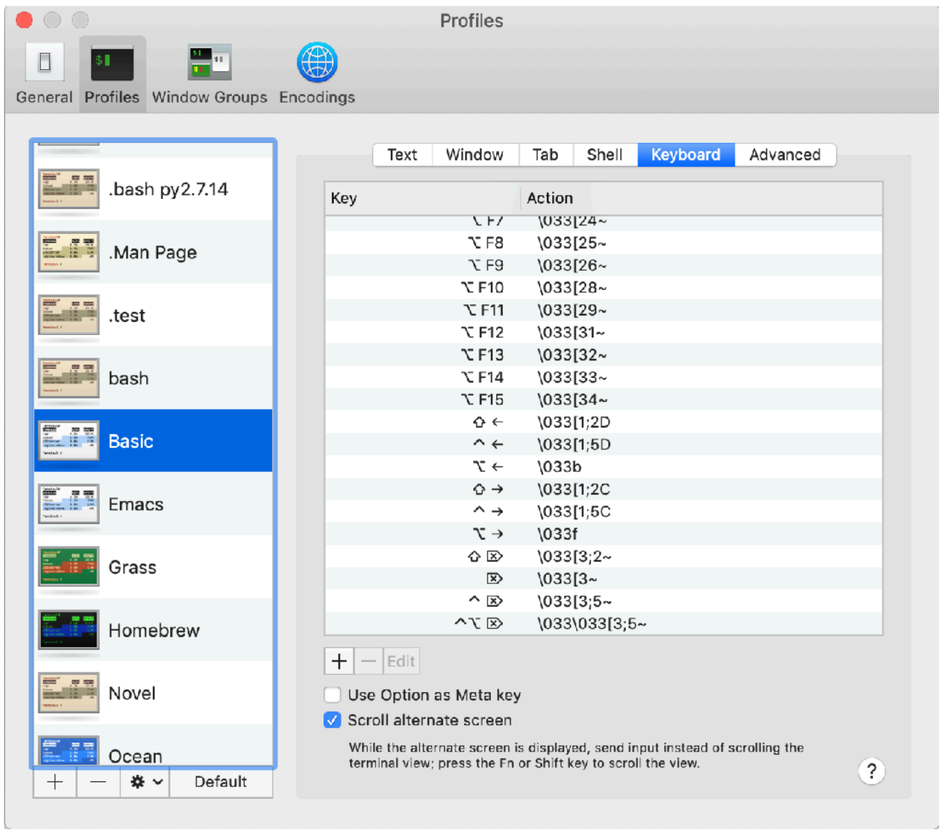
↖ Topic Index

Terminal Key Sequence Settings & Tools For investigation

Application	Type	Description
Last updated on:	2025-04-01	
macOS Tools	The following tools to investigate the keyboard behaviour in macOS terminal emulators and the OS in general are listed in this table.	
Character Viewer	Builtin macOS Application	Used to get printable symbols that represent keys. Also see this **Penn State site about Symbols and Characters**.Å
Key Codes	Third party macOS Application	Used to get Unicode key codes for the keyboard key pressed. Accessible via App Store Developer Tools.
macOS Terminal	Builtin macOS Application	• Type **^V followed by the key** in terminal to display the character sequence sent to the application for this key. • Use the Terminal Profiles, section Keyboard to add key mappings. The new mappings are available in the current terminal. If the mapping exists in Emacs it takes affect in Emacs as well. Both profiles are available as different bash shells in Terminal.app • Note that in Terminal, you can use ⌘⌥-o to toggle the meaning of the ⌥ (Alt) key between Meta and 'alternate character'.
iTerm2	Third party macOS Application	• Type **^V followed by the key** in terminal to display the character sequence sent to the application for this key. Used to check for codes that are not sent in terminal, so we can add them to **iTerm2** Profiles Keyboard mapping. • In iTerm2, the left ⌥ (Alt) key can be configured as Meta, the right ⌥ (Alt) key can be used as 'alternate character'.

Use the Terminal Preference dialog, in the Profiles section, then in terminal, to identify extra key codes for missing keys in the Terminal.App terminal emulator.

The following screenshot is an example of the dialog.



