September 1993 Late-Breaking Q&As

This is a new way of getting special Q&As to you before they've been completely reviewed, copy-edited, and added to the Q&A Tech Notes. The content is preliminary; it's been verified by each author, but you might notice changes when you see the final version in a Q&A Tech Note.

M.DV.CD-ROM.Q&As
CD ROM driver and Foreign File Access

What do I have to do to make my CD-ROM driver work with Foreign File Access? Is Foreign File Access licensable from Apple?

—

Foreign File Access is independent of the actual driver. In fact, if you dig out the ISO 9660 article from develop issue 3 (it's on the Developer CD), you'll discover that you can create an ISO 9660 floppy disk and read it using foreign file access. So, as long as your CD driver acts like any other disk driver, you'll find it should work with Foreign File Access without difficulty. It's actually implemented as a component within the file system and issues normal read calls to the disk driver to get the information it needs.

There are two specific requirements. You must support all the driver calls documented in the Macintosh Technical Note "CD-ROM Driver Calls." Also, if your driver is called for a normal read of an audio track (that is, a read request that derives from a driver Prime call), it must return a block of zero bytes and noErr status; it shouldn't return an error. This can happen if the user inserts a normal audio CD in the reader.

If you want to ship Foreign File Access with your product, you should contact Apple Software Licensing (AppleLink SW.LICENSE) for details.

M.DV.SCSIMgr.Q&As
Don't reset bus on SCSI error

My SCSI driver sometimes hangs and I have to reset the bus. Am I making a common mistake?

—

Don't reset the bus. Once you have selected a device, it's absolutely essential that you always exit a SCSI Manager sequence by calling SCSIComplete, even if some other SCSI operation (command, read, or write) detected an error. In other words, the logic of your SCSI driver should be essentially as follows:

```
finalStatus = SCSIGet();
if (finalStatus == noErr)
    finalStatus = SCSISelect();
```

```
if (finalStatus == noErr) {
    /* From now on, you must exit through SCSIComplete */.
    phaseStatus = SCSICmd();
    if (phaseStatus == noErr) {
        phaseStatus = SCSIRead(); /* or SCSIWrite */
    }
    finalStatus = phaseStatus;
    completionStatus = SCSIComplete(&scsiStatus, &scsiMessage, ...);
    if (finalStatus == noErr)
        finalStatus = completionStatus;
 }
```

Note that finalStatus relates only to the state of the SCSI Manager; it doesn't relate to any device-detected errors. For those, you'll have to examine the scsiStatus byte that was returned by SCSIComplete, in a way similar to this:

```
if (finalStatus == noErr && scsiStatus == scsiCheckCondition)
    finalStatus = statusErr; /* Contrived error code */
if (finalStatus == noErr && scsiStatus != scsiGood)
    finalStatus = ioErr; /* Contrived error code */
```

When your higher-level code receives an error status, it will be either one of the SCSI Manager status codes, statusErr or ioErr. If it receives statusErr, the higher-level function should issue a Request Sense SCSI command to get the real error, If it receives ioErr, it should look at the scsiStatus and/or scsiMessage values for more information (for example, the device may be busy).

You may wish to read Macintosh Technical Note "Fear No SCSI" for more details.

M.HW.Sound.Q&As
Macintosh built-in analog sound input port signal

What's the Macintosh built-in analog sound input port signal specification?

—

The Microphone jack is a "1/8 inch stereo mini jack." The tip is the sound input signal, the next ring is the microphone power, and the final ring (closest to the wire) is the ground. The input source should provide a 20 mV amplitude and a 600 Ohm input impedance. The voltage provided by the CPU is 8V DC at up to 1 mA.

M.ME.MemoryMgr.Q&As
Disabling the stack sniffer to set up stack

How do I disable the stack sniffer so I don't get a dsStknHeap error (System Error 28) when I create a stack in my heap?

—

Disabling the stack sniffer for a custom stack is reasonably simple—storing a 0 in the low memory global longint StkLowPt ($110) will turn the sniffer off. *But*, when using your own internal stack be sure not to call any ToolBox routines, because many of them rely on the stack for temporary storage. Playing with the value of A7 will screw things up.

When you're using a custom stack within your heap, you should definitely save the values of StkLowPt and A7 before changing them, and restore them before any Toolbox calls.

M.NW.AppleShare.Q&As
AppleShare 3.0 and program linking

Can program linking be used at the same time as AppleShare 3.0? Since the Users & Groups Control Panel on the server machine can't turn program linking on, users on remote machines can't send Apple events to the server.

