↖ Topic Index

Number Keypad

<table>
<tr><th>Operation</th><th>Keystroke</th><th>Function</th><th>Note</th></tr>
<tr><td>**PEL Number Keypad Handling**
See also:
• terminal settings
• macOS Keys</td><td colspan="3">The PEL system implements a numlock and non-numlock mode that works when Emacs operates in Graphics mode and also in Terminal (TTY) mode, despite different key behaviour. The key bindings in non-numlock mode provide access to useful keys for navigation and copy and paste operations.
⚠️ For macOS, the mechanism used by PEL currently only works for Emacs running in terminal mode under macOS Terminal.
It does not yet work for Emacs running under iTerm2. See my StackExchange question about it.
The mechanism works fine under Linux running as VM under macOS host, but then PEL cannot detect it's running under macOS.</td></tr>
<tr><td>**Testing the Keys from the shell**</td><td colspan="3">To see if a key generates an escape code sequence, run the following command from a shell and type the specific key(s).
The escape sequences generated by key press can be shown in the terminal shell by one of these 2 methods:
• typing **Control-V followed by the key**.
• Executing: `sed -n l`   Where the last character is the L lower case letter. This prints all values in caret (^) format.</td></tr>
<tr><td>**Key behaviour when Number Keypad is not num-locked**

See also: ⅀ Navigation</td><td colspan="2">**Not Numlocked**
<table>
<tr><td>pel-toggle-mac-numlock</td><td>=</td><td>/</td><td>*</td></tr>
<tr><td>pel-home</td><td>forward-line -1</td><td>pel-scroll-down</td><td>pel-kill-or-delete-marked-or-whole-line</td></tr>
<tr><td>left-char</td><td>recenter-top-bottom</td><td>right-char</td><td>pel-copy-marked-or-whole-line</td></tr>
<tr><td>pel-end</td><td>forward-line</td><td>pel-scroll-up</td><td rowspan="2"><enter></td></tr>
<tr><td colspan="2">yank</td><td>delete-char</td></tr>
</table></td><td>With PEL, right after pel-init is called, the number keypad is placed in non num-lock mode and 14 of the 18 keys take the special meaning described in the picture to the left.
• With Emacs in graphics mode the top-left-most key registers as `<clear>` and is explicitly bound by PEL to pel-toggle-mac-numlock to switch the keypad numlock mode on or off.
• With Emacs in terminal mode running under Terminal.app with the settings identified in the terminal settings which maps the `<clear>` key to NumLock key.
  • ⚠️However, Emacs operates in terminal (TTY) mode, hitting the <clear> changes the keypad key meaning but the <clear> key does not generate an event that informs Emacs. Because of that the, in macOS terminal mode the pel-toggle-mac-numlock command bound to the `<f11> #` key sequence does nothing and warns.
    • You can still hit the `<clear>` key in macOS terminal mode to toggle the NumLock mode on or off though and the number keypad keys will work appropriately.
• Four keys implement cursor functionality according to the normal cursor position.
• The pel-home and pel-end commands are available in the left column.
• The center key, the 5 key, is bound to **recenter-top-bottom**.
• The pel-scroll-down & pel-scroll-up are available in the right column.
• The big “0” key is mapped to **yank**
• The “.” key is bound to **delete-char**.
• The “-” key is pel-kill-delete-marked-or-whole-line
• The “+”key is pel-copy-marked-or-whole-line</td></tr>
<tr><td>**Key behaviour when Number Keypad is num-locked**</td><td colspan="2">**Numlocked**
<table>
<tr><td>pel-toggle-mac-numlock</td><td>=</td><td>/</td><td>*</td></tr>
<tr><td>7</td><td>8</td><td>9</td><td>-</td></tr>
<tr><td>4</td><td>5</td><td>6</td><td>+</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td rowspan="2"><enter></td></tr>
<tr><td colspan="2">0</td><td>.</td></tr>
</table></td><td>When PEL numlock mode is activated, the behaviour of the keys never to their default meaning.