Key	Action
⌥ F7	\033[24~
⌥ F8	\033[25~
⌥ F9	\033[26~
⌥ F10	\033[28~
⌥ F11	\033[29~
⌥ F12	\033[31~
⌥ F13	\033[32~
⌥ F14	\033[33~
⌥ F15	\033[34~
⇧ ←	\033[1;2D
^ ←	\033[1;5D
⌥ ←	\033b
⇧ →	\033[1;2C
^ →	\033[1;5C
⌥ →	\033f
⇧ ⌦	\033[3;2~
⌦	\033[3~
^ ⌦	\033[3;5~
^⌥ ⌦	\033\033[3;5~

☐ Use Option as Meta key

☑ Scroll alternate screen

While the alternate screen is displayed, send input instead of scrolling the terminal view; press the Fn or Shift key to scroll the view.

The table below shows all codes I was able to configure for the macOS Terminal.app in macOS 10.14.6 (Mojave).

macOS Terminal.app & iTerm2 Keys — Profile Mappings

Key Label	Modifier / Unicode (hex)	Terminal.app Profile mapping	Sequence shown in Terminal after ^V	Add inside the Keyboard list of macOS Terminal Preferences	Sequence Shown in iTerm2 after ^V	Add to iTerm2 Profile Key Mapping	Value string extracted in xml file	Notes - all related to Emacs running inside a Terminal/shell window. See: • **macOS Terminal** • **iTerm2**
Terminal.app Keys	The following table mainly contains the Terminal input ANSI Escape Sequence key codes I was able to configure for Terminal.app on macOS 10.14.6 (Mojave), and it still works under macOS 14.7.4 (Sonoma). • This is stored inside a .terminal file, an XML PropertyList-1.0.dtd file. • The Terminal.app supports a large set of key built-in, but not all. For example it supports all the ASCII codes for keys. It also support the cursor keys. For example the <right> key corresponds to the sequence **`ESC [ C`**. This is a built-in sequence. Terminal.app supports a limited number of built in sequences. • Other sequences must be identified inside the Terminal.app Profile and Apple configures several of them, as we can see above in the screenshot of the Basic profile of Terminal.app. But again, that does not include all combination of keys we would need to use Emacs effectively. • It is possible to manually enter more sequences in Terminal.app via the dialog shown above. I have entered several key combinations by learning what sequence the terminal generates by using Terminal.app feature of displaying the sequence by first typing Control-V followed by the key sequence. This trick unfortunately does not work for all key sequences; for instance I could not find a way to distinguish the numeric keypad `/` from the main `/` key. This table list the key sequences that are not already part of the Basic Terminal.app profile that I was able to identify and added inside a new profile file. • Several people are annoyed by the current state of terminal emulators and their limitations in identifying all key combinations. Some have written description of the problem, some made proposals and some have implemented packages that try to circumvent the problem, proposing something often called "*lossless keyboard input*". See the references at the bottom of this page. Notes: • **\033** is the octal notation for decimal 27 which is the ASCII value for the **`<Esc>`** key. • **^[** is the terminal notation to describe Control-[, which also identifies the decimal value 27: the ASCII ASCII value for the **`<Esc>`** key. The **key names** have the following color codes: • **BLACK**: key available on most keyboards • **PINK** : key available on large desktop keyboards, not available on laptops • **RED** : key only available on specialized keyboards. The Modifier/Unicode(hex) column identifies the main key Unicode value, prefixed with zero, one or several modifier identifier characters. The characters are: • **$** : Shift key (⇧) • **^** : Control key (^) • **~** : Option key (⌥) • # : Numpad The Profile mapping is the escape sequence identified. • Sequences in bold are identifiable from the Terminal.app shell by pressing **Control-V** on macOS 10.14.6 (Mojave) and can therefore be entered manually inside the Terminal.app Profile. Note that this also works in some Line terminals. • It is also possible to see the values by executing **`sed -n l`** followed by the key. The last character in the command is a lower-case L.							