—

AppleShare and program linking work fine together. What you'll need to do is use the AppleShare Admin program to manipulate user privileges rather than the Users & Groups control panel (since AppleShare does "take over" user privileges here).

M.PR.PrtDvr.Q&As
LaserWriter 8.0 requires Chooser to change printers

We've been manually writing 'PAPA', 'STR ' and 'alis' resources in the System file and in the LaserWriter driver to change printers without using the Chooser. This method sometimes causes errors with LaserWriter 8.0. What do we need to do?

—

LaserWriter 8.0 needs to know more about the printer than its AppleTalk name—it also has to have a PPD file for that printer, parsed and ready to be used. Since there's so little memory available in applications like the Finder during printing, the parsing is done at Chooser time, not at print time.

Apple has *always* said "We can't guarantee that you can change printers behind the Chooser's back," and with LaserWriter 8.0 this is true. If the driver has parsed a PPD file and has it ready, things should work OK, but you'll have to have manually chosen that printer before and set everything up. If you set up a LaserWriter IINTX printer with the correct PPD file, then chose a LaserWriter IIf, and then another printer driver, you could probably programmatically switch back to the LaserWriter IINTX, but the driver will use the LaserWriter IIf PPD file with it, which might or might not cause PostScript errors. Designing the driver to be switchable by other applications simply wasn't a priority of the Adobe/Apple development team.

As long as you try to switch to a printer that uses the same PPD file that the driver last parsed (meaning the PPD associated with the last printer selected in the Chooser), there shouldn’t be any more problems than there were before.

M.PR.PrtDvr.Q&As
Downloading fonts for EPS illustrations
69830 MD LS 4/20/93

With the LaserWriter printer driver, assuming the user has selected the “Unlimited Downloadable Fonts” option, how does an application arrange to download the fonts necessary to draw an Encapsulated PostScript illustration? (The unlimited downloadable option appears to allow only a single font to be downloaded at a time.) I know how to get the list of fonts from the EPS file, and how to match PostScript names from the FOND style mapping tables, but I need to download a set of fonts, more-or-less simultaneously, so that they’ll be available to the EPS drawing.

—

If you want to send EPS illustrations to the printer, you’re responsible for making sure the fonts are available in the printer. The system software provides no support for helping you do this, which is one of the side effects of being QuickDraw-based and not PostScript-based.

The new LaserWriter driver version 8 has a new PrGeneral API that will let you pass it a font name or ID and get back a PostScript stream necessary to make that font available on the printer, including the TrueType scaler if necessary. You can’t require that feature in order to print EPS files, however. See “Print Hints” in this month’s issue for more details.

To download a font, you normally draw one space character in that font and then use the current font (or other PostScript machinery) to use the font the driver made available. Other than this trick, there’s no way to make the driver download a font for you, much less do the proper things to TrueType or bitmap fonts. You can try to do it all yourself, but that’s unbelievably complicated.

It’s probably easier to just tell users that unlimited downloadable fonts may cause EPS files to print text in Courier instead of the appropriate font. We’ve seen this done before; it’s not pretty, but it works. You can try to do more yourself with LaserWriter 8, but you’d still be responsible for managing printer memory and other such pleasantries even in that situation.

M.PR.QDGXPrt.Q&As
QDGX printer driver and unsupported comm connection type

I’m trying to use the new QuickDraw GX printing architecture with an unsupported communications connection type. How do I make the 'comm' resource to support a device that doesn’t have a PAP, SCSI, or serial interface? What messages do I have to override? Is there any documentation other than the beta Inside Macintosh volumes on this?

—

You need to create a custom 'comm' resource that reflects your communication mechanism.

You'll need to override the following messages: OpenConnection, CloseConnection, BufferData, WriteData...at a minimum. You're using buffering (that is, they have the ability to do asyncIO), you'd also need to override DumpBuffer and FreeBuffer.

You must specify customIO in your 'iobm' resource.

You'll also need to override the ChooserMessage and/or DefaultDesktopPrinter to store the 'comm' resource into the DTP.

Finally, by default, the contents of the cell (that you fill into the list by overriding the ChooserMessage) are copied straight into the 'comm' resource, and are also assumed to be the name of the desktop printer.

You have all of the documentation currently available for QuickDraw GX. We're in the process of updating the printing and extensions book with in the QuickDraw GX documentation. We're also in the process of creating a sample QuickDraw GX printer driver which uses a nonstandard communication protocol.

