

Emacs support for the Seed7 Programming Language

Description	Keystroke	Function	Note	
Seed7 Editing Help & customization Alt File, Comments Template Expansion Seed7 abbreviations Marking Menu iMenu/ Speedbar Outline Navigation Xref Navigation Syntax-aware Indentation Compilation Seed7 Information	PEL supports for the Seed7 programming language uses seed7-mode . PEL also provides useful extensions & utilities that can't be implemented in the major mode.  The seed7-mode external package is installed when  the pel-use-seed7 user-option is set to t. - Seed7 files are files with .sd7 and .s7i extensions. The seed7-mode supports: - Seed7 code highlighting - Insertion of Seed7 bock or line-end comments. Ability to select which type is inserted by comment-swim. - PEL also provides a command to select the comment style allowing easy selection of different styles of multi-line comments, a feature provided by Emacs that PEL uses and provides an easy selection at prompt. - Seed7 code navigation across function and procedures as well to start/end of blocks inside functions/procedure as well as enum and struct. - imenu support, allowing use of all imenu-based navigation commands and pop-up menus. Identifies callable (functions and procedures), interfaces, enums, structs.  Speedbar support and top menu with available commands. (see  Menus) - Seed7-syntax-aware auto-indentation and auto-fill-mode are supported. - Code keyword expansion to Seed7 statements with ability to jump to next field to fill with tempo markers and navigation to those. - outline-minor-mode to list the name of Seed7 callables. See  Outline for more information. - Invocation of Seed7 compiler tools to perform static analysis or compilation of Seed7 code.			
Last updated on:	2025-12-04	All PEL functions that extend the seed7-mode have names that are shown using light green color . See  Legend for more info.		
Open this PDF file. See also:  Help/Info	<f11> SPC 7 <f1> <f12> <f1>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the  Seed7 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 Seed7 support	<f11> SPC 7 <f2> <f12> <f2>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL Seed7 support. If OTHER-WINDOW is non-nil (use C-u), display in another window.	
 Customize Emacs Seed7 support	<f11> SPC 7 <f3> • <f12> <f3> • C-c <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs Seed7 support: seed7 If OTHER-WINDOW is non-nil (use C-u), display in another window.	
Show seed7-mode version information	C-c ?	(seed7-mode-version)	Print `seed7-mode' version UTC time stamp.	
Show seed7-mode info	• C-h m • <f1> m	(describe-mode &optional BUFFER)	Display information about the seed7-mode extracted from seed7-mode key-map and docstring. See  Help/Info for more on navigating Emacs help.	
Open file with alternate extension	M-<f11> M-f M-f M-<f12> M-f	(pel-open-file-alternate)	Open a file with same name but an alternate extension: .sd7 <--> .si7  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.	
Comments	More are available to insert & manipulate comments, listed in  Comments . Some are duplicated here for convenience. The seed7-mode specific are listed first.			
Toggle between Seed7 (* block *) and # line style	C-c ;	(seed7-toggle-comment-style &optional ARG)	Toggle the Seed7 comment style between block and line comments. Optional numeric ARG, if supplied, switches to block comment style when positive, to line comment style when negative, and just toggles it when zero or left out. Note: the default style for all Seed7 buffers is controlled by the `seed7-uses-block-comment' customizable user-option. The default is line style comments.	
Insert, realign, comment/uncomment region	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. If line/region is already commented, uncomment it. C-u M-; executes comment-kill	
		(pel-comment-dwim ARG)	Same as comment-dwim but comment the current line with M-0 M-;	
Toggle display of comments in buffer or active region	<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.  Requires hide-comnt.el package (see  Comments).  activated by pel-use-hide-comnt	
Change comment style for buffer	<f11> ; s	(pel-comment-style &optional CUSTOMIZE)	Select a comment style for the buffer: prompts with the list of available styles, showing the currently used one. Apply the choice to the current buffer. With C-u prefix, open the customize buffer to control selection of the default comment style for all buffers (the comment-style user option).	
