Using Virtual Memory

The routines described in this section allow drivers and applications with critical timing needs to intervene in the otherwise automatic workings of the virtual memory paging mechanism.

> **Note:** The vast majority of applications do not need to use these routines. They are used primarily by drivers, debuggers, and other interrupt-servicing code.

If necessary, your software can request that a range of memory be held in physical memory. *Holding* means that the specified memory range cannot be paged out to disk, although it might be moved around within physical RAM. As a result, no page faults can result from reading or writing memory addresses of pages that are held in memory.

Similarly, a page or range of pages can be locked into physical memory. *Locking* means that the specified memory cannot be paged out to disk and that the memory cannot change its real (physical) RAM location. (Locking, therefore, is a superset of holding.) You can also request that a range of pages be locked into contiguous physical memory, although contiguity is not guaranteed. Locking pages into contiguous physical memory is used primarily when external hardware transfers data directly into physical RAM. This might be useful for keeping a contiguous range of memory stationary during operations of an external CPU (on a NuBus card, for example) that cannot support a DMA action.

Most applications do not need to hold or lock pages into physical RAM because the operation of virtual memory is usually fast enough that your application is not affected by any delay that might result from paging. Software that does need to hold or lock pages, such as drivers or sound and animation applications with critical timing requirements, normally needs only to hold memory, not lock it. Here are some general rules regarding when to hold or lock memory:

- Avoid executing tasks that could cause page faults at interrupt time. The less work done at interrupt time, the better things are for all applications running.

- You cannot hold or lock memory (or call any other **Memory Manager** routine) at interrupt time.

- Do not lock or hold everything in RAM. Sometimes either holding or locking pages in RAM is essential, but if you are ever in doubt, then probably neither one is needed.

- Whatever is held or locked must be explicitly unheld or unlocked by the application. If for some reason an area of RAM gets held and locked, or held twice, then it must be unheld and unlocked, or unheld twice.

This last directive is especially important. Your application is responsible for undoing any of the effects it causes by locking or holding ranges of memory. In particular, virtual memory does not automatically unlock pages that have been locked down. If you do not undo these effects in a timely fashion, you are likely to cause poor performance. In the worst case, you could cause the system to run out of physical memory.