Note that PEL activates non-numlock mode by default: to activate the numlock mode you can use the `<f11> #` key sequence or press the top-left-most key (in graphics mode only): this executes pel-toggle-mac-numlock.

Note:
⚠️ 🐧 With a terminal-mode Emacs running in Linux in a VM hosted on macOS, do not use `<f11> #` . Just use the `<clear>` key. Otherwise the internal state PEL sees may be out of sync and the keypad keys will not behave properly. In that case, try pressing `<clear>` again.</td></tr>
<tr><td>**Toggle PEL Keypad Numlock mode**</td><td>• `<f11> #`
• `<clear>`</td><td>(pel-toggle-mac-numlock)</td><td>Toggle PEL numlock mode for Emacs running in graphics mode.
• Complements the use of NumLock key (or `<clear>` used as NumLock).
⚠️This does not work when Emacs is running in macOS terminal mode. In that case the command simply print a warning message reminding you.
• When running in a terminal under Linux or running macOS under Terminal.app properly configured (as described in terminal settings), you can use the `<clear>` key to toggle from NumLock to non-NumLock mode but it will **not** execute the command.
  • To find out the state, press one of the keypad number keys to see if the result is insertion of a number or a cursor operation.
  • ⚠️When running Linux in a VM hosted under macOS, PEL assumes invalidly assume Linux has access to a real NumLock key. In that case just do never use this command from <f11> # and always use the `<clear>` key.
❖🐧 With PC computers the top-left-most key is an explicit num-lock key.
• See extra info related to macOS keyboard below.</td></tr>
<tr><td>**Show PEL Numlock Mode state**</td><td>`<f11> ? k #`</td><td>(pel-show-mac-numlock)</td><td>Display state of PEL Keypad num-lock mode.
• This has the same limitation as the above command.</td></tr>
<tr><td>**PEL Copy Keypad Keys**</td><td colspan="3">The “+” keypad key can also be used for copy operation. The first of the 3 binding only works when PEL is in non numlock mode, but the other 2 bindings use modifier keys and the commands are bound regardless of the PEL numlock mode.</td></tr>
<tr><td>**Copy region or line at point**

★PEL Enhanced Key★
Available in PEL non numlock mode

See also: ⅀ Cut & Paste</td><td>• `M-w`
• `<f11> = l`
• `<f11> +`
• `<kp-separator>`</td><td>(pel-copy-marked-or-whole-line)</td><td>Flexible copy to kill ring.: copy visible region if any, otherwise copy current line to kill ring.
The copy operation is controlled by the (optional) argument:
• If N = 0: copy region (regardless of whether it is visible or not.
• If a region is active/visible: copy the region's text.
• if no region is active/visible copy N lines:
  • If no argument, (N=1) copy current line.
  • If N > 0: copy current line and N-1 following lines.
  • If l < 0: copy current line and N-1 previous lines.
All copied lines are complete.
The copied text is saved in the kill-ring.
All copy operations are performed by `**kill-ring-save**’ (the original binding for that key).<br>• Replaces standard binding to kill-ring-save which only copies region.<br>• In graphics mode: text is also copied to the OS clipboard.<br>• In terminal (TTY) mode the `keypad +` key is interpreted as <kp-separator> on macOS so this key is bound to the command (in non numlock mode)</td></tr>
</table>