As of Emacs 30, Emacs supports 8 different comment styles, listed here: ➡➡	Emacs supports several comment styles, as specified by the comment-styles user-option (which can be modified). Some of these styles only take effect when a region of several lines is comments. By changing the style you can create the boxed comments, for instance and also uncomment the box comment with comment-swim (bound to M-;) and then change for another comment style in the same buffer. - The style selected by the command only affects the current buffer. It is not persistent. The persistent setting is the comment-style user option. 0 = plain: Start in column 0 (do not indent), as in Emacs-20 1 = indent-or-triple: Start in column 0, but only for single-char starters 2 = indent: Full comment per line, ends not aligned 3 = aligned: Full comment per line, ends aligned 4 = box: Full comment per line, ends aligned, + top and bottom 5 = extra-line: One comment for all lines, end on a line by itself 6 = multi-line: One comment for all lines, end on last commented line 7 = box-multi: One comment for all lines, + top and bottom			
Seed7 Code Template insertion & expansion	The seed7-mode supports a set of code keyword expansion to Seed7 statements with ability to jump to next field to fill with tempo markers and navigation to these fields to complete the template easily. - Code keyword expansion is performed by the seed7-complete-statement-or-indent command, bound to the <tab> key. - To use keyword expansion: type the keyword then type <tab> to expand the keyword into the corresponding code that will be properly indented. There are 2 groups of supported keywords. - The keywords shown in the first part of the table expand to their corresponding code template when the keyword is the only word on the line and point is placed just after the last keyword character.			
Top level or block declarations. Type the keyword at the beginning of the line and hit <tab> to expand the corresponding code.	inc	include statement	for	for statement
	const	constant declaration	foru	for-until statement
	var	variable declaration	fors	for-step statement
	proc	procedure declaration	fore	for-each statement
	func	function declaration	foreu	for-each statement combined with an until condition
	funcs	short function declaration	forek	for-each-key statement
	enum	enum type declaration	foreku	for-each-key statement combined with an until condition
	struct	struct type declaration	fork	for-key statement
	case	case statement	forku	for-key statement combined with an until condition
	if	if statement	repeat	repeat - until statement
	ife	if statement with an else clause	while	while statement
	ifei	if statement with an elsif clause		
	ifeie	if statement with an elsif and an else clause		
Parameter declarations Also expand with <tab>	• The second group of keywords are expanded when the keyword precedes a closing parenthesis; they are use to expand the parameter declarations.			
	in	Declaration of an in-parameter .	callbn	Declaration of a call-by-name parameter .
	inout	Declaration of an inout-parameter .	ref	Declaration of a reference-parameter .
	invar	Declaration of an in-var-parameter .	val	Declaration of a value-parameter .
	<tab>	(seed7-complete-statement-or-indent)	If point follows a valid code keyword properly located, this perform code expansion, leaving point at the first location that must be filled. In that case you can then type <backtab> to move to the next field that needs to be filled (or has already been filled). Those are tempo markers that stay in the buffer until the buffer is closed. If point is located anywhere else indent the line or selected block.	
Move to next field	<backtab>	(tempo-forward-mark)	Move point to the next tempo marker , the next template field to fill.	

Description	Keystroke		Function		Note	
Seed7-specific abbreviations	The seed7-mode supports Seed7-specific abbreviations for Emacs abbrev-mode when the seed7-support-abbrev-mode customizable user-option is on (the default).					
See also:	- All abbreviation and their text expansion are set by the seed7-abbreviations customizable user-option list. - The default list is shown below. All abbreviations start with a semi-colon. - You can modify the default and add other abbreviations through customization. - These abbreviations are <i>system</i> abbreviations, treated specially by the abbrev-mode in the sense that youcanngt modify them dynamically via the abbrev-mode commands. But you don't need to since they can be modified by customization. - Of course you can create other abbreviations that help you write code or comments. See Abbreviations for more details related to abbreviations in Emacs.					
