Custom Cursors

Author: Micah Bly
Version: 1.0.0, May 18, 2025

Overview

This is a system extension (INIT) for Mac System 6.x through 9.x that remaps 4 keys to act as cursor keys when a specified modifier key is held down. The extension is designed so that users who are comfortable with ResEdit can easily customize the modifier choice and/or the keys to remap. This is a successor to 68k-mac-cursor-keys and 68k-mac-cursor-keys-on-pt based on input from users wanting more control. A future version may include a control panel or standalone app for editing the mapping, but the INIT is intentionally minimalist so that it can be run on a Mac 128K without sacrificing too much memory.

Why you might find this useful

You have a Mac 128/512 keyboard (M0110)

These keyboards have no cursor keys at all. While arguably an important choice to make at the beginning of the Mac's life in order to force developers to embrace a mouse-drive UI, Apple reversed course on this fairly quickly: people like cursor keys! Now you can have cursor keys for your Mac 128 without using an unsightly Mac Plus keyboard or paying $$$ for keypad keyboard extension.

You have a Mac Plus (M0110A), ADB Standard Keyboard (M0116), IIgs keyboard, or similar

These keyboards have cursor keys, but the positioning is awkward for modern sensibilities. Use this extension to remap 4 other keys into an inverted T shape. Numpad 8-4-5-6 or 5-1-2-3 are obvious choices. WASD is another good option but only if using the Option key and not CapsLock (unless you never need W, A, S, and D).

You really like MacWrite but wish it had cursor support

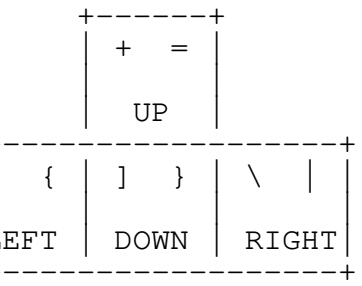
Many versions of MacWrite don't even support the cursor keys built into the Mac Plus. Some of those versions can be fooled into giving you cursors with any Mac keyboard. Note that very early versions of MacWrite don't appear to even know what to do with cursor codes.

How to use it

After installing and restarting your Mac, it will be active.
Whenever the chosen modifier key is held down, the 4 configured keys stop acting as they had before, and start acting as cursor keys (or whatever you mapped them to, if you overrode the default behavior).

Out of the box mapping

- Modifier key: CapsLock (Mode 2 -- no uppercase)
- Remapped keys: =, [,], \



Installation

1. Get it onto your Mac somehow
2. Expand the .SIT file
3. Drop the “Custom Cursors” INIT (extension) on your system folder. For System 7+, the Mac will automatically move it into your System Extensions folder.

Customizable behavior

Modifer Key

You can choose between 2 modifiers: Option key and CapsLock. For CapsLock, there are 2 modes. In the first mode, uppercasing will be applied as normal (except for the keys you remapped, if you used alpha keys). In the second mode, all uppercasing is disabled (unless you are also using SHIFT). This second mode is designed for people that want to have cursors available all the time (and have suitable 4 keys they don’t use very often -- the numpad, for example).

Keys to remap

By default, it remaps =, [,], and \, because those are a near perfect inverted T on the original Mac 128 keyboards. You can remap it to any standard set of keys though. You cannot remap a modifier key (SHIFT, etc.) to be a cursor key.

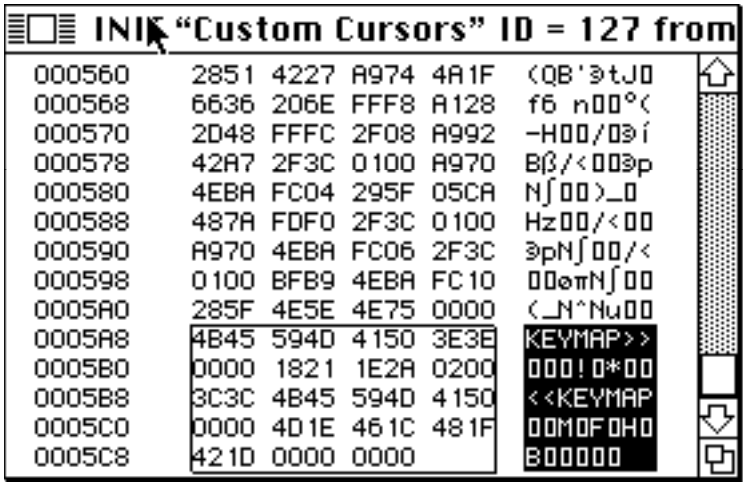
Remap target
Most people will have no use for this, but it is also possible to modify the remapped keys so they do something other than cursors. You might want to remap ` to ESC, or A to S and S to A, just to mess with all those friends that come over to use your ancient Mac tech. You only have 4 keys to play with, this is not a general key remapper utility.

How to configure with ResEdit

This version of the extension is designed to be customized by you: the hardy, ResEdit-wielding Mac Guru of days gone by. There is not currently a standalone editor, so let me know if you want to pair up on writing one.

Standard ResEdit warnings apply: it will destroy your System Folder, make your cat pee when upset, and create small black holes if used improperly.

