NOTE: This Technical Note has been <u>retired</u>. Please see the <u>Technical Notes</u> page for current documentation.

Technical Note HW31
Sleep Queue Tasks

This Technical Note demonstrates how one can write an application to display a dialog box before a portable Macintosh goes to sleep.

[Sep 01 1992]

Introduction

Many Developers have asked about the sleep queue code that is documented in the Power Manager chapter of *Inside Macintosh* Volume VI. Many have troubles writing an application that adds an entry to the sleep queue and displays a dialog or alert box when their routine receives a sleep demand. It all looks quite simple, but after trying it myself I was able to see that it is simpler said than done.

The Power Manager chapter in *Inside Macintosh* Volume VI clearly states that "if you are writing an application that might be affected by the sleep state of the computer, you can place in the sleep queue a routine that handles whatever preparations are necessary to protect your program when the PowerBook enters the sleep state." It also goes on to say that

"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."

What the chapter fails to tell you is that there are special tricks that need to be done to display an alert box or dialog box from within an **application**.

This Technical Note describes the troubles that one may run into when attempting to use a sleep queue routine to display an alert or dialog box when a sleep demand occurs. This note also explains just what one needs to do to get around such difficult situations.

Back to top

How Do I Add an Entry to the Sleep Queue?

The Power Manager chapter in *Inside Macintosh* volume VI provides the assembly code needed to add an entry to the sleep queue.

Back to top

What Happens When the Macintosh Is Put to Sleep?

The Macintosh is usually put to sleep from within the Finder, when the user chooses Sleep from the Special menu. At this time the Power Manager walks the sleep queue and calls every subroutine that it finds there. When your subroutine gets called:

* The current A5 world is (usually) the Finder's. This means that you cannot access your applications globals from your sleep routine (unless you follow the guidelines further described in this note).

* The current resource chain is also the Finder's so you can not access your applications resources either (unless you follow the guidelines further described in this note).

Back to top

The Power Manager Calls My Subroutine From the Sleep Queue bu My Dialog Box Never Gets Drawn. Why?

To draw your dialog box, you need to restore your A5 world so you can use your application's data and jump table. To do this, you will need to save a copy of your application's A5 to a memory location your subroutine can find later. A good time to do this is when you install your sleep queue routine. The following code demonstrates how to do this:

```
MyA5ref     PROC     EXPORT          ; store our A5 in memory
        DS.L    1     ; location referenced by
                ; myA5ref
        ENDP

StoreA5Ref    PROC     EXPORT
        IMPORT    MyA5Ref
        LEA     MyA5ref,a0     ; point to our storage area
        MOVE.L    4(sp),d0
        MOVE.L    d0,(a0)     ; save a copy of our A5 there
        RTS
        ENDP


DoSleep    PROC     EXPORT
        IMPORT    MyA5Ref,sleepdmd

        LEA        MyA5ref,A0     ; point to our storage area.
        MOVE.L    (a0),-(sp)     ; push our saved A5 onto the
                                ; stack
        JSR        sleepdmd     ; call our C routine
        ADDQ       #4,sp       ;We need to clear the stack
                                ; since
        RTS             ;C routines clear the stack on
                                ; entry
        ENDP             ;Pascal routines clear the
                        ; stack on exit.
```

After this is done you will need to save the current A5 and restore A5 in your sleep subroutine as follows:

```
void SLEEPDMD(long theA5) {

    short        item = 0;
    long         oldA5;
    DialogPtr    myDialog;

    oldA5 = SetCurrentA5();
    SetA5(theA5);

    // Do whatever you need to do here to show your dialog box.
    // The one thing to remember is that if adding a dialog box in
    // your sleepQroutine, you should always make it a dialog box
    // that times out. If the user is not present to answer the
    // alert box or dialog box, control is never returned to the
    // Power Manager, and the portable does not go to sleep. This is a
    // great way to permanently burn in unwanted images in your screen!
        :
        :
    // SetA5(oldA5);

}
```

I used the following code to make sure that my dialog box timed out. You may have your own special technique for timing out your dialog box, but nevertheless you must ensure that the dialog box will time out in case the user is not present to answer the dialog box.

```
void TimeoutDialog(short dialogID, long duration)
{
    short itemHit = 0;
    long startTicks = 0;
    DialogPtr dialog = nil;

    dialog = GetNewDialog(dialogID, nil, (WindowPtr) -1);
    ShowWindow(dialog);
    startTicks = TickCount();
    while(((TickCount - startTicks) < duration) && (itemHit !=
    kOKButtonItemID))
    {
    ModalDialog(nil, &itemHit);
    }
    DisposeDialog(dialog);
}
```

Back to top

The Sleep Queue Calls My Subroutine but It Cannot Find My Dialog Box.

