

🚧 Emacs support for Ada 🚧

Description	Keystroke	Function	Note
Ada Editing	Emacs does not provide any built-in mode for Ada programming language. PEL provides support for Ada via: 📦 The ada-mode external package 📦 PEL installs and activates it when pel-use-ada user-options is set to t. 📦 The ada-ts-mode external package 📦 PEL installs and activates it when pel-use-ada user-options is set to with-tree-sitter, Emacs >= 30 and pel-use-tree-sitter is t. See 🔗 Tree Sitter and 📦 Tree-sitter 🔗 iMenu is supported by ada-mode and ada-ts-mode. PEL: - Prioritizes the use of the excellent ada-ts-mode external package on Emacs >= 30. - Associates the .ada, .adb and .ads file extensions with Ada buffers. - Adds Ada support for 🔗 Speedbar when the pel-use-speedbar user-option is turned on. - Activates PEL functions keys under the <f12> prefix inside Ada buffers. This table shows the key bindings used by both major modes for Ada. More PEL support is still preliminary and does not document everything.🚧🚧🚧		
Last updated on:	2025-12-03		
Open this PDF file. See also: 🔗 Help/Info	<f11> SPC A <f1>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the 🔗 - Ada 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.
	<f12> <f1>		
🔗 Customize PEL Ada support	<f11> SPC A <f2>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL Ada support. If OTHER-WINDOW is non-nil (use C-u), display in another window.
	<f12> <f2>		
🔗 Customize Emacs Ada support	<f11> SPC A <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs Ada support: ada If OTHER-WINDOW is non-nil (use C-u), display in another window.
	<f12> <f3>		
Show PEL setup for Ada	<f11> SPC A ?	(pel-ada-setup-info &optional APPEND)	Display Ada setup information inside a "pel-ada-info" buffer with buttons providing quick access to the customization buffer of each variable shown. The information shown includes the value and interpretation of: - pel-use-ada (whether the classic or tree-sitter based major mode is used). - indentation and hard tab control customizable user-options To append information in the buffer instead of clearing the previous content type any prefix argument (such as C-u) before the command keystroke.
	<f12> ?		
Open file with alternate extension	M-<f11> M-f M-f	(pel-open-file-alternate)	Open a file with same name but an alternate extension: specification <--> implementation 👾The pel-alternate-extension-alist user option defines the extension pairs. If no alternate file found, save the file basename in the kill ring and prompt for the file name to open.
	M-<f12> M-f		
Comments			
Toggle display of comments in buffer or active region See also: 🔗 Comments	<f11> ; ;	(hide/show-comments-toggle &optional START END)	Toggle hiding/showing of comments in the active region or whole buffer. If the region is active then toggle in the region. Otherwise, in the whole buffer. 📦 This requires the hide-comnt.el package (see 🔗 Comments). 📦 PEL activates it when the pel-use-hide-commnt user option is t.
Insert, realign, comment/uncomment region See also: 🔗 Comments With PEL: Comment the current line with M-0 M-;	M-;	(comment-dwim ARG)	Insert or realign comment on current line (or region if a region is active). - On a single line, the comment is placed <i>after</i> the code. C-u M-; executes comment-kill
		(pel-comment-dwim ARG)	Same as comment-dwim but comments the current line with a numeric ARG or 0.
ada-mode	The key bindings specific to the ada-mode are shown here.		
Align region	C-c C-a	(ada-align)	If region is active, apply ‘align’. If not, attempt to align current construct.
Convert subprogram specifications	C-c C-b	(ada-make-subprogram-body)	Convert subprogram specification after point into a subprogram body stub.
Run the make_cmd	C-c C-c	(ada-build-make &optional CONFIRM)	Run the make_cmd project variable. - By default, this compiles and links the main program. - If CONFIRM is non-nil, prompt for user confirmation of the command.
Expand the the token-identified skeleton	C-c C-e	(wisi-skel-expand &optional NAME)	Expand the token or placeholder before point to a skeleton. - Tokens are defined by ‘wisi-skel-token-alist’; they must have symbol syntax. - A placeholder is a token enclosed in generic comment delimiters. - If the symbol before point is not in ‘wisi-skel-token-alist’, assume it is a name, and use the symbol before that as the token.
Show wise parse errors	C-c C-f	(wisi-show-parse-error)	Show current wisi-parse errors.
Set main project build command and perform the build	C-c RET	(ada-build-set-make &optional CONFIRM)	Set the main project variable to the current file, then run make_cmd. - By default, this compiles and links the new main program. - If CONFIRM is non-nil, prompt for user confirmation of the command.
Move declaration to the other file	C-c C-o	(ada-find-other-file)	Move to the corresponding declaration in another file. - If region is active, assume it contains a package name; position point on that package declaration. - If point is in the start line of a non-nested child package or subprogram declaration, position point on the corresponding parent package specification. - If point is in a context clause line, position point on the first package declaration that is mentioned. - If point is in a separate body, position point on the corresponding specification. - If point is in a subprogram body or specification, position point on the corresponding specification or body.
Refresh cached data	C-c C-q	(wisi-refresh-prj-cache NOT-FULL)	Refresh all cached data in the current project, and re-select it. With prefix arg, very slow refresh operations may be skipped.
Show all references of identifier at point	C-c C-r	(wisi-show-references &optional APPEND)	Show all references of identifier at point. With prefix, keep previous references in output buffer.
Jump to a mark popped off from the global mark ring	C-c C-s C-x C-@ C-x C-SPC <f11> . g	(pop-global-mark)	Pop off global mark ring and jump to the top location. See also: 🔗 Marking
Find file in current project	C-c C-t	(ada-find-file)	Find a file in the current project. Prompts with completion, defaults to filename at point.
Build check	C-c C-v	(ada-build-check &optional CONFIRM)	Run the check_cmd project variable. - By default, this checks the current file for syntax errors. - If CONFIRM is non-nil, prompt for user confirmation of the command.
Adjust case of symbol at point	C-c C-w	(wisi-case-adjust-at-point &optional IN-COMMENT)	If ‘wisi-auto-case’ is non-nil, adjust case of symbol at point. Also move to end of symbol. - With prefix arg, adjust case as code even if in comment or string; otherwise, capitalize words in comments and strings. - If ‘wisi-auto-case’ is nil, capitalize current word.