Operation	Keystroke	Function	Note
Copy complete word at point See also: • ⅀ Cut & Paste • ⅀ Text Modes	• `<f11> = w` • `C-<kp-add>`	(pel-copy-word-at-point)	Copy word at point. • Shows the text copied in the echo area. ☝ See table ⅀ **Text Modes** for information on text modes that affects this. • The `<f11> t m ?` command displays the mode and the `<f11> t m` prefix allows modifications of the mode. • See changing the word mode to include or exclude some characters as word delimiters: • subword-mode . To toggle that mode: `<f11> t m b` • superword-mode . To toggle that mode: `<f11> t m p`
Copy complete symbol at point See also: ⅀ **Cut & Paste**	• `<f11> = .` • `M-+` • `M-<kp-add>`	(pel-copy-symbol-at-point)	Copy symbol at point. Syntax depends on the syntax table for the buffer. • Shows the text copied in the echo area. ☝ The syntax of the symbol depends on the major mode used by the current buffer.
PEL Kill Keypad Keys	The "-" keypad key can also be used for kill operation. The binding only works when PEL is in non numlock mode.		
Kill/Delete marked region/line(s) **★PEL Enhanced Key ★** **Available in PEL non numlock mode** See also: ⅀ **Cut & Paste**	• `C-w` • `<f11> - l` • `<kp-subtract>` • `⌘-x`	(pel-kill-or-delete-marked-or-whole-line &optional N)	Flexible region/whole-line kill/delete. • N=0 := kill region (active/visible or not) • Sign of N selects operation: • positive := kill (default) • negative := delete • Select text to delete/kill based on presence of region: • if a region is marked: kill/delete region's text, • if no region: kill/delete abs(N) lines, start at point. • If operation is to kill 1 line and the line is empty, then delete line instead of killing it. • **Scenarios:** • With no arg: • with no active/visible region: kill current line, but if line is empty delete it. • with an active/visible region: kill region's text. • With arg **0**: (`M-0 C-w`) : kill region's text, whether region is active/visible or not. • With a non zero arg: • With no region active/visible: • With arg **-** : (`M- - C-w`) or (`C- - C-w`) : delete current line • With arg **- 1** : (`M- - 1 C-w`) or (`C- - 1 C-w`) : delete current line • With arg **4**: (`M - 4 C-w`) : kill 4 lines including current one. • With arg **-3**: (`M- - 3 C-w`) : delete 3 lines including current one. • With a region active/visible: • With any negative mark argument: delete the region's text. • With no argument or any positive argument: kill the region's text. ✂ This replaces the standard Emacs binding to kill-region which always kill text between mark and point, even when the region is not marked. When text is killed it is killed with kill-region, so it retains the filtering and kill ring text appending capabilities. ☝ In graphics mode this also copies text to the OS clipboard. ☝ With PEL in non-numlock mode, the "-" key on the number keypad is bound to this command. On macOS in graphics mode only: PEL rebinds `⌘-x` from (kill-region) to this command, making this easy to use key able to perform more.
**Implementation Notes **	On macOS keyboards with number keypads, the keys available when Emacs runs in graphics mode differ from the keys available when Emacs runs in terminal mode.		

** "Emacs -Q" Keypad in Graphics mode**

`<clear>`	`=`	`<kp-divide>`	`<kp-multiply>`
`<kp-7>`	`<kp-8>`	`<kp-9>`	`<kp-subtract>`
`<kp-4>`	`<kp-5>`	`<kp-6>`	`<kp-add>`
`<kp-1>`	`<kp-2>`	`<kp-3>`	`<kp-enter>`
`<kp-0>`		`<kp-decimal>`	

When Emacs is running in graphical mode, the **`<clear>`** key is available and the number keys are distinguishable from the self-insert digits. The keys on the right-most row are also distinguishable and so is the key labelled **`<kp-decimal>`**.

** "Emacs -Q" Keypad in Terminal.app TTY mode**

	`=`	`/`	`*`
`7`	`8`	`9`	`<kp-subtract>` -
`4`	`5`	`6`	`<kp-separator>` +
`1`	`2`	`3`	`RET`
`0`		`M-O n`	

When Emacs is running in terminal (TTY) mode inside Terminal.app:

- The **`<clear>`** key is not detectable.
- The **`+`**, **`/`** and **`*`** keys only register as self-insert. The digit keys register as self-insert digits but if we bind the corresponding **`<kp-digit>`** key Emacs is able to handle it properly.
- On the right-most row the **`<kp-subtract>`** is detectable, but the key below is detected as **`<kp-separator>`** instead of the normal **`<kp-add>`**.
- Also, the **`<kp-decimal>`** is not detected, instead Emacs detects the key sequence **`M-O n`**.