Key Label	Modifier / Unicode (hex)	Terminal.app Profile mapping	Sequence shown in Terminal after ^V	Add inside the Keyboard list of macOS Terminal Preferences	Sequence Shown in iTerm2 after ^V	Add to iTerm2 Profile Key Mapping	Value string extracted in xml file	Notes - all related to Emacs running inside a Terminal/shell window. See: • macOS Terminal • iTerm2
Clear (keypad)	F739	**Num Lock**		Yes		⚠️		I did not find a way to map the **`<clear>`** key to NumLock for iTerm2, as it can be done for macOS Terminal.app. • With Terminal.app I can use that to provide the PEL Emacs features described in Numkeypad . • With iTerm2 it is possible to map each numeric keypad key to something, but not provide dual meaning based on the state of what would be the NumLock state as can be done in macOS Terminal.app. At least I did not fond a way to do it and I posted a question in StackExchange about this.
F1	F704	**\033OP**	`^[OP`	Yes	`^[OP`	No need		Note: 'No need' *means that key sequence was already in profile*
F2	F705	**\033OQ**	`^[OQ`	Yes	`^[OQ`	No need		
F3	F706	**\033OR**	`^[OR`	Yes	`^[OR`	No need		
F4	F707	**\033OS**	`^[OS`	Yes	`^[OS`	No need		
F5	F708	**\033[15~**	`^[[15~`	Yes	`^[[15~`	No need		
F6	F709	**\033[17~**	`^[[17~`	Yes	`^[[17~`	No need		Note that this has the same code as ⌥F1
F7	F70A	**\033[18~**	`^[[18~`	Yes	`^[[18~`	No need		Note that this has the same code as ⌥F2
F8	F70B	**\033[19~**	`^[[19~`	Yes	`^[[19~`	No need		Note that this has the same code as ⌥F3
F9	F70C	**\033[20~**	`^[[20~`	Yes	`^[[20~`	No need		Note that this has the same code as ⌥F4
F10	F70D	**\033[21~**	`^[[21~`	Yes	`^[[21~`	No need		
F11	F70E	**\033[23~**	`^[[23~`	Yes	`^[[23~`	No need		
F12	F70F	**\033[24~**	`^[[24~`	Yes	`^[[24~`	No need		
F13	F710	**\033[25~**	`^[[25~`	Yes	`^[[1;2P`	No		Nothing found to allow Emacs to recognize these keys directly. Terminal.app and iTerm2 do not generate the same sequences for these keys. • Since I use Terminal.app most of the time, I am able to access these keys from Terminal.app, but not from iTerm2.
F14	F711	**\033[26~**	`^[[26~`	Yes	`^[[1;2Q`	No		
F15	F712	**\033[28~**	`^[[28~`	Yes	`^[[1;2R`	No		
F16	F713	**\033[29~**	`^[[29~`	Yes	`^[[1;2S`	No		
F17	F714	**\033[31~**	`^[[31~`	Yes	`^[[15;2~`	No		
F18	F715	**\033[32~**	`^[[32~`	Yes	`^[[17;2~`	No		
F19	F716	**\033[33~**	`^[[33~`	Yes	`^[[18;2~`	No		
F20	F717	**\033[34~**	`^[[34~`	Yes		No		Key not available on standard keyboards.
⇧F1	$F704	**\033[1;2P**	`^[[1;2P`	Yes	`^[[1;2P`	No		Emacs (even in graphics mode) does not support Shift-F1
⇧F2	$F705			No	`^[[1;2Q`	No		^V with these keys beeps in Terminal, but displays value in iTerm2, however it does not work inside Emacs.
⇧F3	$F706			No	`^[[1;2R`	No		
⇧F4	$F707			No	`^[[1;2S`	No		
⇧F5	$F708	**\033[15;2~**	`^[[15;2~`	Yes	`^[[15;2~`	Yes		
⇧F6	$F709	**\033[17;2~**	`^[[17;2~`	Yes	`^[[17;2~`	Yes		
⇧F7	$F70A	**\033[18;2~**	`^[[18;2~`	Yes	`^[[18;2~`	Yes		
⇧F8	$F70B	**\033[19;2~**	`^[[19;2~`	Yes	`^[[19;2~`	Yes		
⇧F9	$F70C	**\033[20;2~**	`^[[20;2~`	Yes	`^[[20;2~`	Yes		
⇧F10	$F70D	**\033[21;2~**	`^[[21;2~`	Yes	`^[[21;2~`	Yes		
⇧F11	$F70E	**\033[23;2~**	`^[[23;2~`	Yes	`^[[23;2~`	Yes		
⇧F12	$F70F	**\033[24;2~**	`^[[24;2~`	Yes	`^[[24;2~`	Yes		
⇧F13	$F710			No	`^[[1;2P`	No		Nothing found to allow Emacs to recognize these keys. By default iTerm2 generates the same sequences as for the first set of shift function keys. Both Terminal.app and iTerm2 allow setting action to these key bindings. It might be possible to find a new set of character escape sequences that could be used by Emacs to identify these keys but I did not find any so far. Therefore I'm not using them. For instance I noticed that pressing the <f13> twice in Emacs under iTerm2, I can see that it detects <f1><f13> sequence. But not when Emacs is running under Terminal.app.
⇧F14	$F711			No	`^[[1;2Q`	No		
⇧F15	$F712			No	`^[[1;2R`	No		
⇧F16	$F713			No	`^[[1;2S`	No		
⇧F17	$F714			No	`^[[15;2~`	No		
⇧F18	$F715			No	`^[[17;2~`	No		
⇧F19	$F716			No	`^[[18;2~`	No		
^F1	^F704			No		No		
^F2	^F705			No		No		
^F3	^F706			No		No		
^F4	^F707			No		No		
^F5	^F708	**\033[15;5~**	`^[[15;5~`	Yes	`^[[15;5~`	Yes		
^F6	^F709	**\033[17;5~**	`^[[17;5~`	Yes	`^[[17;5~`	Yes		
^F7	^F70A	**\033[18;5~**	`^[[18;5~`	Yes	`^[[18;5~`	Yes		
^F8	^F70B	**\033[19;5~**	`^[[19;5~`	Yes	`^[[19;5~`	Yes		
^F9	^F70C	**\033[20;5~**	`^[[20;5~`	Yes	`^[[20;5~`	Yes		
^F10	^F70D	**\033[21;5~**	`^[[21;5~`	Yes	`^[[21;5~`	Yes		
^F11	^F70E	**\033[23;5~**	`^[[23;5~`	Yes	`^[[23;5~`	Yes		
^F12	^F70F	**\033[24;5~**	`^[[24;5~`	Yes	`^[[24;5~`	Yes		
^F13	^F710			No		No		Nothing found to allow Emacs to recognize these keys. By default iTerm2 generates the same sequences as for the first
^F14	^F711			No		No		

