↖ Topic Index

Cursor / Multiple-Cursors

Operation	Keystroke	Function	Note
Controlling Emacs Cursor See also: Σ **Customize** Last updated on:	You can control Emacs cursor color and shape **when Emacs is running in graphical mode** only. Emacs provide the **cursor-type** customize option to select the default cursor shape. This is part of the **display** customization group. PEL provides the following user options cursor control for Emacs running in graphics mode: • **pel-cursor-overwrite-mode-color**: Selects cursor color in overwrite-mode. Default is black on white background and white on black background. • **pel-cursor-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 the region. Also, the **cursor-chg.el** file also exists but I have found it to slow Emacs. Therefore PEL implements its own control. 2025-11-19		
Open this PDF file. See also: Σ **Help/Info**	`<f11> m <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the Σ **Cursor** local PDF. If the prefix argument (like `C-u` or `M--`) is used, then it opens the remote GitHub hosted raw PDF instead. If the **pel-flip-help-pdf-arg** user-option is set it's the other way around.
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: provide access to the following customization groups: • **cursor** : where most cursor settings are, including the PEL user-options. • **display**: where **cursor-type** is defined. To change default cursor only. • **multiple-cursor**: for controlling the multiple cursor settings.
Change Cursor	These work in graphical emacs only		
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: Σ **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 ⚠ **see warning below.** See **demo on Emacs-Rocks** Use it to mark multiple areas and operate on all of them simultaneously. But be aware of the limitations (see the warnings below) See also: Σ **Marking** Σ **Windows**	With this set of commands you can set multiple cursors in the window to operate on each location simultaneously. This requires the **multiple-cursors** external package. With PEL, set the **pel-use-multiple-cursors** user-option set to t to install and activate it. There's 2 main methods with this package, both start by identifying the locations of the cursors: • Make a vertical selection of several lines (on any column) and use the **mc/edit-lines** (mapped to `<f11> m m`) to activate one cursor per line. • Highlight some text and then use the other 3 commands to activate a cursor before the marked area and on the next, previous or all instances of the same text in the buffer: **mc/mark-next-like-this , mc/mark-previous-like-this , mc/mark-all-like-this** There are other methods to set the cursors: • Another one is to use Visual Regexp (see below). • See also **Pi- Lispy** which supports multiple cursor on potentially different text in multiple locations of Lisp source code. ⚠ While this is fine and very useful for some editing commands be aware ***that every command issued*** from the buffer with multiple cursor actives will also be potentially applied at the location of each cursor. Therefore if you issue prompting commands, like execution via `M-x` or help request with `C-h` or `<f1>`, Emacs will prompt asking whether that command applies to all cursors. ☝ **To cancel a multi-cursor operation** you often have to issue `C-g` twice: once to cancel the text marking and then again to cancel the multi-cursor mode. ☝ The number of matches is shown on the window mode-line with something like "mc:**56**" identifying 56 matches. You can use these commands to quickly get a count of matches in the buffer: look at the count displayed in the mode-line. Just remember to cancel. To see all cursors in a larger area of the current buffer that your screen height can show, split your window with several side-by side ones (with `C-x 3`) and use the follow-mode (with `<f11> w f`) to line up the windows. Exit follow-mode and then use the multi-cursor command to perform the change. ☝ Once you have identified some matches, use the `C-'` or `<f11> m /` key to hide non-matching lines to see more of those matches then proceed with your editing commands. ⚠ Multiple cursors under Emacs does not always work properly. There are multiple edge cases and potential problems. Chris Wellons explains why. ⚠ Multi-cursor will operate **only** on the **visible** part of a buffer for most commands. See the mc match count on the modelling to confirm.		
See also: Σ **Keyboard Macros**	**Alternatives to multiple-cursor technique**: (they all work more reliably): • The **iedit-mode**, describe later in this page, can be used to replace all or some instances of selected text like variables, function names, etc… • See Xah Lee comment on this promoting the use of keyboard macros and other techniques instead of using multi-cursors. • Personally I like multi-cursor when modifying text inside a single buffer and use keyboard macros when modifying text in a buffer taking data from another or several other windows. I often use multi-cursor *and* keyboard macros together for even better leverage.		
