Enabling or Disabling the Idle State

You can reset the activity timer to 15 seconds, disable and enable the idle state, and read the current CPU clock speed by using **Power Manager** routines.

Keep in mind that it is almost always better to design your code so that it is not affected by the idle state; then the Macintosh Portable can conserve power whenever possible. Note also that disabling the idle state does not disable the sleep state. To prevent your program from being adversely affected by the sleep state, place a routine in the sleep queue, as described in Placing a Routine in the Sleep Queue.

To reset the activity timer to count down another 15 seconds before the **Power Manager** puts the Macintosh Portable in the idle state, use the **IdleUpdate** function. The **IdleUpdate** function takes no parameters; it returns the value in the Ticks global variable at the time the function was called.

If you want to disable the idle state-that is, prevent the Macintosh Portable from entering the idle state-for more than 15 seconds, use the **DisableIdle** procedure. If your application cannot tolerate the idle state at all, you can call the **DisableIdle** procedure when your application starts up and then call the **EnableIdle** procedure when your application terminates.

The **EnableIdle** procedure cancels the last call to the **DisableIdle** procedure. Note that canceling the last call to the **DisableIdle** procedure is not the same thing as enabling the idle state. For example, if the user has used the Portable control panel to disable the idle state, then a call to the **EnableIdle** procedure does not enable the idle state. Similarly, if your routine called the DisableIdle procedure more than once or if another routine has called the DisableIdle procedure, then a call to the **EnableIdle** procedure cancels only the last call to the **DisableIdle** procedure; it does not enable the idle state.

The **Power Manager** does not actually reenable the idle state until every call to the **DisableIdle** procedure has been matched by a call to the **EnableIdle** procedure, and then only if the user has not disabled the idle state through the Portable control panel. For this reason, you must be very careful to match each call to the **DisableIdle** procedure with a single call to the **EnableIdle** procedure. Be careful to avoid making extra calls to the **EnableIdle** procedure so that you do not inadvertently reenable the idle state while another routine needs it to remain disabled.

Calls to the **EnableIdle** procedure are not cumulative; that is, after you make several calls to the **EnableIdle** procedure, a single call to the **DisableIdle** procedure still disables the idle state. Disabling the idle state always takes precedence over enabling the idle state. A call to the **DisableIdle** procedure disables the idle state no matter how many times the EnableIdle procedure has been called and whether or not the user has enabled the idle state through the Portable control panel.

The following examples might help to clarify these concepts:

- If the application calls the **EnableIdle** routine but the user disables or has disabled the idle state, the idle state is disabled.

- If the application calls the **DisableIdle** routine and the user enables or has enabled the idle state, the idle state is disabled.

- If the application calls the **DisableIdle** routine twice in a row and then calls the **EnableIdle** routine once, the idle state is disabled.

- If the application calls the **EnableIdle** routine twice in a row and then calls the **DisableIdle** routine once, the idle state is disabled.

- If the idle state is initially enabled and if the application calls the **DisableIdle** routine twice in a row and then calls the **EnableIdle** routine twice, the **Power Manager** first disables and then reenables the idle state.

To determine whether the Macintosh Portable is currently in the idle state, read the current clock speed with the **GetCPUSpeed** function. The only values returned by the **GetCPUSpeed** function are 1 and 16, indicating the effective clock speed in megahertz.