In the same way that the Finder's world is unable to find your application's globals (because they reside in your application's A5 world), the Finder is also unable to access your dialog box's resource. When the user chooses Sleep from the menu, the Finder tries to access the application's dialog resource. The Finder knows nothing about the application's

resource list, nor does it have access to it. It can only look to its own resource list to see if the dialog box can be found. To resolve this situation we will need to use the same trick we used above. We need to save our dialog box's resource so that it will be readily available when we call for it.

You might be tempted to create a DLOG/DITL pair, then call GetNewDialog at application startup time and save your dialogPtr away for later use by the sleep routine. However, this will not work because the dialog box will be created in your application's window layer, so when you try to show the window later, it will appear behind the frontmost application (usually the Finder) when the portable is put to sleep.

One way around this is to create the dialog box from scratch using NewDialog at sleep time. This ensures that the dialog box is created and drawn in the frontmost window layer when the PowerBook tries to go to sleep. You do not need to create the dialog box completely from scratch. You can read the items in from a resource at application startup time, save the handle away, then pass this item handle to NewDialog. Then all you have to specify is the dialog box's rectangle.

This means that you will need to save your application's A5, the handle to the dialog box's DITL, and the pointer to the dialog box itself. This will ensure that you will be properly pointing your application's window layer and that your dialog box will be the frontmost window when it appears.

Therefore, in your initialize routine your code will look as follows:

```
void Initialize()
{
    InitGraf((Ptr) &qd.thePort);
    InitFonts();
    InitWindows();
    InitMenus();
    TEInit();
    InitDialogs(nil);
    InitCursor();

    gDitlHandle = Get1Resource('DITL', 130);
    // error checking here
    STOREA5REF(gMyDialog, gDitlHandle, SetCurrentA5());
        :
        :
    /* install our assembler routine in the sleep process queue */
    myRec.sleepQLink = 0;
    myRec.sleepQType = slpQType;
    myRec.sleepQProc = (ProcPtr) &sleepQroutine;
    myRec.sleepQFlags = 0;

    SleepQInstall(&myRec);
        :
        :
} /* Initialize*/
```

