New Technical Notes
Macintosh

®

Developer Support

OS Queue Utility Q&As
Operating System Utilities

Revised by: Developer Support Center May 1993
Written by: Developer Support Center May 1993

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 this month:
Allocating parameter blocks for slot interrupt handler

Allocating parameter blocks for slot interrupt handler
Date Written: 12/11/92
Last reviewed: 3/1/93

I need to issue multiple concurrent asynchronous file I/O calls from within my interrupt handler. How should I allocate the parameter blocks for these calls? I can't use the stack, as it isn't persistent beyond my interrupt handler routine.

___

At interrupt time, you can't allocate (or deallocate) parameter blocks with Memory Manager calls. Since you're using the parameter blocks to make asynchronous calls, you can't allocate the parameter block on the stack. So, that means you'll have to preallocate the parameter blocks and put them somewhere where you can grab them at interrupt time.

We suggest you allocate a block of parameter blocks large enough to handle a reasonable number of interrupts and put them in an OS queue owned by your code (see *Inside Macintosh* Volume II, chapter 13, for a description of OS queues, and the Enqueue and

Dequeue functions). Whenever your interrupt code needs a parameter blocks, it can attempt to grab the parameter block at the head of the queue with the Dequeue function. Whenever an asynchronous call’s completion routine no longer needs a parameter block, it can put the parameter block back into with the Enqueue procedure. Since Enqueue and Dequeue disable interrupts briefly while searching and adding or removing a queue element, they’re safe from race conditions.

To handle the situation where the number of parameter blocks originally put in the OS queue isn’t enough to handle a burst of activity, you can use a driver (this assumes that you can’t just “do nothing” when a parameter block can’t be found). When your interrupt handler can’t get a parameter block from the OS queue (because it’s empty), it can drop the request in another OS queue and continue execution. The driver can use the time it gets during an accRun control call to check that queue for deferred requests. If it finds a deferred request, it can allocate another block of parameter blocks, add them to the parameter block OS queue, and handle the deferred request. This shouldn’t happen often (if at all) if you preallocate enough parameter blocks to handle bursts of activity in the first place. To prevent this situation from happening in the future, you should probably store the number of parameter blocks used so the next time the system is restarted, you can start with a larger default.