M.PS.NotifMgr.Q&As
Notification Manager and changing cursor

When using the Notification Manager with the Show Alert option, the cursor gets changed to an arrow and doesn't get changed back afterwards. Is there some way to recover the original cursor, or to detect when the Notification Manager may have changed the cursor?

—

There's no good way to find out when the Notification Manager has changed your cursor (other than maybe patching _SetCursor and checking who's doing what, but who wants to do that?) The best thing to do to make sure that your cursor is always set properly is to call SetCursor every time through your main event loop or when you receive an activate event.

Calling it every time through the event loop isn't as bad as it may sound, because that's when your cursor tracking region should be updated for the next call to WaitNextEvent, and you may have to change the cursor anyway depending on its current position.

If you're not tracking the mouse's position, just make sure the cursor is set properly when your application receives an activate event, since one will be caught once you've responded to the notification.

M.PT.MiscTool.Q&As
Code for determining whether At Ease is running

My code quits all other applications, including the Finder, while our application is running. However, if the user is running At Ease before launching our application, the Finder is launched

when our application quits. How do I determine if At Ease is running, and how can I relaunch At Ease when my application is ready to terminate?

—

You're probably shutting down At Ease when you're trying to shut down other things. This is why At Ease isn't coming back when you quit. You should test to see if At Ease is running. If it isn't running, your existing method is correct. If At Ease is running, don't bother trying to shut down any other processes since they're all dormant anyway.

To determine whether At Ease is running, call Gestalt and ask for the At Ease information handle. The selector and the structure returned are:

```
#define kAtEaseGestalt    'kids'
typedef struct {

       short    giVersion;            /* structure version */
       short    giIsActive;            /* true if at ease is currently running */
       short    giAutoCreateAlias;    /* if true then auto create alias */
       short    giRequestFloppy;    /* if true then request floppy on new saves */
       short    giStacksAreApps;    /* if true then HyperCard stacks are shown with
applications */
       FSSpec    giItemsLocation;    /* location of the At Ease Items folder */
}    GestaltRec, *GestaltRecPtr, **GestaltRecHand;

// The following code snippet shows how to detect if At Ease is installed and running

   GestaltRecHand    aeGestaltInfoHdl;

   if (Gestalt(kAtEaseGestalt, &(long)aeGestaltInfoHdl) == noErr ) {
       if ( (**aeGestaltInfoHdl).giIsActive ) {
           AtEaseIsRunning();
       }
```

M.PT.MPWC.Q&As
Importing code with "/" as path separators

I have a problem importing code to MPW C from the UNIX and DOS environment. Some of our #include's have file names with relative paths in them such as this: #include "core/core.h". The MPW C compiler accepts the "/" as a valid character in a file name, and not as a path separator. The compiler doesn't appear to have any switch to allow the "/" character to be interpreted as the ":" character. How do I import code with embedded "/" path separators without having to change all my headers?

—

The MPW C compiler doesn't have an option to switch path separators. As you mentioned, on the Macintosh a "/" is a perfectly legal character in a folder or file name. The following is probably the best way to work around your problem:

Use StreamEdit to process all #include directives and pipe the output to C:

```
 StreamEdit -s IncludeScript "{TheSource}" | c -o "{TheSource}".o
```

with the IncludeScript file containing:

```
 /ç•[     ]*∂#include/ replace -c ∞ /∂// "∂:"
```

This will compile your C sources without modifying the original file or creating temporary source files.

In a Makefile you could change the default build rule to compile C sources to include this StreamEdit step. The time overhead for using this additional step is negligible when compared to other alternatives (such as keeping two versions of your sources—one for UNIX/DOS and one for the Macintosh).

More information on the use of StreamEdit can be found in the MPW Command Reference, in the StreamEdit section.

M.PT.MPWC.Q&As
Controlling the alignment settings in MPW C

MPW appears to pad structures so that all members begin on even boundaries. Is there a way to disable this padding feature so we can read data from disk files into structures without misalignment? We recognize that accessing shorts or longs on odd addresses causes errors on 68000 machines but our application is meant for 68020 and later CPUs.

