

# New Technical Notes

## Macintosh



®

---

**Developer Support**

### **Macintosh Common LISP Q&As Platforms & Tools**

Revised by: Developer Support Center

October 1992

Written by: Developer Support Center

October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you’ve sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don’t have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

---

### **Highlighting Macintosh Common LISP table dialog strings**

Date Written: 8/13/91

Last reviewed: 8/1/92

I have defined a table dialog with “sequence dialog items” in the form of a list like

```
(( " string1" . (function1)) ("string2" . (function2)) ... )
```

How can I highlight the strings so that they are highlighted when the table dialog appears on the screen?

---

Create the window with (... :WINDOW-SHOW NIL ...), highlight cells with CELL-SELECT, and then ask the window to (WINDOW-SHOW).

### **MCL 2.0 heap allocation**

Date Written: 5/18/92

Last reviewed: 7/13/92

I have Macintosh Common LISP (MCL) v2.0b1. How do I repartition my memory so that I can enlarge the application heap, without enlarging the LISP heap? In other words, I would

like to keep the LISP heap the same size and give my application heap a little more room. I'm trying to allocate large blocks of memory using `_NewPtr` and am running short on memory.

—

OK, here's the story. In MCL 2.0b1 the only way to change the heap split (app/LISP) is to define the values in the LSIC resource. There's a template in the application, so if you drag the application over ResEdit you should see this resource with a template. In this one you are able to specify the application heap min/max and median values (defined in bytes). So the trick is to increase the min value which will take more heap space from the LISP environment. Be careful, though, because the stack is also using this application heap. By monitoring Using (room) you might see the values during runtime.

In MCL 2.0 final there's a parameter in save-application which defines the application heap usage. Even better, the application heap grows and takes space from the LISP heap if needed. However, the heap values don't shrink, because the MCL team didn't want to rewrite the whole Toolbox Memory Manager.