Key Label	Modifier / Unicode (hex)	Terminal.app Profile mapping	Sequence shown in Terminal after ^V	Add inside the Keyboard list of macOS Terminal Preferences	Sequence Shown in iTerm2 after ^V	Add to iTerm2 Profile Key Mapping	Value string extracted in xml file	Notes - all related to Emacs running inside a Terminal/shell window. See: • macOS Terminal • iTerm2
^F15	^F712			No		No		set of shift function keys. Both Terminal.app and iTerm2 allow setting action to these key bindings. It might be possible to find a new set of character escape sequences that could be used by Emacs to identify these keys but I did not find any so far. Therefore I'm not using them.
^F16	^F713			No		No		
^F17	^F714			No		No		
^F18	^F715			No		No		
^F19	^F716			No		No		
⌥F1	~F704	\033[17~	^[[17~	Yes		No		This has the same code as F6. Emacs see F6
⌥F2	~F705	\033[18~	^[[18~	Yes		No		This has the same code as F7. Emacs see F7.
⌥F3	~F706	\033[19~	^[[19~	Yes		No		This has the same code as F8. Emacs see F8.
⌥F4	~F707	\033[20~	^[[20~	Yes		No		This has the same code as F9. Emacs see F9.
⌥F5	~F708	\033[15;3~	^[[15;3~	Yes	^[[15;3~	Yes		
⌥F6	~F709	\033[17;3~	^[[17;3~	Yes	^[[17;3~	Yes		
⌥F7	~F70A	\033[18;3~	^[[18;3~	Yes	^[[18;3~	Yes		
⌥F8	~F70B	\033[19;3~	^[[19;3~	Yes	^[[19;3~	Yes		
⌥F9	~F70C	\033[20;3~	^[[20;3~	Yes	^[[20;3~	Yes		
⌥F10	~F70D	\033[21;3~	^[[21;3~	Yes	^[[21;3~	Yes		
⌥F11	~F70E	\033[23;3~	^[[23;3~	Yes	^[[23;3~	Yes		
⌥F12	~F70F	\033[24;3~	^[[24;3~	Yes	^[[24;3~	Yes		
⌥F13	~F710	\033[32~	^[[32~	Yes		No		Nothing found to allow Emacs to recognize these keys. By default iTerm2 generates the same sequences as for the first set of shift function keys. Both Terminal.app and iTerm2 allow setting action to these key bindings. It might be possible to find a new set of character escape sequences that could be used by Emacs to identify these keys but I did not find any so far. Therefore I'm not using them.
⌥F14	~F711	\033[33~	^[[33~	Yes		No		
⌥F15	~F712	\033[34~	^[[34~	Yes		No		
⌥F16	~F713			No		No		
⌥F17	~F714			No		No		
⌥F18	~F715			No		No		
⌥F19	~F716			No		No		
^⌥F1				No		No		Nothing found to allow Emacs to recognize these keys. Both Terminal.app and iTerm2 allow setting action to these key bindings. It might be possible to find a new set of character escape sequences that could be used by Emacs to identify these keys but I did not find any so far that do not also mean another sequence already used. More investigation might be needed. Therefore, at this point, PEL does not use them.
^⌥F2				No		No		
^⌥F3				No		No		
^⌥F4				No		No		
^⌥F5		\033[15;7~		Yes		No		
^⌥F6		\033[17;7~		Yes		No		
^⌥F7		\033[18;7~		Yes		No		
^⌥F8		\033[19;7~		Yes		No		
^⌥F9		\033[20;7~		Yes		No		
^⌥F10		\033[21;7~		Yes		No		
^⌥F11		\033[23;7~		Yes		No		
^⌥F12		\033[24;7~		Yes		No		
^⌥F13				No		No		
^⌥F14				No		No		
^⌥F15				No		No		
^⌥F16				No		No		
^⌥F17				No		No		
^⌥F18				No		No		
^⌥F19				No		No		
^⌥⇧F1				No		No		Nothing found to allow Emacs to recognize these keys. Both Terminal.app and iTerm2 allow setting action to these key bindings. It might be possible to find a new set of character escape sequences that could be used by Emacs to identify these keys but I did not find any so far that do not also mean another sequence already used. More investigation might be needed. Therefore, at this point, PEL does not use them.
^⌥⇧F2				No		No		
^⌥⇧F3				No		No		
^⌥⇧F4				No		No		
^⌥⇧F5		\033[15;8~		Yes		No		
^⌥⇧F6		\033[17;8~		Yes		No		
^⌥⇧F7		\033[18;8~		Yes		No		
^⌥⇧F8		\033[19;8~		Yes		No		
^⌥⇧F9		\033[20;8~		Yes		No		
^⌥⇧F10		\033[21;8~		Yes		No		
^⌥⇧F11		\033[23;8~		Yes		No		
^⌥⇧F12		\033[24;8~		Yes		No		
^⌥⇧F13				No		No		Nothing found to allow Emacs to recognize these keys. Both Terminal.app and iTerm2 allow setting action to these key bindings. It might be possible to find a new set of character escape sequences that could be used by Emacs to identify these keys but I did not find any so far that do not also mean another
^⌥⇧F14				No		No		
^⌥⇧F15				No		No		
^⌥⇧F16				No		No		