——

There's no way to disable this padding through the use of a compiler option. Pretty much the only way to get around this is to read the data off a disk and unpack the data yourself using macros to properly offset into the data structure. A quick example of this follows. Suppose you have a structure as follows:

```
typedef struct {
  char   oneField;
  short  twoField;
}whatever;
```

so the data saved to disk is 3 bytes per structure (which would be padded up to four bytes). What you can do is read each structure from disk into an array and set up some macros. So if the array containing raw data from disk is declared as

```
char buf[3];
```

Set up some macros as follows.

```
#define firstField(str) ((char)(str[0]))
#define secondField(str) (*(short *)&str[1])
```

and you can use firstField(buf) to refer to the firstField of buf and secondField(buf) to refer to the second field of buf. This provides a reasonably quick and easy way to access the data needed.

M.PT.SharedLibraryMgr.Q&As
InitLibraryManager default exception handling

We're having problems with InitLibraryManager. When a debugger is installed, the problem manifests as a DebugStr call from InitLibraryManager with the message "An exception was thrown and the application didn't catch it." Without a debugger, it becomes a system error (of course). Could you shed some light on this?

—

When the InitLibraryManager call is made, it installs a default exception handler which will catch any exception that is thrown and not caught by the client. When you end up in this default exception handler, your code will appear to be executing in the InitLibraryManager routine. The code will have executed a jump instruction that was set up with a setjmp rather than actually calling InitLibraryManager. When this happens, you'll see the DebugStr you mentioned and then an ExitToShell is done.

If you want to avoid this, you'll need to set up an exception handler (using TRY/ENDTRY) around any code that may throw an exception. This includes code that creates an object in a library that isn't already loaded, or code which calls a function in a function set that's in an unloaded library. Use LoadClass, LoadFunctionSet, LoadLibraries, and/or NewObject if you want to avoid having to setup an exception handler.

If you want to see where the exception is being thrown, (1) install the debug version of the Shared Library Manager, (2) load the Inspector application, so you'll know the ASLM is loaded, and (3) break into MacsBug, switch to the system heap, and set a break on "Fail." When the exception is generated, check to see where the exception came from by doing a stack crawl. Chances are it will be in some ASLM code that trying to load the library. You probably just ran out of memory so we couldn't load the library.

M.PT.SharedLibraryMgr.Q&As
Exporting a class that isn't a subclass of TDynamic

Using the Shared Library Manager, can I share a class that isn't derived from TDynamic? I seem to recall a statement that a TDynamic class and subclassed object has a v-table pointer at the front of the object.

—

With SLM 1.1, you can export a class that isn't a subclass of TDynamic; just add the "Class" declaration. The only provisos are: No constructor may be "inline," and the destructor may not be inline. SLM exports all the nonvirtual member functions automatically. You can use the -dontExport flag to keep certain functions from being exported. Any static methods of a class still should be exported in a separate Function Set.

If you export a class that doesn't have the vtable at the front, make sure you don't try calling CastObject

on it. Otherwise, everything should work.

M.PT.SharedLibraryMgr.Q&As
Generating a link map with BuildSharedLibrary script

How can I generate a link map with the BuildSharedLibrary script?

—

Include the following option with the other parameters to the BuildSharedLibrary script:

```
-link -map mapFileName
```

M.QD.BasicQD.Q&As
Calling StdText directly

Is it OK to call StdText directly, instead of calling one of the normal text-drawing traps? Sometimes I want to use the extra scaling options StdText has.

—

Calling StdText directly usually interferes with printing, where the printer driver captures the QuickDraw bottlenecks to redirect their output. If you call the trap directly, the currently-chosen printer driver won't get a chance to see the text.

If you want this functionality, extract the address of the text bottleneck procedure from the current grafPort or CGrafPort (as described in Inside Macintosh Volume I, or Volume V for Color QuickDraw) and call that routine directly as if it were StdText.

The bottleneck routine takes the same parameters and usually goes right through to the trap; it's the cases where it *doesn't* that require you to call the routine in the current port instead.

M.QD.ColorQD.Q&As
Saving and restoring current port's highlight color

What's the best way to save and restore the current highlight color?

—

Get the current hilite color from the GrafVars handle to save it. Don't worry about accessing the handle directly, as Inside Macintosh warns against. It's the only way to do this. Then just use HiliteColor to set the new hilite color, and HiliteColor again to restore the saved one.

See Inside Macintosh Volume V, page 67, for a description of the GrafVars structure.

M.QD.QDGX.Q&As
QuickDraw GX font that calculates check digits

I want to create a QuickDraw GX font which will calculate check digits as the user types the numbers in. For example, if you type “123458723” the check digit would be 5; if you type “098732734” the check digit would be 7; and so on. The formula is Check digit = sum of 9 numbers MOD 10.

