Cursor / Multiple-Cursors

Operation	Keystroke	Function	Note
Controlling Emacs' Cursor See also: ∑ Customize	You can control Emacs cursor color and shape when Emacs is running in graphical mode (X mode). When Emacs is running internal mode, there are no standard way to manipulate the cursor for all terminals and in some the controls are very limited. When Emacs runs in graphics mode, PEL provides the following user options cursor control for Emacs running in graphics mode: • **pel-cursor-overwrite-mode-color** : Selects the cursor color when overwrite-mode is active. Default is black on white background and white on black background. • **pel-cursort-type-when-mark** : Selects the cursor type (shape) when mark is active. Default to no cursor type change. Set it to a different type than 'cursor'. A popular setting is to use 'bar' type when mark is on to help see what is in the region.		
Open this PDF file. See also: ∑ Help/Info	`<f11> m <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open Open the local copy of the ∑ **Cursor** PDF file unless a command prefix (like `C-u`) was used. In that case it opens the Github-hosted file instead.local PDF file.
Customize PEL Cursor control	`<f11> m <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL support for cursor and multiple-cursors. • If OTHER-WINDOW is non-nil (use `C-u`), display in other window.
Customize Emacs cursor control.	`<f11> m <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs support for cursor and multiple-cursors.
Temporary change the cursor's color	`<f11> C-c`	(**pel-set-cursor-color** COLORNAME)	Set cursor to specified COLORNAME string. Prompts for the color name, support color name completion with tab. ⚠ Only available in graphics mode.
		• Ignore the request when color is not a string. • Return the COLOR string on success, nil otherwise. • ⚠ When used as an interactive command the new cursor color sticks only until the overwrite-mode is toggled. • ☝ To make the color change persist, modify the `**cursor**' or the `**pel-cursor-overwrite-mode-color**' user options.	
Permanently change the cursor's color See also: ∑ Customize	`<f11> <f2> E C-c`	(**pel-customize-cursor** &optional OTHER-WINDOW)	Quicks access to the customize buffer to set the cursor default color. • It sets the color permanently if the customization is saved. ⚠ Only available in graphics mode.
Multiple Cursors Mode	The popular multiple-cursors external package allows you to use several active cursors on multiple locations at once. • This provides a very intuitive method of modifying content of a file at several locations. • An alternative to this method is to use the keyboard macros, and other techniques. See Xah Lee comment on this. But, both can be used together. PEL provides access to multiple-cursors if the **pel-use-multiple-cursors** user option is set to **t**. It then also activates key bindings shown in this section. PEL also provides access to Visual Regexp search/replace that set the position of the multiple cursors when the corresponding appropriate user options are set to **t**: **pel-use-visual-regexp** and **pel-use-visual-regexp-steroids**.		

Activate multiple cursors on marked lines	`<f11> m m`	(**mc/edit-lines** &optional ARG)	Add one cursor to each line of the active region.
			Starts from mark and moves in straight down or up towards the line point is on.
			• What is done with lines which are not long enough is governed by 'mc/edit-lines-empty-lines'. The prefix argument ARG can be used to override this.
			  • If ARG negative, short lines will be ignored.
			  • Any other non-nil value will cause short lines to be padded.
			• Exit the multiple-cursors mode with **`RET`** or `C-g`.
			Requires the external multiple-cursors package. PEL activates it when **pel-use-multiple-cursors** is set to **t**.