Toggle multiple cursor mode	`<f11> m M`	(**multiple-cursors-mode** &optional ARG)	Toggle the 'Multiple-Cursors mode' minor mode. If the prefix argument is positive, enable the mode, and if it is zero or negative, disable the mode. • This is useful in some conditions when you want to disable the multiple cursor mode.
Hide un-matched	Once you have selected matched patterns with the commands below, you can hide the other, non-matched lines, showing only lines with matched area, with some lines before and after. Possibly several such sections separated by special lines. • Hiding these non matching lines help see the modifications over a large number of separated matching lines.		
Hide/show unmatched lines	`C-'` `<f11> m M-/`	(**mc-hide-unmatched-lines-mode** ARG)	Hide/show all lines that do not have one of the multiple-cursors. • Show some lines before and after. • If there are several group of matches in several areas, show separating lines between them. • Issue the command again to restore a normal, complete view to the buffer.
• **Set Multiple**	**Cursors on all instances of guessed area based on position of point**		
Mark all from point, repeat to increase number of selections	`<f11> m m`	(**mc/mark-all-like-this-dwim**)	Tries to guess what you want to mark all of. • Can be pressed multiple times to increase selection to a larger area. • You can also use `<f5>` to 'repeat' instead of retyping `<f1> m m`. • With prefix, it behaves the same as `mc/mark-all-like-this'
Mark all from point	`<f11> m M-m`	(**mc/mark-all-dwim**)	Tries even harder to guess what you want to mark all of. • If the region is active and spans multiple lines, it will behave as if `mc/mark-all-in-region'. • With the prefix ARG, it will call `mc/edit-lines' instead. • If the region is inactive or on a single line, it will behave like `mc/mark-all-like-this-dwim'.
Mark more like this at point using cursor navigation	`<f11> m .`	(**mc/mark-more-like-this-extended**)	Like mark-more-like-this, but then lets you adjust with arrows key. • The adjustments work like this: • `<up>` Mark previous like this and set direction to 'up. While going up: • `<left>` Skip past the cursor furthest up • `<right>` Remove the cursor furthest up • `<down>` Mark next like this and set direction to 'down. While going down: • `<left>` Remove the cursor furthest down • `<right>` Skip past the cursor furthest down
• **Set Multiple**	**Cursors on the some columns of multiple lines.**		☛ Note: no need to mark lines first
Mark multiple lines on a column ★★★ ⚠ Remember that multi-cursor only operate on the **visible part of the buffer.**	`<f11> m c`	(**set-rectangular-region-anchor**)	Anchors the rectangular region at point. Activates **rectangular-region-mode**. • Think of this one as `set-mark' except you're marking a rectangular region. • It is an exceedingly quick way of adding multiple cursors to multiple lines. • Issue the command then move cursor to identify area. • ☝ Unaffected by 'void' space on sorter lines! Making this very useful to: • insert or remove indentation after some leading text (like inside a table). • delete or fill a rectangle of text with any columns of text.

Operation	Keystroke	Function	Note
• **Set Multiple**	**Cursors on *marked* lines …**		☝ Note: the lines must be marked first!
… each line on same column	**<f11> m l**	(**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.
… at beginning of line	**<f11> m C-a**	(**mc/edit-beginnings-of-lines**)	Add one cursor to the beginning of each line in the active region.
… at end of line	**<f11> m C-e**	(**mc/edit-ends-of-lines**)	Add one cursor to the end of each line in the active region.
• **Set Multiple**	**Cursors on marked area like this …**		(on the same side as point on currently marked area)
… in buffer	**<f11> m a**	(**mc/mark-all-like-this**)	Find and mark all the parts of the buffer matching the currently active region.
… in defun	**<f11> m C-M-a**	(**mc/mark-all-like-this-in-defun**)	Mark all like this in defun. ⚠ The concept of "defun" depends on the major mode. The area actually selected is controlled by the function **narrow-to-defun** and its support by the current major mode.
… in region	**<f11> m ?**	(**mc/mark-all-in-region**)	Find and mark all the parts in the region matching the given search. • Prompt for string to search inside the marked region.
