SysError Perform action normally taken upon a system error

#include <Errors.h> **System Error Codes**

void **SysError(***errID* **);**
short *errID* ; Error to simulate.

SysError is called by the system when a catastrophic error occurs. It displays the "Sorry, a system error..." dialog (with the bomb icon) and takes some precise steps, as described below.

errID is a **System Error Code**, such as dsAddressError, and so forth.

Returns: none

Notes: You usually won't use this in an application, but it is handy for testing your "resume" proc (as setup via **InitDialogs**), and some applications may find reasons to intercept this trap.

This function relies heavily on something called the "Sytem Alert Table" which gets loaded at startup. A pointer to it gets stored into the global variable DSAlertTab, at 0x2BA. For details, see Inside Macintosh, Volume 2, page 359.

The steps taken by the system error handler are as follows:

1 Save all registers
2 Store *errID* in global variable DSErrCode
3 Check DSAlertTab and if no error table in memory, draw Sad Mac Icon
3 Allocate memory for Quickdraw globals on stack and initialize a GrafPort for use in the error dialog.
4 If *errID* is < 0 the alert box will not be drawn (redrawn).
5 If *errID* matches an entry in the error alert table, display an error-specific message, otherwise, display the generic "Sorry, a system error occurred" message.
6 Draw the alert in the global-coordinates of the Rect in global variable DSAlertRect.
7 If the string IDs in the alert table are non-0, draw both strings.
8 If the Icon ID in the alert table is non-0, draw the icon.
9 If the procedure definition ID in the alert table is non-0, execute the procedure.
10 If the button definition ID in the alert table is 0, return to the caller.
11 If there's a resume procedure, increment the button definition ID by one.
12 Draw the buttons matching the button definition ID from the alert table.
13 Hit-test the buttons. If a resume-proc exists, and Resume is clicked, set the stack pointer to the value in CurStackBase (throwing away the stack), set A5 to what is was earlier, and jump to resume procedure.
14 Return to the caller.

Notice that "resume procedures" must be written with the assumption that the stack has been lost so there is no "return address". After performing whatever error recovery is possible, the routine must be able to jump to a known "starting point" in the program.

THINK Reference © 1991-1992 Symantec Corporation