Key Label	Modifier / Unicode (hex)	Terminal.app Profile mapping	Sequence shown in Terminal after ^V	Add inside the Keyboard list of macOS Terminal Preferences	Sequence Shown in iTerm2 after ^V	Add to iTerm2 Profile Key Mapping	Value string extracted in xml file	Notes - all related to Emacs running inside a Terminal/shell window. See: • macOS Terminal • iTerm2
^⌥⇧F17				No		No		sequence already used. More investigation might be needed. Therefore, at this point, PEL does not use them.
^⌥⇧F18				No		No		
^⌥⇧F19				No		No		
⌥⇧F1				No		No		Nothing found to allow Emacs to recognize these keys. It is possible to set key sequences to these keys inside the preference of both applications. However, I did not yet find an other unique escape key sequence that I could assign to those keys to provide support in Emacs. iTerm2 behaves a little better than Terminal when the keys are not defined: it passes the function key. But it could also be configured to ignore the sequence.
⌥⇧F2				No		No		
⌥⇧F3				No		No		
⌥⇧F4				No		No		
⌥⇧F5		\033[15;4~		Yes		Yes		
⌥⇧F6		\033[17;4~		Yes		Yes		
⌥⇧F7		\033[18;4~		Yes		Yes		
⌥⇧F8		\033[19;4~		Yes		Yes		
⌥⇧F9		\033[20;4~		Yes		Yes		
⌥⇧F10		\033[21;4~		Yes		Yes		Emacs does not seem to be able to distinguish this from **⌥F10**
⌥⇧F11		\033[23;4~		Yes		Yes		
⌥⇧F12		\033[24;4~		Yes		Yes		
⌥⇧F13				No		No		Nothing found to allow Emacs to recognize these keys. It is possible to set key sequences to these keys inside the preference of both applications. However, I did not yet find an other unique escape key sequence that I could assign to those keys to provide support in Emacs. iTerm2 behaves a little better than Terminal when the keys are not defined: it passes the function key. But it could also be configured to ignore the sequence.
⌥⇧F14				No		No		
⌥⇧F15				No		No		
⌥⇧F16				No		No		
⌥⇧F17				No		No		
⌥⇧F18				No		No		
⌥⇧F19				No		No		
^⇧F1	$^F704			No		No		Nothing found to allow Emacs to recognize these keys. It is possible to set key sequences to these keys inside the preference of both applications. However, I did not yet find an other unique escape key sequence that I could assign to those keys to provide support in Emacs. iTerm2 behaves a little better than Terminal when the keys are not defined: it passes the function key. But it could also be configured to ignore the sequence.
^⇧F2	$^F705			No		No		
^⇧F3	$^F706			No		No		
^⇧F4	$^F707			No		No		
^⇧F5	$^F708	\033[15;6~		Yes		Yes		
^⇧F6	$^F709	\033[17;6~		Yes		Yes		
^⇧F7	$^F70A	\033[18;6~		Yes		Yes		Emacs in either terminal application does not seem to detect.
^⇧F8	$^F70B	\033[19;6~		Yes		Yes		Emacs in either terminal application does not seem to detect.
^⇧F9	$^F70C	\033[20;6~		Yes		Yes		
^⇧F10	$^F70D	\033[21;6~		Yes		Yes		Emacs in either terminal application does not seem to detect.
^⇧F11	$^F70E	\033[23;6~		Yes		Yes		
^⇧F12	$^F70F	\033[24;6~		Yes		Yes		
^⇧F13	$^F710			No		No		Nothing found to allow Emacs to recognize these keys. Both Terminal.app and iTerm2 allow setting action to these key bindings. It might be possible to find a new set of character escape sequences that could be used by Emacs to identify these keys but I did not find any so far that do not also mean another sequence already used. More investigation might be needed. Therefore, at this point, PEL does not use them.
^⇧F14	$^F711			No		No		
^⇧F15	$^F712			No		No		
^⇧F16	$^F713			No		No		
^⇧F17	$^F714			No		No		
^⇧F18	$^F715			No		No		
^⇧F19	$^F716			No		No		
^⇧F20	$^F717			No		No		
⌥←		~~\033b~~ \033Y		**Replace**		Replace		This original key sequence here is **\033b** However this keys sequence is problematic: **\033b** corresponds to "`Esc b`" which is translated to M-b by Emacs. • The consequence is that it becomes impossible to distinguish `M-b` from `M-<left>`. • PEL provides a work-around to allow terminal to distinguish `M-b` from `M-<left>`: set the **pel-map-meta-left-right-to-Y-Z** user-option on so that PEL expects `M-Y` for the commands that are supposed to be mapped to `M-<left>`. Then setup the terminal profile to generate \033Y. • The `M-Y` key sequence was selected because it is not normally used and also because the `M-y` key sequence does not use the Shift marking concept.
^←		\033[1;5D	`^[[1;5D`	Yes		No need		
⇧←		\033[1;2D		Yes		No need		Supported: does shift-select on corresponding unshifted keys
^⌥←		\033[1;7D		Yes		Yes		
^⌥⇧←		\033[1;8D		Yes		Yes		Supported: does shift-select on corresponding unshifted keys.
⌥⇧←		\033[1;4D		Yes		Yes		Supported: does shift-select on corresponding unshifted keys.

