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

Technical Note TB13

'LDEF' Madness

CONTENTS

This Technical Note uncovers a problem with writing Pascal list definition procedures and two (yes, count 'em, two) different methods to work around it.

[Jun 01 1990]

The Hook

List definition procedures (`'LDEF'` resources) are pieces of stand-alone code that specify the behavior of a list (i.e., how items are drawn and highlighted, etc.) You can write these procedures in a high-level language or in assembly-language, and they have an entry point with the following calling convention:

```
PROCEDURE MyList(lMessage: INTEGER; lSelect: BOOLEAN; lRect: Rect;
     lCell: Cell; lDataOffset, lDataLen: INTEGER; lHandle: ListHandle);
```

Note that the `lRect` parameter is a structure greater than four bytes in length, so you must pass a pointer to it. If you write the list definition procedure in a language like Pascal, the rectangle pointed to by `lRect` is copied into a safe, locally modifiable place.

Back to top

The Line

When an application calls `LNew`, the List Manager performs its own initialization prior to calling the list definition procedure with the `lInitMsg` message. Note that since the initialization of the list does not deal with any cells directly, the `lSelect`, `lRect`, `lCell`, `lDataOffset`, and `lDataLen` parameters are supposed to be ignored by the list definition procedure when dealing with the `lInitMsg` message.

Back to top

The Sinker

The problem is that the List Manager stuffs garbage into these parameters. Therefore, when the list definition procedure tries to copy the cell rectangle into a local copy, the pointer to the cell rectangle has a chance of being odd, which causes an address error on 68000-based machines, and it is likely to generate a bus error on all other machines.

Back to top

Solution A

A simple assembly-language header for the list definition procedure to even out the cell rectangle pointer for the `lInitMsg` message can fix the problem:

```
MainLDEF            MAIN    EXPORT
                    IMPORT  MyLDEF
; Stack Frame definition
LHandle             EQU     8                ; Handle to list data record
LDataLen            EQU     LHandle+4        ; length of data
LDataOffset         EQU     LDataLen+2       ; offset to data
LCell               EQU     LDataOffset+2    ; cell that was hit
LRect               EQU     LCell+4          ; rect to draw in
LSelect             EQU     LRect+4          ; 1=selected, 0=not selected
LMessage            EQU     LSelect+2        ; 0=Init, 1=Draw, 2=Hilite, 3=Close
LParamSize          EQU     LMessage+2-8     ; # of bytes of parameters
                    BRA.S   @0               ; enter here
; standard header
                    DC.W    0                ; flags word
                    DC.B    'LDEF'           ; type
                    DC.W    0                ; LDEF resource ID
                    DC.W    0                ; version
@0                  LINK    A6,#0
                    MOVE.W  LMessage(A6),D0  ; get the message
                    CMP.W   #lInitMsg,D0
                    BNE.S   @1               ; not initialization message
                    MOVE.L  #0,LRect(A6);    ; guarantee that this is even
@1                  UNLK    A6
                    JMP     MyLDEF
                    RTS
```

The code fragment guarantees that when the list definition procedure tries to copy the `lRect` parameter to a safe place, a bus error does not occur.

Back to top

Solution B

A simpler solution is to declare the entry point to your Pascal `'LDEF'` to be the following:

```
PROCEDURE MyList(lMessage: INTEGER; lSelect: BOOLEAN; VAR lRect: Rect;
     lCell: Cell; lDataOffset, lDataLen: INTEGER; lHandle: ListHandle);
```

This revised declaration disables the Pascal compiler's automatic copying of the rectangle data; you need to take care not to modify the cell rectangle passed in `lRect`.

Back to top

References

Inside Macintosh , Volume IV, The List Manager Package

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