**SystemEvent** Used internally by Event Manager

#include <Desk.h> **Desk Manager**

```
Boolean     SystemEvent(theEvent );
EventRecord *theEvent ;     address of a 16-byte EventRecord structure
            returns         Is this a system event (i.e., a DA event)?
```

**SystemEvent** is called by the Event Manager as a way to watch for certain events and pass some of them directly to DAs, without bothering your application. This function should not be called by applications.

*theEvent* is the address of a 16-byte EventRecord. It contains information received from a previous call to **GetNextEvent**.

**Returns**: A Boolean; it identifies whether the event should be handled by an application or a DA. It is one of:

FALSE (0) This event should be passed to the application. It may be a system event the application should handle by calling **SystemClick**.

TRUE (1) This event should be handled by the system (i.e., a DA). It will not be forwarded to the application.

---

Notes: **SystemEvent** is called internally by the **GetNextEvent** function. The idea is to avoid clogging up your event loop; let the system handle such events as keystrokes, mouse-ups, updates, and activate events occuring in a DA window.

If you want to get a look at all events, you can store a 0 in the 1-byte global variable SEvtEnb (at 0x015c). This will cause **GetNextEvent** to forward all unmasked events to you.

When a DA creates a window (including a modeless dialog) it must set the windowKind to its refnum, which is a negative number. When the application calls **GetNextEvent**, as explained above, the Event Manager calls **SystemEvent**. If it returns TRUE then your DA gets the event. Since your window is a modeless dialog you would call **IsDialogEvent**, which returns FALSE What is going on is that **IsDialogEvent** (like **SystemEvent)** checks the windowKind looking for a value of 2 (for dialogs). Since your dialog's windowKind is a negative number, the DA's refnum, **IsDialogEvent** does nothing. The solution is to change the windowKind of your window to 2 before calling **IsDialogEvent**. This allows the Dialog Manager to recognize and handle the event properly. Be sure to restore the windowKind to its former value before returning to **SystemEvent**. That way, when the application calls the Dialog Manager with the same event (it should pass all events to the Dialog Manager if it has any modeless dialogs), the Dialog Manager will ignore it.