New Technical Notes

Macintosh

Developer Support

Interrupt Task Overview Q&As

Processes **M.PS.ProcessOV.Q&As**

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.

New Q&As and Q&As revised this month are marked with a bar in the side margin.

Macintosh VBL documentation
Written: 10/29/90
Last reviewed: 8/1/92

Comments in the Technical Note "MultiFinder Miscellanea" examples mention that A0 points to the VBL record when a VBL task is called, but we can't find this documented in Inside Macintosh. Is this really true?

—

Although we'd like our documentation to be perfect, sometimes it isn't. We are always striving to improve the documentation. One method of making corrections to published volumes already in the hands of developers is to issue technical notes. So, although we will strive to have Inside Macintosh document that A0 points to the VBL record entry on when called, until it is published in a new volume we provided this Technical Note as additional update documentation. You can depend on A0's contents as it is documented in "MultiFinder Miscellanea."

What can I do and not do from a Macintosh interrupt routine?
Written: 5/14/90
Last reviewed: 8/1/92

What can I do and not do from a Macintosh interrupt routine?

___

The most popular examples of interrupt code are I/O completion routines, VBLs, Time Manager tasks, and deferred tasks. Contrary to popular belief, the Deferred Task Manager will run your task at interrupt level. All code that runs at interrupt level must follow very strict rules. The most significant rule is that the code cannot allocate, move, or purge memory, nor can it call a Toolbox routine that would. This eliminates nearly, if not all, QuickDraw operations. For a more complete list of these toolbox routines, refer to the Inside Macintosh X-Ref.

Additionally, interrupt code must avoid accessing a low-memory global or calling a trap that would access one. While MultiFinder is running, application's low-memory globals are being swapped in and out. Because of this, the interrupt code cannot rely on which application's globals are currently available. Even if CurApName is correct, the interrupt routine may be called while MultiFinder is in the process of swapping the application's globals. This restriction is difficult to deal with because it isn't documented as to which low-memory globals are swapped by MultiFinder, nor which globals are accessed by traps.

A typical example of this problem is interrupt routines that attempt to restore A5 by examining CurrentA5. This low-memory global is only valid while the current application is running at non-interrupt time. Thus, the MPW routine SetCurrentA5 (or the obsolete SetupA5) cannot be used at interrupt level. It is necessary to place the application's A5 somewhere it can find while in interrupt routine. This is documented in DTS's Macintosh Technical Notes "MultiFinder Miscellanea" and "Setting and Restoring A5." In fact, the exact code you need is in "MultiFinder Miscellanea."

It is best to avoid interrupt code if at all possible. Also, there are limitations while System 7.0 is running under virtual memory. Move the functionality of the interrupt code into the application. For example, if you do require a VBL, limit the code to an absolute minimum. Set a global flag for the application to check in its event loop.