Is there a way to do this using the glyph modification properties in QuickDraw GX? I know it can be done by creating an entry in the 'mort' table of each for all unique possibilities, but there are 9 to the power of 9 possibilities which comes to 387,420,489. Can I enter a formula instead of creating tables? If so, how?

—

It’s a fascinating idea, but it’s not something the 'mort' tables can handle.

The tables are basically a state machine; when they detect a particular state they take some action based on entries in the table. There’s no way to add a formula or code in any language to these tables.

All the mort tables can do is take a series of 2 or 3 glyphs and replace them with a different glyph if a given feature is enabled by Line Layout. For example, you can change “ffi” into the ligature (one glyph) for the same three letters.

If you wanted to do this for check digits, you’d have to have an entry in the mort table for each glyph combination you wanted substituted. For 300000000, you’d have to have one entry that substituted 3000000003, and the substitution has to be one glyph.

Since order is significant, that means you’d have to have one billion entries in the mort table and one billion glyphs representing all the entries with their check digits added.

This is impractical because:

1. It would make the font very large
2. Searching the tables for each character typed would take a *very* long time, so drawing layouts would slow to a crawl.
3. It would take forever to create the font.

This is impossible because:

1. Offsets in the font have to fit within a long integer, which is 4.2 billion, and it’s unlikely that all offsets could fit within 4.2 billion bytes for 1 billion entries
2. The size of the font would exceed Resource Manager limits
3. The mort tables don’t support mapping nine characters to a single glyph, just a small number (2-3, I believe).

It’s a really neat idea, but it won’t work.

M.QD.QDGX.Q&As
QuickDraw GX: Window resizing

When I resize a window in my QuickDraw GX application, nothing draws outside the area of the previous window contents. If the user repositions the window by dragging on its title bar, and then does anything which causes the window contents to be redrawn, the contents appear correctly. I've been getting around this by calling MoveWindow after SizeWindow, but is this really hiding a bug in my code?

—

This is actually the correct behavior. When you attach a parent viewPort to a window by calling:

```
parentViewPortOfTheWindow = NewWindowViewPort(gWindow)
```

QuickDraw GX assumes the clip shape of the viewPort won't change (that is, the window's size will never change). The parent viewPort attached to the window isn't to be changed by the application. As you've determined, we have a few patches in the Window Manager to update information between the QuickDraw and QuickDraw GX worlds to keep the viewPort(s) attached to the window in sync with the window as the user drags it around the screen, but we assume that the user never resizes the parent viewPort.

QuickDraw GX realizes that the viewPort clips aren't correct after the user has repositioned the window, and then updates the viewPort's clip shape and cache information.

There's a simple solution to the problem you're encountering. You need to attach a child viewPort to the parent viewPort attached to the window. With this setup, you can do anything you wish to the child viewPort. However, it's the application's job to update the child viewPort's clipping shape after the user resizes or zooms the window. QuickDraw GX also concatenates the information between the parent and child viewPorts to keep every thing in sync at drawing time.

A sample called "GX Scrolling," accompanying the article "Getting Started with QuickDraw GX" in issue 15 of develop, handles all the window and viewPort operations described above.

M.QT.ImageCompr.Q&As
Using QuickTime dither tables in a codec

How can I use QuickTime fast dither tables provided by the Image Compression Manager to write a codec? I haven't been able to find any documentation on how to access and use them. Are these tables available?

—

For QuickTime 1.0 you could use the MakeDitherTable and DisposeDitherTable calls in

ImageCompression.h. The calls were taken out for QuickTime 1.5 because the format is likely to change and your code would break in the future. The current dither table format isn't available for that reason, though the documentation on the QuickTime 1.0 CD describes the calls, if that helps.

You can use QuickTime to perform the dithering. If you do use QuickTime, you could draw the image in an off-screen GWorld, using the DrawTrimmedPicture with the dither flag set, and then compress it with your codec.

(Note: “DrawTrimmedPicture” supersedes “DrawTrimmedPictureFile” in Q&A Tech Note version)

M.TB.DesktopMgr.Q&As
Which of multiple application copies gets launched

If I double-click a document on a disk with multiple copies of an application, which one gets launched?

