# Time Tracking

| Operation | Keystroke | Function | Note |
|---|---|---|---|
| **Time Tracking with Emacs** | Two main packages are used to track time with Emacs:<br>• Emacs built-in **timeclock** with **timelog** external library.<br>• **Org-mode with clock time**<br>PEL currently explicitly supports the first set and currently provides non-documented support for Org-mode but no extra customization.<br><br>Time information can be displayed on Emacs mode line when it is enabled. See ∑ **Mode Line** for more information. | | |
| **Open this PDF file.**<br>See also: ∑ **Help/Info** | `<f11> T <f1>` | **(pel-help-pdf** &optional OPEN-WEB-PAGE) | Open the ∑ **Time Tracking** 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. |
| **Open PEL abbreviation customization group.**<br>See also:∑ **Customize** | `<f11> T <f2>` | **(pel-customize-pel** &optional OTHER-WINDOW) | Open the PEL customize group(s) for the current context.<br>Use this to open to change PEL user option variables the activate and control the various abbreviations features.<br>• When a prefix argument (like **C-u**) opens the buffer inside another window. |
| **Customize Emacs built-in abbreviation support**<br>See also:∑ **Customize** | `<f11> T <f3>` | **(pel-customize-library** &optional OTHER-WINDOW) | Customize Emacs *time related* groups which includes: display-time, timeclock, timelog.<br>• When a prefix argument (like **C-u**) opens the buffer inside another window.<br>• Group belonging to files that have not yet been loaded are normally not accessible in Emacs and via the customize-group command. PEL, however, attempts to locate the file that defines a non-loaded customization group and will prompt you for loading the file if it finds it. |
| **Control timeclock display on Modeline** | 👓 To activate the display of this information you can set the timeclock-mode-line-display user-option on.<br> • Access the custom group buffer with `<f11> T <f3>`<br>This can also be toggled using the following command. | | |
| **Toggle display of time left in todays' workday on the mode line**<br><br>See also: ∑ **Mode Line** | `<f11> T M-d` | **(timeclock-mode-line-display** &optional ARG) | Toggle display of the amount of time left today in the mode line.<br>• With prefix ARG, turn mode line display on if and onlyif ARG is positive. Returns the new status of timeclock mode line display (non-nil means on). |
| | • If 'timeclock-use-display-time' is non-nil (the default), then the function 'display-time-mode' must be active, and the mode line will be updated whenever the time display is updated. Otherwise the timeclock will use its own sixty second timer to do its updating. | | |
| **Using Emacs built-in timeclock** | • The simple **built-in timeclock package** with provides a set of commands to define a task name, start and stop timer. This logs into time into the `~/.emacs.d/timelog` file by default. This can be modified by changing the **timeclock-file** user-option.<br>• 🔧 Activated by pel-use-timeclock user-option. Use `<f11> T <f2>` to access the custom group to set its value.<br>• timeclock can display the following information on the modeline:<br> • time left in today's workday, as controlled by the user-option and the command described just above. | | |
| **Clock in specific activity** | `C-x t i` | **(timeclock-in** &optional ARG PROJECT FIND-PROJECT) | Clock in, recording the current time moment in the timelog.<br>• With a numeric prefix ARG, record the fact that today has only that many hours in it to be worked.<br>• If ARG is a non-numeric prefix argument (non-nil, but not a number), 0 is assumed (working on a holiday or weekend).<br>• This feature only has effect the first time this function is called within a day.<br>PROJECT is the project being clocked into. Prompt for project (activity) name. |
| **Clock out** | `C-x t o` | **(timeclock-out** &optional ARG REASON FIND-REASON) | Clock out, recording the current time moment in the timelog.<br>• If a prefix ARG is given, the user has completed the project that was begun during the last time segment.<br>Prompt for the user's reason for clocking out. |
| **Change activity** | `C-x t c` | **(timeclock-change** &optional ARG PROJECT) | Change to working on a different project.<br>• This clocks out of the current project, then clocks in on a new one.<br>• With a prefix ARG, consider the previous project as finished at the time of changeover.<br>• PROJECT is the name of the last project you were working on. |
| **Re-read timeclock file** | `C-x t r` | **(timeclock-reread-log)** | Re-read the timeclock, to account for external changes inside the timelog file.<br>• Returns the new value of 'timeclock-discrepancy'.<br>👆 If you modify the content of the timelog file, run this command to update all data. |
| **Update timeclock info shown on the mode line** | `C-x t u` | **(timeclock-update-mode-line)** | Update the 'timeclock-mode-string' displayed in the mode line. |