Description	Keystroke	Function	Note
Define exception in auto-casing	C-c C-y	(wisi-case-create-exception &optional PARTIAL)	Define a word as an auto-casing exception in the current project. - The word is the active region, or the symbol at point. - If PARTIAL is non-nil, create a partial word exception. - Prompt to choose a file from the project case-exception-files if it is a list.
Show secondary error	C-c `	(gnat-show-secondary-error)	Show the next secondary file reference in the compilation buffer. A secondary file reference is defined by text having text property 'gnat-secondary-error', set by 'gnat-compilation-filter'.
ada-ts-mode	The key bindings specific to the ada-ts-mode are shown here.		
Create comment box for function	C-c C-b	(ada-ts-mode-defun-comment-box)	Create comment box for defun enclosing point, if exists.
Find other file	C-c C-o	(ada-ts-mode-find-other-file)	Find other Ada file.
Find GNAT project file	C-c C-p	(ada-ts-mode-find-project-file)	Find GNAT Project file.
Case Control			
Toggle auto-casing		(ada-ts-auto-case-mode &optional ARG)	Toggle the 'Ada-Ts-Auto-Case mode' minor mode for auto-casing in Ada buffers. If the prefix argument is positive, enable the mode, and if it is zero or negative, disable the mode.
Apply casing at point		(ada-ts-mode-case-format-at-point)	Apply case formatting at point.
Apply cling on entire buffer		(ada-ts-mode-case-format-buffer)	Apply case formatting to entire buffer.
Apply casing intelligently		(ada-ts-mode-case-format-dwim)	Apply case formatting intelligently.
Apply casing to defined region		(ada-ts-mode-case-format-region BEG END)	Apply case formatting to region bounded by BEG and END.

Emacs & Ada — References

Document	Notes
The Ada programming language	Ada @ Wikipedia Ada home Ada-95: A Guide for C and C++ programmers, By Simon Johnston
Tools for Ada	ALIRE: Ada Library Repository GNU GNAT AdaCore Community Edition (last one created was on 2021) GNAT Pro