Responding When the Sleep Queue Calls Your Routine

When you add an entry to the sleep queue and your **SleepQRec** record has the sleep flag bit set to 1, the **Power Manager** calls your routine when the **Power Manager** issues a sleep request, a sleep demand, a wakeup demand, or a sleep-request revocation. Whenever the **Power Manager** calls your routine, the A0 register contains a pointer to your sleep queue record and the D0 register has a code indicating the reason your routine is being called, as follows:

Value in D0	Meaning
1	Sleep request
2	Sleep demand
3	Wakeup demand
4	Sleep-request revocation

When your routine receives a sleep request, it must either allow or deny the request and place its response in the D0 register. To allow the sleep request, clear the D0 register to 0 before returning control to the **Power Manager**. To deny the sleep request, return a nonzero value in the D0 register. (Note that you cannot deny a sleep demand.)

If your routine or any other routine in the sleep queue denies the sleep request, the **Power Manager** sends a sleep-request revocation to each routine that it has already called with a sleep request. If none of the routines denies sleep, then the **Power Manager** sends a sleep demand to each routine in the sleep queue. Because your routine will be called a second time in any case, it is not necessary to prepare for sleep in response to a sleep request; your routine need only allow or deny the sleep request and return the result in the D0 register.

When your routine receives a sleep demand, it must prepare for the sleep state and return control to the **Power Manager** as quickly as possible. Because sleep demands are never sent by an interrupt handler, your routine can perform whatever tasks are necessary to prepare for sleep, including making calls to the **Memory Manager**. You can, for example, display an alert box to inform the user of potential problems, or you can even display a dialog box that requires the user to specify the action to be performed. However, if several applications display alert or dialog boxes, the user might become confused or alarmed. More important, if the user is not present to answer the alert box or dialog box, control is never returned to the **Power Manager**, and the Macintosh Portable does not go to sleep.

> **Warning:** If your sleep routine displays an alert box or modal dialog box, the Macintosh Portable does not enter the sleep state until the user responds. If the Macintosh Portable remains in the operating state until the battery voltage drops below a preset value, the **Power Manager** automatically shuts off all power to the system, without preserving the state of open applications or data that has not been saved to disk. To prevent this from happening, you should automatically remove your dialog box after several minutes have elapsed.

When your routine receives a wakeup demand, it must prepare for the

operating state and return control to the **Power Manager** as quickly as possible.

When your routine receives a sleep-request revocation, it must reverse any changes it made in response to the sleep request that preceded it, and return control to the **Power Manager**.

The following example checks the contents of the D0 register to determine whether your sleep queue routine is being called with a sleep request, a sleep demand, or a wakeup demand. If the D0 register contains a value of 1, indicating a sleep request, the routine clears the register to 0 to allow sleep. If the D0 register contains a 2, 3, or 4, the routine executes its sleep, wakeup, or request-cancelation procedures and terminates. If the D0 register contains any other value, the procedure just returns to the caller.

```
#include <Power.h>

long RequestSleep(void);
void SleepDemand(void);
void WakeupDemand(void);
void RevokeRequest(void);

long mySlpQProc(void)
{
    long demand;
    long result=0;

    /* D0 contains the reason we were called */
    asm {
        move.l d0,demand
    }

    switch(demand) {
        case sleepRequest:
            result = RequestSleep();
            break;
        case sleepDemand:
            SleepDemand();
            break;
        case sleepWakeUp:
            WakeupDemand();
            break;
        case sleepRevoke:
            RevokeRequest();
            break;
    }

    return result;
}

long RequestSleep(void)
{
    /* Return a 1 to deny sleep or return a 0 to permit sleep */
}
```

```
void SleepDemand(void)
{
    /* Prepare for sleep */
}

void WakeupDemand(void)
{
    /* prepare to return to operating state */
}

void RevokeRequest(void)
{
    /* reverse any changes made in response to sleep request */
}
```