	Abbreviations					
	To expand the abbreviations, the abbrev-mode must be active: type the abbreviation followed by a word-separating character, such as <space>, <RET>, semi-colon, period, comma, etc..					
	Use the list-abbrevs command to list all abbreviations (<f11> a M-l with PEL), including the following Seed7-specific ones. The list are shown in sorted order.					
Pragmas & in-statement keywords	pragmas		in-statement keywords		in-middle statement keywords	
	;de	decls	;fo	forward	;dt	downto
	;in	info	;n	new	;exc	exception
	;li	library	;no	noop	;lo	local
	;msg	message	;ra	raise	;pa	param
	;na	names	;rt	return	;rg	range
	;syn	syntax			;rs	result
	;sys	system	;tr	trace	;st	step
Block clause keywords	block clause keywords					
	;ct	catch	;e	else	;o	otherwise
			;ei	elsif	;w	when
Pre-defined types	pre-defined types					
	;a	array	;db	database	;rat	rational
	;bi	bigInteger	;du	duration	;rf	reference
	;br	bigRational	;en	enum	;rfl	ref_list
	;b3	bin32	;ex	expr	;s	set
	;b6	bin64	;fi	file	;sq	sqlStatement
	;bt	bitset	;fs	fileSys	;sti	string
	;bo	boolean	;fl	float	;stu	struct
	;bs	bstring	;h	hash	;tx	text
	;ca	category	;i	integer	;ti	time
	;c	char	;ob	object	;ty	type
	;cf	clib_file	;pro	process	;v	void
	;co	color	;pr	program	;pw	PRIMITIVE_WINDOW
	;cx	complex				
	Pre-defined constants	pre-defined constants				
;em		empty	;f	FALSE	;inf	Infinity
			;t	TRUE		
Pre-defined variables	pre-defined variables					
	;ck	CONSOLE_KEYBOARD	;sc	STD_CONSOLE	;sn	STD_NULL
	;gk	GRAPH_KEYBOARD	;se	STD_ERR	;so	STD_OUT
	;kb	KEYBOARD	;si	STD_IN		
Errinfo values	errinfo values					
	;ok	OKAY_NO_ERROR	;dse	DESTROY_ERROR	;me	MEMORY_ERROR
	;ae	ACTION_ERROR	;fe	FILE_ERROR	;ne	NUMERIC_ERROR
	;ce	COPY_ERROR	;ge	GRAPHIC_ERROR	;oe	OVERFLOW_ERROR
	;cre	CREATE_ERROR	;ie	INDEX_ERROR	;re	RANGE_ERROR
	;dbe	DATABASE_ERROR	;ine	IN_ERROR		
Marking	The seed7-mode support specialized marking. It is also compatible with other Emancs native and package commands. See Marking for more information.					
Mark current callable	C-M-h	(seed7-mark-defun)		Mark the current Seed7 function or procedure. - Put the mark at the end and point at the beginning. - If point is before or between 2 functions or procedure, mark the next one.		
Menu Control	The seed7-mode supports the top Menus for more information on how to activate the Emacs menu. Seed7 specific information is in the Seed7 menu section.					
iMenu Control	The seed7-mode supports iMenu symbol identification and Speedbar facility. All of iMenu and Speedbar commands are available for Seed7 code.					
Toggle iMenu grouping of callable: function & procedures listed separately or together	C-c %	(seed7-toggle-menu-callable-list)		Change the way callables are listed inside the current buffer menu. - Toggles listing them together or separately. - When listed separately the function and procedures are listed inside their own group, otherwise they are listed together.		
Toggle iMenu sorting	C-c =	(seed7-toggle-menu-sorting)		Toggle displaying menu entries in code order or sorted order.		
Seed7 Outlining	The seed7-mode supports Emacs outline minor mode. Some info is shown below See Outline for the complete list.					
Outline minor mode	<f11> M-l	(outline-minor-mode &optional ARG)		Toggle Outline minor mode. Headings are defined by outline variables.		
Hide all bodies	C-c C-t	(outline-hide-body)		Collapse the body of all blocks, leaving the first line visible.		