—

Under System 6, the Finder adds information about the last application copied to a disk to the disk’s Desktop file, and that application is used for opening documents with the appropriate creator type. The clue to System 7’s strategy in deciding which application to launch is in the description of the desktop database call PBDTGetAPPL on page 9-53 of Inside Macintosh Volume VI. The desktop database prefers the application with the latest creation date as well as the application most recently copied to a hard drive, although it knows about all copies of the application with BNDLs on the drive. Of course, the Finder works in mysterious ways, and rebuilding the desktop is sometimes necessary to get the proper version of a program to be used (and even then, having multiple copies of the application on the drive can cause unexpected things to happen.) In general, just make sure that the new version has newer creation and modification dates than older versions, and things should work out fine.

If you want to explore the contents of the desktop database, pasted below is a small function which will retrieve information about the applications available to documents on a disk. Call GetAppl with an index starting at 0 and increasing until an error is returned to see the applications known to the desktop database for the specified disk.

```
FUNCTION GetAppl (vRefNum: Integer; index: Integer; signature: OSType; VAR appName: Str255;
VAR parID: LongInt; VAR createDate: LongInt): OSErr;
 VAR
  retCode: OSErr;
  myDTPB: DTPBRec;
  tempName: Str63;
BEGIN
 tempName := '';
 myDTPB.ioNamePtr := @tempName;
 myDTPB.ioVRefNum := vRefNum;
 retCode := PBDTGetPath(@myDTPB);
 IF retCode = noErr THEN
  BEGIN
   tempName := '';
   myDTPB.ioNamePtr := @tempName;
   myDTPB.ioIndex := index;
   myDTPB.ioFileCreator := signature;
   retCode := PBDTGetApplSync(@myDTPB);
   IF retCode = noErr THEN
    BEGIN
     appName := tempName;
     createDate := myDTPB.ioTagInfo;
     parID := myDTPB.ioAPPLParID;
    END
```

```
  END;
 GetAppl := retCode;
END;
```

M.TX.TextUtil.Q&As
System 7.1 workaround for slow IUCompString

Our application uses IUCompString extensively in a sorting routine, and we estimate that this call is about 30 times slower in System 7.1 than it was in System 7.0.1! What's going on?

—

System 7's Script Manager would cache script resources to increase performance. That's why IUCompString was fast. System 7 also allowed for scripts to be installed and removed via the Finder (by dragging the script icon to or from the System file) without rebooting, and this corrupted the cache. System 7.1 solved this problem by removing the cache (making IUCompString much slower).

To work around this caching problem, new routines were added that allow the application to cache the script resources and pass them explicitly when calling Script Manager routines. The Finder will not allow script resources to be removed when applications are open, so this prevents their private caches from being corrupted. Some of these routines may not be available on System 6, so you may have to check to see whether the calls are available and then call the best routine.

Inside Macintosh: Text describes how to use CompareString and CompareText; both take an optional 'itl2' handle. Inside Macintosh Volume VI, page 14-72, describes IUCompPString, which is the older routine. In general, all the Script Manager routines will work faster if you pass the script resources in explicitly.

These calls may actually improve performance above System 7.0.1 levels.

A.AppleIIGS.Q&As
Apple IIgs System 6.0 ModalWindow bug and workaround

I've noticed an Apple IIgs System 6.0 problem: I call SetDAFont passing the font handle I got from the GetSysFont call; then I call SFGetFile2. If I click outside the dialog rect, ModalWindow should call SysBeep2 but instead it crashes. The problem occurs because DoModalWindow crashes when I call SetDAFont with a handle to "FastFont" Shaston 8. It assumes a byte on a direct page will always be zero when with FastFont it's not.

—

You've found a bug in system software, which is fixed for the next system software release. Meanwhile, engineering not only has an instant workaround, they claim the bug you see will never happen on a ROM 3 machine.

While the bug prevents you from passing the results of GetSysFont (an $80 in the high byte tells QuickDraw this is the FastFont handle, so it can't be stripped), you don't have to call GetSysFont at all. Passing $00000000 to SetDAFont tells the Dialog Manager to use the font that was the system font when the dialog was created, so it will never call SetFont to replace the dialog port's font with a new one.

If you want to reinstall the font that was the system font when the dialog was created, SetDAFont(NIL) should always work. If, however, you want a font *other* than that one, you have to get it into the Dialog's grafPort yourself. You can call InstallFont with the port set, or if you already have a font handle just call SetFont. Once you've done this, call SetDAFont(NIL); this places NIL on the Dialog Manager's direct page, so the Dialog Manager never calls SetFont and never overwrites the font handle you've installed manually.