: 3/9/90
Last reviewed: 12/17/90

I want to design the next best-selling graphics pad for Macintosh. I plan to use Apple Desktop Bus (ADB). Is there a chip set available to implement the complex signal timing of the ADB specification?

—

Sorry. Apple's ADB Transceiver IC is proprietary and not currently for sale or licensing. This is the greatest hurdle in designing your own ADB device and requires that you implement your own logic via a dedicated processor or application-specific integrated circuit (ASIC).

Distinguishing between two keyboards on the Macintosh
Date Written: 5/3/89
Last reviewed: 12/17/90

I connected two keyboards to my Macintosh, but I can't distinguish between the two. How can I tell which keyboard a key press came from?

—

Because of a problem in the Apple Desktop Bus (ADB) code, keyboards do not always correctly resolve collisions (multiple devices of the same type on the bus). This problem is in the keyboard, not in the Macintosh. For some applications, simply trying different keyboards until you find two that are assigned different IDs may be a workable solution. An Apple Keyboard and an Apple Extended Keyboard should not collide when connected on the same

bus.

Boot time ADB device address conflict resolution
Date Written: 11/17/89
Last reviewed: 12/17/90

I’m confused about the way Apple Desktop Bus (ADB) device address conflicts are resolved at boot time.

—

The method used by the host to separate and identify the devices at boot time is not well documented, so I’ll try to describe it with some clarity.

The host issues a Talk R3 command to an address. Let’s say there are two devices at that address. Both try to respond to the command, and when they try to put the random number (the address field of register 3) on the bus, one of them should detect a collision. The one that detects the collision backs off and marks itself (internally) as unmovable.

The device that did respond successfully is then told to move to a new address (the highest free address). By definition, moving to a new address means that it now responds only to commands addressed to this new address, and it ignores commands to the original address.

The host then issues another Talk R3 command to the original address. This time the second device responds without detecting a collision. When it successfully completes a Talk R3 response, it marks itself as movable. It then is told to move to a new address.

The host again issues a Talk R3 command to the original address. Since there are no more devices at that address, the bus times out, and the host moves the last device back to the original address. At this point, the host moves up to the next address that has a device and begins the process all over.

Generally, when you have trouble separating devices on the ADB, it is because the collision detection doesn’t work well. In fact, this problem is evident on Apple keyboards. The bug is that the random number returned in R3 isn’t really a random number. Since the microcontrollers on the keyboards are clocked with a crystal, they tend to generate the same “random” number, so when the system attempts to separate them with a Talk R3 command, they never detect the collision.

One possible solution is to use a low-tolerance capacitor on the reset line of the microcontroller, thereby forcing the time from power on to the time reset is negated to be fairly random. In this way, the microcontroller can start a count until it receives the first Talk R3 command, and hopefully it is a different number from the number of another device at the same address on the bus.

If you find your device shows up at all addresses, it may be because it is responding to the move address command when it should be marked as unmovable.

Finally, if the device doesn’t show up at all, it may be because it is unable to respond to the Talk R3 command at boot time (that is, it is not able to initialize itself and start watching the bus in time).

Using ADB to auto power up the Macintosh

Date Written: 11/17/89
Last reviewed: 12/17/89

How can I make my Macintosh II or IIx power up automatically after a power outage?

___

The Macintosh II and IIx power can be turned on via the keyboard through the Apple Desktop Bus port (ADB) since the reset key is wired to pin 2 of the ADB connector. When you press this key, it pulls pin 2 to ground and initiates a power-on sequence. You can emulate this feature with a momentary switch connected to the ADB port. Note that the switch on the back panel of a Macintosh IIcx can be locked in the On position to automatically restart after a power outage.

An idea for a power-on circuit would be to have a momentary (one-shot) relay powered by the same outlet that powers the machine and have the contacts close pin 2 of the ADB connector. (Without having tried this, I am concerned that you may need a delay before the relay fires to give the AC time to stabilize, etc.)