	<f2> t C-c @ C-t					
Show all	C-c C-a	(outline-show-all)		Show all: expand the body of all blocks.		
	<f2> a C-c @ C-a					
Show subtree	C-c C-s	(outline-show-subtree)		Show everything after this heading at deeper levels.		
	<f2> s C-c @ C-s					
Hide others	C-c C-o	(outline-hide-other)		Collapse everything except the current block.		
	<f2> o C-c @ C-o					

Description	Keystroke	Function	Note
<u>Code Navigation</u>	The seed7-mode supports syntax-aware procedure/function as well as block aware navigation commands • PEL provides some extra key bindings to Emacs native navigation commands. • The seed7-mode also supports imenu-compliant parsing which enables the ability to use a large set of navigation packages. • See navigation by symbol definition in the Navigation page for more information. • The seed7-mode navigation commands display the name and type of block found when the seed7-verbose-navigation user-option is turned on (set to t).		
<u>Shift-Selection</u>	If you press and hold the shift key while typing a movement command, that sets the mark before moving point (Emacs name for cursor) so that the region extends from the original point to its new position. This is called: Shift-Selection . • Shift selection is supported by some navigation commands, not all. The following symbols are used to identify whether the command supports shifts selection: •  This command supports shift selection in GUI and terminal mode. •  This command supports shift selection only in GUI mode. •  This command supports shift selection in GUI mode and also in terminal mode under some conditions (described in the description cell for the command). •  This command does not support shift selection. Sometimes for this you can first set the mark before moving. • Pressing the Shift key when using the key binding for commands that do not show any of these 3 arrows have no impact on the shift selection (and may be inappropriate for the command).		
<u>Move Point</u>	The following sub-sections describe how to navigate across various types of textual and syntactical entities.		
• <u>by defun</u>	The commands move point by Seed7 function and procedure definitions. • In PEL: • The <f12> cursor key mappings use <up> and <down> to move to the beginning or end of the function, procedure or other blocks. • The <f6> cursor key mapping use <up> and <down> to move to the beginning or end of the function or procedure. • The <f6> cursor key mapping use <right> and <left> to move to the beginning or end of the next/previous function or procedure. • The advantage of the <f6> and <f12> key bindings is they support Shift-Selection for Emacs in terminal mode, as opposed to the key bindings that sue the Control key which can only support Shift-Selection when Emacs is running in Graphics mode. • After moving successfully, unless you did not push the mark, you can go back to previous location with: M-`</b>, <b><f6></b> <b><f6></b> or <b><f11></b> . <b>`		
<u>Backward to beginning of defun</u>  	• <f6> <up>	(seed7-beg-of-defun &optional N SILENT DONT-PUSH-MARK)	Move backward to the beginning of a defun. • With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. • Prints the name of the function or procedure in the message area. • Supports Shift-Selection in graphics mode. <f6><up> supports it in terminal mode too.
	• C-M-a • C-M-<home> • C-[C-a • Esc C-a		
<u>Forward to end of defun</u>  	• <f6> <down>	(seed7-end-of-defun &optional N SILENT DONT-PUSH-MARK)	Move forward to next end of defun. • With argument, do it that many times. Negative argument -N means move back to Nth preceding end of defun. • Prints the name of the function or procedure in the message area. • Supports Shift-Selection in graphics mode. <f6><down> supports it in terminal mode too.
	• C-M-e • C-M-<end> • C-[C-e • Esc C-e		
<u>Move Forward to beginning of next defun</u> 	• C-c C-n • <f6> <right>	(seed7-beg-of-next-defun &optional N SILENT DONT-PUSH-MARK)	Move forward to the beginning of the next function or procedure. • With optional argument N, repeat the search that many times and succeed only when that many function or procedures are found. • A value of zero means no action. A negative value is not allowed and raises a user error. • Unless SILENT, the function prints a message showing the name of the found function or procedure. • When a new function or procedure is found the function pushes the mark unless DONT-PUSH-MARK is non-nil. Pushing the mark allows future pop to go back to the original position with C-u C-SPC. • Supports shift selection.