Visual Regexp Search to multiple-cursors See also: ∑ Search/Replace	• `<f11> s x M` • `C-c m`	(**vr/mc-mark** REGEXP START END) This requires both visual-regexp and multiple-cursors external packages.	Convert regexp selection to multiple cursors. • First performs a Visual regexp search. When the result of the search is accepted (by hitting **`RET`**) all matches are converted to multiple cursors, which allows performing the same operations on all matches until the user quits the multiple cursor operation with `C-g`. PEL activates this command when both **pel-use-multiple-cursors** is **t** and either **pel-use-visual-regexp** or **visual-regexp-steroids** is **t**. PEL only activates the `C-c m` binding if the **pel-bind-keys-for-regexp** user option is set to **t**.
Visual Regexp Search to multiple-cursors with engine selection See also: ∑ Search/Replace	`<f11> s x M-m`	(**vr/select-mc-mark**) This requires both visual-regexp-steroids and multiple-cursors external packages.	
Highlight Current Line	Highlighting the current line may help to find the cursor when editing with big windows. These commands control line highlighting.		
Toggle line highlight mode See also: ∑ Highlight	• `<f11> b h -` • `<f11> 0`	(**hl-line-mode** &optional ARG)	Toggle highlighting of the current line (HI-Line mode) in the current buffer. • With a prefix argument ARG, enable HI-Line mode if ARG is positive, and disable it otherwise. • When same buffer is shown in several windows, the highlighting might show in each of them. Change that with pel-toggle-hl-line-sticky with: `<f11> b h s` ☝ A quick way to find where your cursor is located is to hit `<f11> 0` quickly to toggle line highlighting on and off (that key binding is easier to type than the alternative, which exists for consistency, remember that you can use `<f1> k` to get help for a specific key binding and see all the key bindings for a command, allowing you to discover its other bindings.)
Change color of highlight line for session See also: ∑ Highlight	`<f11> b h h`	(**pel-set-highlight-color** COLORNAME)	Set the colour of the highlight line used in the line highlight mode (affects all buffers). • Prompt for color name, use tab completion to show available colours with their names. • The change does not persist when Emacs is closed. To select a persistent color, then customize the **highlight** user option (see next row).
Set highlight color and attributes permanently See also: ∑ Highlight	`<f11> b h H`	(**pel-customize-highlight**)	Open the customize buffer to change the **highlight** user option color and other attributes. • As with all customizations, you can activate the change for this Emacs session or save it to make it persist across Emacs sessions. • With this you can set other attributes such as underlining (which will underline only text present in the buffer, useful to detect end-of-line whitespace), and other attributes.
Toggle line highlighting affecting all windows See also: ∑ Highlight	`<f11> b h s`	(**pel-toggle-hl-line-sticky**)	Toggle current line highlight to all windows showing the current buffer or just the current one. • Toggles the value of 'hl-line-sticky-flag' between t and nil.
Highlight current column	The following command provide a vertical line across the entire window at the cursor location. • Useful when creating tables or checking indentation manually. • vline also provides the vline-global-mode to activate the vertical line in all buffers; PEL has no binding for it because it slows Emacs too much.		
Toggle Vline Mode See also: ∑ Highlight, ∑ Hide/Show	`<f11> b h |`	(**vline-mode** &optional ARG)	Toggle the display of a vertical line spanning the entire window at the cursor column. Requires: vline.el PEL activates this when **pel-use-vline** user option is **t**.

Operation	Keystroke	Function	Note
iEdit mode	iEdit Mode - **Edit multiple regions in the same way simultaneously** 📦 This requires the **iedit** external package. 🔄 PEL downloads, installs it when any of the **pel-use-iedit** or **pel-use-lispy** user options is set to **t**.		
Toggle iedit mode See also: • **∑ Highlight** • **∑ Search/Replace**	• `C-;` • `<f11> e` • `<f11> m i`	(**iedit-mode** &optional ARG)	Toggle iEdit mode. Toggle iEdit mode. In iEdit mode you can edit all symbols in scope or region simultaneously. ⚠️ Both iEdit and Flyspell use the C-; key as their default binding. • PEL detects and reports that situation. If you see this warning modify the binding of one of the two user options.
	• This command behaves differently, depending on the mark, point, prefix argument and variable 'iedit-transient-mark-sensitive'. • If iEdit mode is off, turn iEdit mode on. • When iEdit mode is turned on, all the occurrences of the current region in the buffer (possibly narrowed) or a region are highlighted. If one occurrence is modified, the change are propagated to all other occurrences simultaneously. • If region is not active, 'iedit-default-occurrence' is called to get an occurrence candidate, according to the thing at point. It might be url, email address, markup tag or current symbol(or word). • In the above two situations, with digit prefix argument 0, only occurrences in current function are matched. This is good for renaming refactoring in programming. • You can also switch to iEdit mode from **isearch** mode directly. The current search string is used as occurrence. All occurrences of the current search string are highlighted. • With a universal prefix argument, the occurrence when iEdit mode is turned off last time in current buffer is used as occurrence. This is intended to recover last iEdit mode which is turned off. If region active, iEdit mode is limited within the current region. • With repeated universal prefix argument, the occurrence when iEdit mode is turned off last time (might be in other buffer) is used as occurrence. If region active, iEdit mode is limited within the current region. • With digital prefix argument 1, iEdit mode is limited on the current symbol or the active region, which means just one instance is highlighted. This behavior serves as a start point of incremental selection work flow. • If iEdit mode is on and region is active, iEdit mode is restricted in the region, e.g. the occurrences outside of the region is excluded. • If iEdit mode is on and region is active, with a universal prefix argument, iEdit mode is restricted outside of the region, e.g. the occurrences in the region is excluded. • Turn off iEdit mode in other situations. ⚠️ iedit does not properly disable iedit-mode restoration when desktop is restoring previous session. See 🐞 See **iedit bug #115**, and **my fix for this**. • Until this bug-fix is incorporated PEL implements a fix to ensure that iedit-mode restoration is not done during a desktop restoration.		