High-level flow:
1. Open ResEdit
2. Navigate to the Custom Cursors file in your System Folder (Extensions sub-folder if System 7+).
3. Double-click the “INIT” object. It will open a 2nd window.
4. Double-click the “Custom Cursors” line in the new window. A hex view of the code from the INIT will open.
5. Scroll to the bottom of the code. Look for the “KEYMAP>>” and “<<KEYMAP” start and end markers.



6. Make changes
7. Save
8. Restart Mac to see changes.

Changing the keys to be remapped:
Each key to be remapped is represented by 1 byte in the code. The keys are in the order UP, LEFT, DOWN, RIGHT. In ResEdit, each byte is shown as a two-digit hex number. The UP key is the 3rd byte after KEYMAP>>, just after the two "00" bytes. In this example, the four keys that will be remapped are represented by 18, 21, 1E, and 2A.
1. Find the byte for the key you want to change.
2. Select the byte in ResEdit, and delete it. Be careful not to delete more than 1.
3. Look up the key you want to map in Inside Macintosh, Volume I, page 251. For example, if you want the 'W' key to be cursor UP, the key value is 13. Note that these are not ASCII or character codes; they are not affected by SHIFT, etc. These are fixed values the keyboard hardware returns to the Mac.
4. Convert the key value to Hex. Hint: if you are on a modern Mac, your Calculator.app tool has a programmer mode that converts between Hex and non-Hex. Not that I would ever need to have help with that. In this case, 'W' is 13 decimal, and '0D' hex.
5. With the cursor where you deleted the previous key value, type in the new one in hex. Be careful you delete 1 value for every 1 value you enter. You cannot select a value and hope to type over it in ResEdit. It will NOT delete what you have selected, but will add more bytes, and then the INIT will likely crash on next run.
6. Repeat until you have mapped all the keys you want.
7. Save and restart the Mac to see your changes (or keep editing if you wish to change the modifier key as well).

Changing the modifier key:
1. Locate the byte that comes immediately after the 4th remap key. In the example above, we are using CapsLock Mode 2, so it is the "02" part of the "0200" that comes directly before the closing "<<KEYMAP" marker.
2. Decide the behavior you want. 00 for Option key, 01 for CapsLock (keep uppercase behavior), 02 for CapsLock (drop uppercase behavior).
3. Delete the previous value, type in the new one.
4. Save and restart.

Changing the value mapped:
I don't really know if anyone will ever do this, but I made it theoretically possible just in case. I'm not going to write it up, but see the source code comments for more info. Hint: each key to remap needs a 2-byte character-then-key code; the 8 by tes start

after the “<<KEYMAP” marker. In the example above, “4D1E” is the first remap, with “4D” being the keyboard hardware key value for the “LEFT” key on the Mac keypad, and “1E” being the Mac ASCII character for cursor up. Depending on which key you are remapping, you might get away with putting in 00 for the first byte of each remap.

FAQs from Usenet

How do you use it?
Whenever you need cursor keys, hold down whatever you chose as the modifier, and use =, [,], \ or whatever you configured to be your new cursor keys. All other keys will operate as normal.

How can I temporarily disable it?
Hold down the mouse button on startup to disable it until the next reboot.

How do you uninstall it?
1. Open your System Folder
2. Drag the “Custom Cursors” system extension file out of the System Folder
3. Restart your Mac

Does it have a Control Panel?
No. Nothing is UI-configurable, in the interest of keeping size to a minimum. It is “easy” to change the behavior with ResEdit, however.

Which systems is it compatible with?
Good question. So far, it has been tested with:
- Macintosh Plus with 4MB, and 6.0.8
- Basilisk II - 7.5.5
- Mini vMac - various Mac II and Mac Plus configurations. Also tested with System 1.1g and a Mac 128K image.
- One user has reported it worked as expected on their actual Mac 128K with early (1 or 2) version of System.

If you test it on another system, please let me know if it works or not.

Will it work in all apps?
No, but it should work in all apps that are well-behaved, and which actually have some kind of cursor-based behavior. I noticed in

testing, for example, that MacWrite (pre-II versions) didn't do anything with the cursor input. I tried MacWrite with a Mac that had native cursor keys, they didn't do anything either! Your mileage may vary.

Why did you make those keys instead of IJKL?
IJKL does make a better inverted T for cursors, the but problem with that is that you basically can't use the keyboard for any typing at all when you are in that mode. Also, if you use Option as the modifier, Option-I is the ^ accent modifier, so you can some odd behavior when doing cursor UP.

What did you write this awesome help file with?
WriteNow 4.0, the fastest, greatest word processor for a 68K machine. Did you know WriteNow was *almost* MacWrite? It was the backup word processor contracted by Apple in case the first team couldn't get MacWrite ready for launch. It survived to go on the NeXT platform, where it's lightning fast 68K assembly base served it well. It did not, unfortunately, make the leap to PPC.

What is this INIT coded with?
The INIT is written in C using BBEdit 3.1. It is compiled with THINK C 5.0, on a genuine Mac Plus (as long as analog board repairs hold, of course). Lots of help and good ideas received from the crew in the "Hacks" and "Software" forums of 68kMLA. Code, INIT, readmes, etc. are transferred to GitHub via Fetch (FTP) to my modern Mac using the WiFi DaynaPort emulation built into BlueSCSI.