<u>Backward to end of previous define</u>   will be replaced	<f6> <left>	(pel-end-of-previous-defun &optional SILENT DONT-PUSH_MARK)	Move backwards to the end of the previous function definition. • Issue user error non find end of previous function unless SILENT is non-nil. • If the end of previous function is found, push the start location to the mark ring unless DONT-PUSH_MARK is non-nil. • Supports Shift-Selection.
<u>Forward to end of current block statement</u> 	<f12> <down>	(seed7-to-block-forward)	Move forward from the beginning of a Seed7 block to its end. • Supports the Seed7 if/end if, block/end block, case/end case, enum/end enum, for/end for, repeat/until, struct/end struct, while/end while. It also supports moving to the end of a function or a procedure. • Supports Shift-Selection.
<u>Backward to beginning of current block statement</u> 	<f12> <up>	(seed7-to-block-backward)	Move backward from the end of a Seed7 block to its beginning. • supports the Seed7: if/end if, block/end block, case/end case, enum/end enum, for/end for, repeat/until, struct/end struct, while/end while. It also supports moving to the end of a function or a procedure. • Supports Shift-Selection.
<u>Move to top of block</u> 	C-c C-t	(seed7-to-top-of-block)	Move point to the top of the current block; out of any nesting.
<u>Move to block backward</u> 	C-c C-a	(seed7-to-block-backward &optional AT-BEGINNING-OF-LINE DONT-PUSH-MARK)	Move backward from block end to its beginning. • Move point to the beginning of the block keyword or comment. • If point moves to the indenting area as a result, and AT-BEGINNING-OF-LINE optional argument is set, move point to the beginning of the line. • Push mark unless DONT-PUSH-MARK is non-nil. Supports shift-marking. • Return found position if found, nil if nothing found.
<u>Move to block forward</u> 	C-c C-e	(seed7-to-block-forward &optional DONT-PUSH-MARK)	Move forward from the block beginning to its end. • Handle function and forward declarations blocks. • Push mark unless DONT-PUSH-MARK is non-nil. Supports shift-marking. • Return found position or nil if nothing found.
<u>Cross Reference Navigation</u> See also: Navigation Xref	The Seed7 programming language can inspect itself: it is reflective in the sense that it can be parsed easily by the simple Seed7 code. • The seed7-mode takes advantage of this to provide quick cross reference navigation without the need of a language server or ctags parsing. • The seed7-mode provides the s7xref.sd7 Seed7 program that acts as a cross reference parser and is used as the backend for Emacs xref. • This program can be interpreted (the most flexible option) or compiled. • Identify the location of the program in the seed7-xref customizable user-option in the seed7 customization buffer (access it via C-c C or <f12> <f3>). • Once the s7xref.sd7 program properly in place (the seed7-mode default should be enough) you can use the following commands to navigate quickly across Seed7 code. • Note that the seed7-mode back-end implementation of xref-find-definition does more than only looking up for identifiers managed by the s7xref.sd7 program: it is also aware oo block and file scope and can identify the location of a local or global variable.		
<u>Find definition of identifier at point</u>	M- .	(xref-find-definitions IDENTIFIER)	Grab symbol at point and move cursor to its definition. • If there are more than one match, prompt in the "xref" buffer. • To search for a symbol entered manually, type C-u M- . • With dumb-jump this performs a search using ag, ripgrep or git grep if available.
<u>Go back to where M-. was last issued</u>	M- ,	(xref-pop-marker-stack)	• Pop back to where M-. was last invoked. • Marker depth is controlled by the xref-marker-ring-length user option.