Key Label	Modifier / Unicode (hex)	Terminal.app Profile mapping	Sequence shown in Terminal after ^V	Add inside the Keyboard list of macOS Terminal Preferences	Sequence Shown in iTerm2 after ^V	Add to iTerm2 Profile Key Mapping	Value string extracted in xml file	Notes - all related to Emacs running inside a Terminal/shell window. See: • macOS Terminal • iTerm2
^⇧←		**\033[1;6D**		**Yes**		No need		Supported: does shift-select on corresponding unshifted keys
⌥↑		**\033[1;3A**		**Yes**		Replace		iTerm2 default profile maps ⌥↑ to sending 0x1b 0x1b 0x5b 0x41 with corresponds to Esc Up. In PEL we want to distinguish Esc Up from Meta Up. Therefore PEL changes the mapping here.
^↑		**\033[1;5A**	`^[[1;5A`	**Yes**		No need		
⇧↑	$F700	**\033[1;2A**		**Yes**		No need	`^[[1;2A`	Supported: does shift-select on corresponding unshifted keys
^⌥↑		**\033[1;7A**		**Yes**		**Yes**		
^⌥⇧↑		**\033[1;8A**		**Yes**		Replace		• iTerm2 default profile has ^⌥⌘↑ sending this code. • PEL uses ^⌥⇧↑ instead for consistency. Supported: does shift-select on corresponding unshifted keys.
⌥⇧↑		**\033[1;4A**		**Yes**		**Yes**		
^⇧↑		**\033[1;6A**		**Yes**		No need		Supported: does shift-select on corresponding unshifted keys
⌥→		~~**\033f**~~ **\033Z**		**Replace**		Replace		This original key sequence here is **\033f** However this keys sequence is problematic: **\033f** corresponds to “**`Esc f`**” which is translated to M-f by Emacs. • The consequence is that it becomes impossible to distinguish **`M-f`** from **`M-<right>`**. • PEL provides a work-around to allow terminal to distinguish **`M-f`** from **`M-<right>`**: set the **pel-map-meta-left-right-to-Y-Z** user-option on so that PEL expects **`M-Z`** for the commands that are supposed to be mapped to **`M-<right>`**. Then setup the terminal profile to generate \033Z. • The **`M-Z`** key sequence was selected because it is not normally used and also because the **`M-z`** key sequence does not use the Shift marking concept.
^→		**\033[1;5C**	`^[[1;5C`	**Yes**		No need		
⇧→		**\033[1;2C**		**Yes**		No need		Supported: does shift-select on corresponding unshifted keys
^⌥→		**\033[1;7C**		**Yes**		**Yes**		
^⌥⇧→		**\033[1;8C**		**Yes**		**Yes**		Supported: does shift-select on corresponding unshifted keys
⌥⇧→		**\033[1;4C**		**Yes**		**Yes**		Supported: does shift-select on corresponding unshifted keys.
^⇧→		**\033[1;6C**		**Yes**		No need		Supported: does shift-select on corresponding unshifted keys
⌥↓		**\033[1;3B**		**Yes**		Replace		iTerm2 default profile maps ⌥↓ to sending 0x1b 0x1b 0x5b 0x42 with corresponds to Esc Down. In PEL we want to distinguish Esc Down from Meta Down. Therefore PEL changes the mapping here.
^↓		**\033[1;5B**	`^[[1;5B`	**Yes**		No need		
⇧↓		**\033[1;2B**		**Yes**		No need		Supported: does shift-select on corresponding unshifted keys
^⌥↓		**\033[1;7B**		**Yes**		**Yes**		
^⌥⇧↓		**\033[1;8B**		**Yes**		Replace		• iTerm2 default profile binds it to \033[1;5B (the escape sequence for ^↓) Supported: does shift-select on corresponding unshifted keys.
⌥⇧↓		**\033[1;4B**		**Yes**		**Yes**		
^⇧↓		**\033[1;6B**		**Yes**		No need		Supported: does shift-select on corresponding unshifted keys
^⌦		**\033[3;5~**		**Yes**		**Yes**		
⇧⌦		**\033[3;2~**		**Yes**		**Yes**		
⌦		**\033[3~**		**Yes**		No need		Note: ⌦ is the delete key.
^⌥⌦		**\033\033[3;5~**		**Yes**		**Yes: hex**		iTerm2: Send hex code: 0x1b 0x1b 0x5b 0x33 0x3b 0x35 0x7e
⇧End								
End								
^End								
^⇧End								
⌥End								
⌥⇧End								
^⌥End								Does not work in either
^⌥⇧End								
Home								
⇧Home								
^Home								Work in iTerm2, but not in Terminal.app
^⇧Home								
⌥Home								
⌥⇧Home								
^⌥Home								Work in Terminal
^⌥⇧Home								
^`	^0060	**`^[^_*b^_`**						

