New Technical Notes
Macintosh

Developer Support

Control Manager Q&As

Toolbox

Revised by: Developer Support Center October 1992
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.

Use calcCntlRgn and calcThumbRgn instead of calcCRgn high bit
Date Written: 6/3/91
Last reviewed: 8/1/92

We tried using our Macintosh System 6.0 scroll bar 'CDEF's under System 7.0 Golden Master and find that they no longer update properly (although they worked fine in older versions of System 7). What's changed?

___

Nothing much has changed as far as custom 'CDEF's go between System 6 and System 7, except for the way the system gets the indicator (thumb region) for a control. As you may know, the high bit of the parameter passed to calcCRgn used to be set to indicate that only the thumb region was needed. Because this is not 32-bit clean, Apple has defined two new messages that work just like the old calcCRgn: calcCntlRgn (message #10) wants you to fill param with the entire control's region structure, and calcThumbRgn (message #11) simply wants the indicator region. It is also important to note that you must return a non-zero result if you actually return this data. You probably are not yet processing these messages, or you are not setting your result to 1 to indicatethat you have created the regions. You might also want to take

You might find DTS’s current sample 'CDEF', in the Snippets folder on AppleLink and the latest Developer CDs, helpful. Even though it is not a scroll bar, you can see the “standard” way to use the color table information.

Debugging a Macintosh 'CDEF' resource

Date Written: 6/4/91
Last reviewed: 8/1/92

Can you recommend documentation describing techniques for debugging 'CDEF's?

___

You debug 'CDEF's in almost the same way that you debug anything else. You should familiarize yourself with the order that messages are received when certain actions take place. One way to do this is to install some breakpoints in the main routine of DTS’s sample 'CDEF', watch the messages go by, and examine the parameters that are passed. This way you should be able to predict exactly what messages you should receive. (By the way, be sure to read the Macintosh Technical Note “The Joy Of Being 32-Bit Clean” on 32-bit cleanliness for information on the new 'CDEF' messages.)

Just as an aside, when tracing through a 'CDEF', use the following to break conditionally at certain points only if the Shift key is held down:

```
Function TestShift:Boolean;
    InLine $1EB8,$017B;
```

Then, whenever you want to break, insert

```
If TestShift then DebugStr('Stopped because you held down the shift key!');
```

Color with Macintosh radio button and check box controls

Date Written: 6/11/91
Last reviewed: 8/1/92

I can control the color of a Macintosh radio button and its accompanying text using a control color table, but I am not able to exercise much control over the color of the background rectangle. Let’s say, for example, that I create a window whose content area is gray (using SetWinColor). Within the window, I draw a black box, and within the black box, I draw a pair of radiobuttons whose text color is white. I would like the background rectangles of the radiobuttons to be black so that they will match the color of the box. Unfortunately, using Draw1Control, the Control Manager seems to draw the background rectangle using the content color of the window (in this case, gray). Is there any easy way to get around this? (Calls to RGBForeColor and RGBBackColor seem to have no effect during Draw1Control.)

___

The explanation for the problems you are having is very simple. Standard check boxes and radio buttons use the window content color for their background. This is by design since the text of the button is not bounded by anything. Here are two resolutions to your problem: You can simply set your window background to the background you want the button to have, or you can write your own 'CDEF' to handle buttons and have it use the control’s background color instead of the window’s content color. You’ll find sample code that does this in the Snippets folder on AppleLink and the latest Developer CDs.

Changing a Macintosh control’s variant

Date Written: 10/3/91
Last reviewed: 8/1/92

What is the proper method to set a control’s variant field, if there’s no SetCVariant trap?

___

There is no way to set the Variant field of a control from the toolbox. Standard controls do not support the dynamic modification of the variant field; they are not tested for this and an undefined error could result. As for custom controls, if you need to change a variant midway through the control’s use, it’s better not to use the variant field at all, but instead to store your own data structure in the ctrlData field to track what type of control you are drawing. The reason for this is that the variant field is not always in the same place. The only time it is in a well-defined place is when you are operating in 24-bit mode (then it is stored in the high byte of the controls defproc handle, as documented in *Inside Macintosh* Volume I). When you are in 32-bit mode, it is stored in the AuxCtlRec in a private location.(high byte of the Reserved field right now, but that can change in an instant...)