• **Add more**	**cursors like the current one(s)..**		
… set cursor to next instance of current match	**<f11> m n**	(**mc/mark-next-like-this** ARG)	Find and mark the next part of the buffer matching the currently active region • If no region is active add a cursor on the next line. • With negative ARG, delete the last one instead. With zero ARG, skip the last one & mark next.
… set cursor to previous instance of current match	**<f11> m p**	(**mc/mark-previous-like-this** ARG)	Find and mark the previous part of the buffer matching the currently active region • If no region is active add a cursor on the previous line • With negative ARG, delete the last one instead. With zero ARG, skip the last one & mark next.
• **Remove**	**cursors like the current one(s)…**		
… from the end of selected ones	**<f11> m N**	(**mc/unmark-next-like-this**)	Deselect next part of the buffer matching the currently active region.
… from the beginning of selected ones	**<f11> m P**	(**mc/unmark-previous-like-this**)	Deselect previous part of the buffer matching the currently active region.
• **Skip to …**			
… the next match	**<f11> m M-n**	(**mc/skip-to-next-like-this**)	Skip the current one and select the next part of the buffer matching the currently active region.
… the previous match	**<f11> m M-p**	(**mc/skip-to-previous-like-this**)	Skip the current one and select the prev part of the buffer matching the currently active region.
• **By Word:**	**Set cursor(s) to the…**		
• **Match next instance currently highlighted word, or** • **cursor on next line**	**<f11> m w**	(**mc/mark-next-word-like-this** ARG)	Find and mark the next word of the buffer matching the currently active region. • The matching region must be a whole word to be a match. • If no region is active add a cursor on the next line. • With negative ARG, delete the last one instead. • With zero ARG, skip the last one and mark next.
• **Match next instance of marked region, or** • **Match current word its next instance**	**<f11> m M-w**	(**mc/mark-next-like-this-word** ARG)	Find and mark the next part of the buffer matching the currently active region. • If no region is active, mark the word at the point and find the next match. • With negative ARG, delete the last one instead. • With zero ARG, skip the last one and mark next.
• **Match previous instance currently highlighted word, or** • **cursor on previous line**	**<f11> m W**	(**mc/mark-previous-word-like-this** ARG)	Find and mark the previous part of the buffer matching the currently active region. • The matching region must be a whole word to be a match. • If no region is active add a cursor on the previous line. • With negative ARG, delete the last one instead. • With zero ARG, skip the last one and mark next.
• **Match previous instance of marked region, or** • **Match current word its previous instance**	**<f11> m M-W**	(**mc/mark-previous-like-this-word** ARG)	Find and mark the previous part of the buffer matching the currently active region. • If no region is active, mark the word at the point and find the previous match. • With negative ARG, delete the last one instead. • With zero ARG, skip the last one and mark previous.
Match all words matching: • **word at point, or** • **currently marked area**	**<f11> m C-w**	(**mc/mark-all-words-like-this**)	Find and mark all words in the buffer matching the word at point or currently marked.
Match all words like current word, in the current defun	**<f11> m C-M-w**	(**mc/mark-all-words-like-this-in-defun**)	Mark all words like this in defun.
• **By Symbol:**	**Set cursor(s) to the…**		
• **Match next instance currently highlighted symbol, or** • **cursor on next line**	**<f11> m s**	(**mc/mark-next-symbol-like-this** ARG)	Find and mark the next symbol of the buffer matching the currently active region. • The **matching region must** be a **whole symbol** to be a match. • If no region is active add a cursor on the next line. • With negative ARG, delete the last one instead. With zero ARG, skip the last one and mark next.
• **Match next instance of marked region, or** • **Match current symbol its next instance**	**<f11> m M-s**	(**mc/mark-next-like-this-symbol** ARG)	Find and mark the next part of the buffer matching the currently active region. • If no region is active, mark the symbol at the point and find the next match. • With negative ARG, delete the last one instead. With zero ARG, skip the last one and mark next.
• **Match previous instance currently highlighted symbol, or** • **cursor on previous line**	**<f11> m S**	(**mc/mark-previous-symbol-like-this** ARG)	Find and mark the previous part of the buffer matching the currently active region. • The matching region must be a whole symbol to be a match. • If no region is active add a cursor on the previous line. • With negative ARG, delete the last one instead. With zero ARG, skip last one & mark next.