Key Label	Modifier / Unicode (hex)	Terminal.app Profile mapping	Sequence shown in Terminal after ^V	Add inside the Keyboard list of macOS Terminal Preferences	Sequence Shown in iTerm2 after ^V	Add to iTerm2 Profile Key Mapping	Value string extracted in xml file	Notes - all related to Emacs running inside a Terminal/shell window. See: • macOS Terminal • iTerm2
⇧^`	$^0060	`^[^_*c^_`						
^⌥`	^~0060							
⇧^⌥`	$^~0060							

Mappings available in iTerm2 not available in Terminal 🚧

Key Label	Mapping	iTerm2 Emacs	Note
End	\033[F	<end>	
⇧End	\033[1;2F		
^End	\033[1;5F	C-<end>	
^⇧End	\033[1;6F		
⌥End	\033[1;9F		
⌥⇧End	\033[1;10F		
^⌥End	\033[1;13F		
^⌥⇧End	\033[1;14F		
Home	\033[H	<home>	
⇧Home	\033[1;2H		
^Home	\033[1;5H	C-<home>	
^⇧Home	\033[1;6H		
⌥Home	\033[1;9H		
⌥⇧Home	\033[1;10H		
^⌥Home	\033[1;13H		
^⌥⇧Home	\033[1;14H		