And your assembly code will look as follows:

```
; Here's where we make storage for our dialog ptr, handle to our DITL and
; your A5.
MyA5ref     PROC    EXPORT
        DS.L 1    ; storage for DialogPtr
        DS.L 1    ; storage for handle to DITL
        DS.L 1    ; storage for our saved A5
         ENDP
```

```
; void StoreA5Ref(DialogPtr theDialog, Handle ditlHandle, Ptr
; SetCurrentA5());

StoreA5Ref    PROC    EXPORT
        IMPORT    MyA5Ref

         LEA      MyA5ref,a0
         MOVE.L      $4(sp),d0    ; grab our DialogPtr global
         MOVE.L      d0,(a0)     ; store it
         MOVE.L      $8(sp),d0    ; grab our ditlHandle global
         MOVE.L      d0,4(a0)    ; store it
         MOVE.L      $C(sp),d0    ; grab our saved A5
         MOVE.L      d0,8(a0)    ; store it
        RTS
        ENDP

; void StoreDialog(DialogPtr theDialog);

StoreDialog    PROC    EXPORT
        IMPORT    MyA5Ref

        LEA      MyA5ref,a0
        MOVE.L      $4(sp),d0    ; grab our DialogPtr global
        MOVE.L      d0,(a0)     ; store it
        RTS
        ENDP

; DialogPtr GetDialog();

GetDialog    PROC    EXPORT
        IMPORT    MyA5Ref

         LEA      MyA5ref,a0
         MOVE.L      (a0),d0    ; return our DialogPtr
                              ; global
        RTS
        ENDP

; void DoSleep()
; Pushes our A5 and dialogPtr onto the stack, then calls our
; sleep routine.

DoSleep     PROC     EXPORT
        IMPORT     MyA5Ref,sleepdmd

        LEA      MyA5ref,A0
        MOVE.L      (a0),-(sp)     ; push our saved DialogPtr
        MOVE.L    $4(a0), -(sp)     ; push our saved ditlHandle
        MOVE.L      $8(a0),-(sp)     ; push our saved A5
        JSR      sleepdmd     ; call our C routine
        ADD       #$C,sp     ; C routines clear the stack
                  ; on entry.
        RTS             ; Pascal routines clear the
        ENDP             ; stack on exit.
```

```
;Our sleepQroutine (from the Power Manager Chapter of Inside Macintosh VI)
; looks as follows:

sleepQroutine      PROC      EXPORT

StackFrame      RECORD      {A6Link},DECR
ParamBegin      EQU          *
ParamSize      EQU          ParamBegin-*
RetAddr      DS.L          1
A6Link      DS.L          1
LocalSize      EQU          *
    ENDR


    IMPORT      sleepdmd,WAKEUPDMD
    IMPORT      REVOKERQST
    IMPORT      DoSleep
    WITH      StackFrame

    LINK      A6,#LocalSize
    CMPI.L      #1,D0      ; Is it a sleep request?
    BNE.S      @1
    MOVE.L      #0,D0      ; We clear the register to
              ; zero to allow for sleep.
    BRA.S      Exit

@1    CMPI.L      #2,D0
    BNE.S      @2
    BSR      DoSleep      ; Here you are supposed to do
              ; whatever is needed
              ; to prepare for sleep. ie: alert user.
    BRA.S      Exit

@2    CMPI.L      #3,D0      ; Are we being told to wake up?
    BNE.S      @3
    JSR      WAKEUPDMD      ; Here you are to do whatever is
              ; needed to prepare for the operating
              ; state and return control to the
              ; Power Manager.
    BRA.S      Exit

@3    CMPI.L      #4,D0      ; Is a network sleep request
              ; being cancelled?
    BNE.S      Exit
    JSR      REVOKERQST


Exit    UNLK      A6
    MOVEA.L      (SP)+, A0
    ADDA.L      #ParamSize, SP
    JMP      (A0)

    END
```

Our sleep subroutine will therefore look as follows:

```
void SLEEPDMD(long theA5, Handle myDitl, DialogPtr myDialog) {

    short         item = 0;
    long          oldA5;
    Rect          wRect;

    oldA5 = SetCurrentA5();
    SetA5(theA5);

    //Need to build a dialog box from scratch using our previously saved DITL.
    if (myDialog == nil) {
            SetRect(&wRect, 50, 50, 400, 200);
            myDialog = NewDialog(nil, &wRect, "", false, dBoxProc,
            (WindowPtr) -1L, false, nil, myDitl);
            STOREDIALOG(myDialog);
    }
    ShowWindow(myDialog);
    SelectWindow(myDialog);
    SetPort(myDialog);
    while (item != 1) {
        ModalDialog(NULL, &item);
    }
    HideWindow(myDialog);
    SetA5(oldA5);

}/*sleepdmd*/
```

Back to top

Conclusion

This note goes over a few special tricks that the Power Manager chapter in *Inside Macintosh* Volume VI failed to cover. The Power Manager chapter states that an application could be written to add a routine in the sleep queue to put up a dialog or alert box before the system goes to sleep. It does not discuss any of the barriers that one may face when attempting to do this. The above tricks will allow you to use your application to display a dialog box before the system goes to sleep.

References

Technical Note #256: Stand-Alone Code, ad nauseum

develop Issue #12: Another Take on Globals in Standalone Code

Inside Macintosh, Volume VI, Power Manager chapter

Inside Macintosh, Volume II, Segment Loader

"Tell me if you are sleepy" code sample available on *Developer CD Series*

Back to top

Downloadables

Acrobat version of this Note (K)   Download

Technical Notes by Date | Number | Technology | Title
Developer Documentation | Technical Q&As | Development Kits | Sample Code