• **Match previous instance of marked region, or** • **Match current symbol its previous instance**	**<f11> m M-S**	(**mc/mark-previous-like-this-symbol** ARG)	Find and mark the previous part of the buffer matching the currently active region. • If no region is active, mark the symbol at the point and find the previous match. • With negative ARG, delete the last one instead. • With zero ARG, skip the last one and mark previous.
Match all symbols matching: • **symbol at point, or** • **currently marked area**	**<f11> m C-s**	(**mc/mark-all-symbols-like-this**)	Find and mark all symbols in the buffer matching the symbol at point or currently marked.
Match all symbols like current symbol, in the current defun	**<f11> m C-M-s**	(**mc/mark-all-symbols-like-this-in-defun**)	Mark all symbols like this in defun.
• **Align …**	See also Σ **Align**		
Vertical align with spaces	**<f11> m |**	(**mc/vertical-align-with-space**)	Aligns all cursors with whitespace to the one with the highest column number (the rightest). • Keep multiple cursors active, use this to align text while working with the multiple cursors. • Might not behave as intended if more than one cursors are on the same line.
• **Numerate at**	**all cursors**		
Insert increasing numbers at each cursor	**<f11> m i n**	(**mc/insert-numbers** ARG)	Insert increasing numbers for each cursor, starting at `**mc/insert-numbers-default**' or ARG. • **mc/insert-numbers-default** user-option sets the initial value, defaults to 0.
Insert increasing letter at each cursor	**<f11> m i l**	(**mc/insert-letters** ARG)	Insert increasing letters for each cursor, starting at 0 or ARG. • Where letter[0]=a letter[2]=c letter[26]=aa

Operation	Keystroke	Function	Note
• Sort ...			
Sort marked ares of lines	`<f11> m o`	(**mc/sort-regions**)	Sort the marked portion of the lines that have one of the cursors. ⚠️ The non-marked areas of the lines affected are **NOT** moved.
Reverse order of marked area	`<f11> m O`	(**mc/reverse-regions**)	Reverse the order of the marked regions. ⚠️ The non-marked areas of the lines affected are **NOT** moved.
• SGML tags...			
Mark current SGML tag and its pair	`<f11> m t`	(**mc/mark-sgml-tag-pair**)	Mark the SGLM tag we're in and its pair for renaming. ☝ Useful for editing SGML, HTML, etc…
Visual Regexp to multiple cursors See also: Σ **Search/Replace**	Another way to create multiple cursor is to use the following commands that perform a regular expression search to identify the location of each cursor. 📦 Both require the **multiple-cursors** external package. 🔄 With PEL, set the **pel-use-multiple-cursors** user-option set to t to install and activate it. 📦 **vr/mc-mark** requires the **visual-regexp** external package. 🔄 With PEL, set the **pel-use-visual-regexp** user-option set to t to install and activate it. 📦 **vr/select-mc-mark** requires the **visual-regexp** external package. 🔄 With PEL, set the **pel-use-visual-regexp-steroids** user-option set to t to install and activate it.		
Visual Regexp Search to multiple-cursors	• `<f11> s x M` • `C-c m`	(**vr/mc-mark** REGEXP START END)	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 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	`<f11> s x M-m`	(**vr/select-mc-mark**)	
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: Σ **Highlight**	• `<f11> h -` • `<f11> 0`	(**hl-line-mode** &optional ARG)	Toggle highlighting of the current line (Hl-Line mode) in the current buffer. • With a prefix argument ARG, enable Hl-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> 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: Σ **Highlight**	`<f11> 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: Σ **Highlight**	`<f11> 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: Σ **Highlight**	`<f11> 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: Σ **Highlight**, Σ **Hide/Show**	`<f11> 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**.
iEdit mode ★★★★	iEdit Mode - **Edit multiple regions in the same way simultaneously.** 👍 **Extremely useful!!** 📦 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: • Σ **Search/Replace** • Σ **Highlight**	• `C-;` • `<f11> e` • `<f11> h e` • `<f11> m e`	(**iedit-mode** &optional ARG)	Toggle iEdit mode: **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: modify the binding of one of them if you see it. ➤ See Σ **Search/Replace** where all the iedit-mode commands are described.