Terminal Emulator Concepts — References

Topic & Link	Description and Notes
Background Information	The first list of references provide the knowledge on character encoding and escape sequence used by terminal emulators required to understand the way keys are encoded and the limitations of terminal emulators. Understanding this is required if one which to understand the various proposals for “lossless keyboard input” for terminal emulators.
Wikipedia - ASCII simple	A quick overview of what ASCII standard is. The ASCII table shows the control codes in the first column. Those control codes are called Control-*X* where *X* is the character shown in the third column of the table. Which makes **`Ctrl-@`**, **`CTRL-A`** up to **`Ctrl-_`** . Note that has historically been type by holding the **Control** key and the key **A**, without holding the Shift key.
Wikipedia - ASCII	More complete description of the ASCII standard and its history.
Wikipedia - ANSI escape code	The basis of terminal emulator software taking information from typed keys is the ANSI escape sequence codes, more specifically the CSI sequences. This page explains the overall concepts and their history. Note the following: • The **`ESC`** ASCII character is value 27 (base 10), which is 033 octal and 0x1B hexadecimal. • All **escape sequences** start with ESC followed by a second byte in the range 0x40-0x5F (ASCII **`@A-Z[\]^_`**) . • This is the same range of characters selected to represent control characters. • That represent a total of 32 escape sequences. • This 2 byte sequence can be replaced by a single byte, but we can’t use that now: it clashes with UTF-8 values. • The **CSI** (Control Sequence Introducer) is a sequence of several bytes: • starting with **`ESC [`** • followed by any number (could be none) of parameter bytes in the range 0x30-0x3F (ASCII **`0-9:;<=>?`**) • sequences containing the parameter bytes **`<=>?`** are considered “private” to the manufacturer. • followed by any number of *intermediate bytes* in the range of 0x20-0x2F (ASCII **`<space>`** and **`!"#$%&'()*+,-./`**) • ending with a *final byte* in the range 0x40-0x7E (ASCII **`@A–Z[\]^_`a–z{|}~`** )     • final byte in the range 0x70-0x7E (**`p–z{|}~`**) are private.
Wikipedia - Unicode range 0000-0FFF	The Unicode range 0000-0FFF holds all letters, numbers and punctuation available on US and most European keyboards. Those values, augmented with modifier keys can be used to represent values normally not supported by terminal emulators, such as **`C-S-a`** and **`` C-` ``** (which do not correspond to ASCII control characters).
Wikipedia - Unicode range E000-F8FF used as private use area	The macOS Unicode value for the cursor and function keys are in 0xF700 - 0xF72F range, which makes them part of the “private use area”.
Limitations of Terminal Emulators and improvement proposals	
🚧	TODO
Packages providing Lossless Keyboard Input	
Editing Property Lists with plutil	macOS provides the **plutil** command line utility to test, read, convert and modify macOS Property list files, like the file **~/Library/Preferences/com.apple.Terminal.plist** which contains all Terminal.app preferences. This is the file that needs to be modified to add key bindings, you can use the instructions in term-keys.el package (see below) to do so. ⚠️ Before modifying the file with plutil, **make a backup copy,** in case something goes wrong! ⚠️

Topic & Link	Description and Notes
Github - term-keys - lossless keyboard input for Emacs	This package allows creating binding to several keys that are not available to Emacs running inside a text (termcap) terminal emulator process. For example, the `C-`` and `C-/` key-chords are normally not accessible in terminal mode, simply because these do not correspond to ASCII control character values. • The term-key package can build the list of translation codes to make these key-chords accessible in terminal-base Emacs. The mechanism used is specific to the terminal emulator software, and several terminal emulators are supported, including the macOS Terminal.app. • Term-key uses a byte sequence prefix that is used for all the extra key definitions. To be able to bind the new keys in Emacs **the prefix used by term-key must not be already used in any Emacs binding**. • The default (but customizable) prefix is “\033\037" which corresponds to **`ESC C-_`** which is **`C-M-_`** binding in Emacs, that used to not be bound to anything until Emacs 28 with bounds it to undo-redo. The term-keys.el readme describes how to make modifications to the Terminal.app Property to support new keys for Emacs. See the macOS Terminal section of the file (⚠ make a backup of the file first!). • To edit a macOS plist file, use the open command from the shell. It will open the plist file inside Xcode.