Apple IIGS
#57: The Memory Manager and Interrupts

Revised by: Dave "nocturnal" Lyons December 1991
Written by: Dave Lyons July 1989

This Technical Note discusses how you can use the Memory Manager from interrupt routines and documents a flag byte that debugging utilities can use to temporarily prevent the Memory Manager from moving or purging memory.
Changes since July 1989: Expanded and retitled Note to discuss safe use of the Memory Manager at interrupt time.

The Memory Manager does not disable interrupts while it's busy. Instead, it increments the system `BUSY` flag when it's in the middle of something important.

Can interrupt routines call the Memory Manager?

If you write code that executes at interrupt time, you must check the `BUSY` flag (the byte at $E100FF) before making any Memory Manager calls. If the `BUSY` flag is zero, it's safe to call the Memory Manager. If the `BUSY` flag is nonzero, the Memory Manager may be in the middle of a call, so it is not safe to call it.

What routines must check the BUSY flag?

Classic desk accessory main routines and shutdown routines do not need to check the `BUSY` flag. If the Event Manager is active, the CDA gets control during `GetNextEvent`, not at interrupt time. If the Event Manager is not active, the CDA gets control only when the `BUSY` flag reaches zero.

GS/OS signal handlers do not need to check the BUSY flag, because the system dispatches signals only when the BUSY flag is zero.

Run Queue tasks do not need to check the `BUSY` flag before calling the Memory Manager. The system dispatches Run Queue tasks at `SystemTask` time—the `BUSY` flag may not be zero, but no Memory Manager call will be in progress.

Heartbeat interrupt tasks and other interrupt handlers **do** need to check the `BUSY` flag before calling the Memory Manager.

Interrupt-time use of moveable memory blocks

If an interrupt-time routine needs access to an unlocked, non-fixed memory block, you must check the `BUSY` flag. It is not sufficient to lock the block, use it, and then unlock it (even if you twiddle the handle's access word directly). If the `BUSY` flag is non-zero, the Memory Manager could be in the middle of compacting memory, which means your block could be "in transit" from one address to another (some bytes copied, some not).

To use already-allocated memory at interrupt time, either keep the block locked or fixed, or check that the `BUSY` flag is zero before using the memory at interrupt time.

What if BUSY is nonzero?

If the `BUSY` flag is nonzero, you may want to (depending on your application) exit the interrupt routine and hope the `BUSY` flag is zero the next time, or call `SchAddTask` in the Scheduler to make the system call your routine when the `BUSY` flag next returns to zero. Keep in mind, though, that only four scheduled tasks can be pending at a time.

Interrupt-time flag byte

If the byte at location $E100CB is non-zero, the Memory Manager will not move any memory blocks, and it will not purge any blocks while trying to allocate memory (`PurgeHandle` and `PurgeAll` will still purge blocks).

Debugging utilities may temporarily increment this byte to allocate memory in situations when it is not safe for existing memory blocks to be moved or purged.

This flag byte is for use **only** by debugging aids and System Software. It would be mind-numbingly stupid for an application to use this flag instead of using `HLock` and `HUnlock`, since the advantages of a Memory Manager architecture with relocatable blocks would be lost.

It is not useful to check the value of the $E100CB flag. It is always set during interrupt handling whether any non-reentrant system component is busy or not.