| | • The value of 'timeclock-relative' user-option affects the display as described in that variable's documentation:<br> Whether to make reported time relative to 'timeclock-workday'.<br> For example, if the length of a normal workday is eight hours, and you work four hours on Monday, then the amount of time "remaining" on Tuesday is twelve hours -- relative to an averaged work period of eight hours -- or eight hours, non-relative. So relative time takes into account any discrepancy of time under-worked or over-worked on previous days. This only affects the timeclock mode line display.<br>• To have anything show on the mode line, first do `M-x display-time` to activate time display. | | |
| **Display time of the end of today's workday** | `C-x t w` | **(timeclock-when-to-leave-string** &optional SHOW-SECONDS TODAY-ONLY) | Return a string representing the end of today's workday.<br>• This string is relative to the value of 'timeclock-workday' which defaults to 8 hours.<br>• If SHOW-SECONDS is non-nil, the value printed/returned will include seconds. If TODAY-ONLY is non-nil, the value returned will be relative only to the time worked today, and not to past time. |
| **timelog extension for timeclock** | This external package complements the built-in timeclock, providing the ability to create time accumulation summaries, something that is lacking from timeclock.<br>📦 Requires **timelog** external package 🔧 activated when **pel-use-timeclock-timelog** user-option is turned on.<br>📦 Originally developed by **Markus Flambard and saved as a gist**, I cloned and modernized the file and **stored it in Github**. | | |
| **Print time summary for the specified date** | `C-x t l d` | **(timelog-summarize-day** DATE-STRING) | Prompts for the specified date in YYYYMMDD format.<br>• Print a time summary report for the specified day in the current buffer. |
| **Print time summary for today** | `C-x t l t` | **(timelog-summarize-today)** | Print a time summary report for today in the current buffer. |
| **Print time summary for the specified month** | `C-x t l m` | **(timelog-summarize-month** MONTH-STRING) | Prompts for the specified moth in YYYYMM format.<br>• Print a time summary report for the specified month in the current buffer. |
| **Print time summary for the specified period** | `C-x t l r` | **(timelog-summarize-range** FIRST-DAY LAST-DAY) | Print a summary for the period starting the first day and ending on the last day.<br>• Prompts for the first and last (inclusive) date in YYYYMMDD format.<br>• ⚠️ Both dates **must** be inside the timelog file otherwise the operation fails; the function uses simple date string searches to locate the first and last entries inside the file. |
| **Print day-by-day time summary for the specified period** | `C-x t l D` | **(timelog-summarize-each-day-in-range** FIRST-DAY LAST-DAY) | Print a summary for the each days inside the period starting the first day and ending on the last day.<br>• Prompts for the first and last (inclusive) date in YYYYMMDD format.<br>• ⚠️ Both dates **must** be inside the timelog file otherwise the operation fails; the function uses simple date string searches to locate the first and last entries inside the file. |
| **Display time spent on current project** | `C-x t l p` | **(timelog-current-project)** | Prints a summary of time spent in the current project on the echo area. |
| **Display time worked today** | `C-x t l e` | **(timelog-workday-elapsed)** | Prints the amount of time worked today on the echo area. |
| **Open the timeclock-file in current buffer** | `C-x t l f` | **(timelog-open-file)** | Open the timeclock-file in the current buffer. This is the file where all timeclock activity is stored. |

| Operation | Keystroke | Function | Note |
|---|---|---|---|
| **Using Org-Mode to keep track of time** | Org-mode is most probably the most flexible, powerful and extensible way to track time via tasks. 🚧 <br><br> I will provide more information about it once I get the timelog to completely working and add some features.   Org-mode is a huge topic and is very well documented. For the moment please refer to Org Mode documentation itself.  See the references below. | | |

## Time Tracking — References

| Topic & Link | Notes |
|---|---|
| **Clocking Work Time - The Org Manual** | |
| **Time Tracking in Emacs with org-clock** | Short article written by David Charte, on November 2017. |

| Operation | Keystroke | Function | Note |
|---|---|---|---|
| **Using Org-Mode to keep track of time** | Org-mode is most probably the most flexible, powerful and extensible way to track time via tasks. 🚧 <br><br> I will provide more information about it once I get the timelog to completely working and add some features.   Org-mode is a huge topic and is very well documented. For the moment please refer to Org Mode documentation itself.  See the references below. | | |