Description	Keystroke	Function	Note
Syntax-aware automatic Indentation See also ↗ Indentation	Emacs approach to indentation control is based on what the major-mode provides. For Seed7 code, the indentation is controlled by logic implemented by the seed7-mode . Unless explicitly disabled by setting the seed7-auto-indent user-option to nil, the <tab> and <return> key perform syntax-aware automatic indentation of Seed7 code. The <return> key also supports the auto-fill-mode. - When you type code and hit the <return> key the line you just typed is indented to the location corresponding to the Seed7 indentation rules. 👉 When you press the <tab> key <i>anywhere</i> on the line, the seed7-mode code shakes if the current line is properly indented and updates the indentation if required. If it's already properly indented, then nothing occurs! - That may seem unusual behaviour for new Emacs users. But that's the way Emacs deals with indentation in all modes!! It is the expected behaviour for Emacs. - If you want to insert spaces or hard tab character, you have to use another key combination. See ↗ Indentation for more information on this topic. - Note that in Seed7 buffers, the <tab> key is bound to the seed7-complete-statement-or-indent command which check if point (the cursor location) follows a code keyword that should be expanded and if so performs the expansion instead of the indentation. Type <tab> again to perform the indentation in that case. - The number of columns used for each indentation level is controlled by the seed7-indent-width user-option, which defaults to 2. - Emacs can use hard tabs as appropriate when you activate the indent-tabs-mode. If it is off Emacs only uses space characters. - The seed7-mode also supports standard Emacs indentation commands.		
Auto-fill-mode	The seed7-mode supports Emacs auto-fill-mode, useful when typing comments. See the ↗ Fill/Justify page and the pel-comment-style command above.		
Expand keyword or indent	<tab>	(seed7-complete-statement-or-indent)	If point follows a valid code keyword properly located, this perform code expansion, leaving point at the first location that must be filled. In that case you can then type <backtab> to move to the next field that needs to be filled (or has already been filled). Those are tempo markers that stay in the buffer until the buffer is closed. If point is located anywhere else indent the line or selected block.
Indent enclosing block	C-M-q	(seed7-indent-block)	Indent the block enclosing point. Do not move point.
Refill/justify	M-q	(seed7-fill)	Refill/justify comment and string paragraph, re-indent current code block.
Compilation	The Seed7 source code is either interpreted or compiled. In both cases you can verify it's validity by performing a static check of the code, an operation that does not generate any binary file but perform the same language checking that the compiler will do.		
Static check or compile Seed7 file See ↗ Compilation Mode C-c C-c static check C-u C- C-c compile	C-c C-c <f12> c	(seed7-compile &optional <i>COMPILE</i>)	Static check current Seed7 file, show errors in compilation-mode buffer. - With optional COMPILE argument: compile the file to executable instead. - Any argument>. Use C-u C-c C-c or M-0 M-<F12> c - For example: type C-u <f12> c for compiling the file. Without the C-u prefix it just static checks the file, an operation that is much faster. - The static analysis is performed by the command identified by the seed7-checker user-option, which defaults to s7-check. - You can specify any command with or without its path. - The compilation is performed by the command identified by the seed7-compiler user-option, which defaults to s7c. - You can specify any command with or without its path. - Any detected error is shown in a *compilation* ↗ Compilation Mode buffer. Use it to navigate to the line of the code in error.

Emacs & Seed7 — References

Document	Notes	
The Seed7 Programming Language	Seed7 @ Wikipedia Seed7 Home Seed7 @ Github	Seed7 Manual Seed7 Language Reference
	Seed7 @ reddit Seed7 @ Rosetta code Seed7-users mailing list archive	
Presentations	The Seed7 Programming Language @ Youtube The Seed7 Programming Langage Presentation at CPP Vienna @ Youtube Another speech about the Seed7 Programming Language	
	Modern Extensible Languages. Daniel Zingaro, McMaster U. April 11, 2007 (pdf)	
Emacs support 🚧 is partial, not yet completed.	seed7-mode @ Github	
Other tools that support Seed7	ripgrep a very fast grep replacement - supports seed7 file types with this pull request accepted April 7 2025 - With this version of ripgrep, you can use deadgrep to identify Seed7 files by name in Emacs. See ↗ Grep ugrep another very fast grep replacement - supports seed7 files with this pull request .	