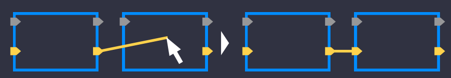
RAFCON
Cheatsheet

https://github.com/DLR-RM/RAFCON

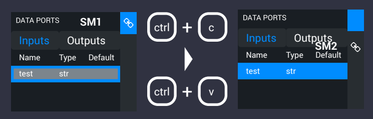
Important Keyboard Shortcuts (can be customized in preferences)

esc	abort (for e.g. editor, pop-ups, dialogs...)
ctrl + a	add
ctrl + alt + b	add barrier state
ctrl + alt + e	add execution state
ctrl + alt + h	add hierarchy state
ctrl + alt + c	add preemptive state
ctrl + alt + n	add data input
ctrl + alt + u	add data output
ctrl + alt + o	add logic outcome
ctrl + shift + v	add scoped variable
ctrl + p	show aborted and preempted
ctrl + d	show data flows
ctrl + l	show data values
ctrl + t	show logical flows
ctrl + g	group states in hierarchy
ctrl + e	set state to start state
ctrl + shift + c	add logic flow from closest sibling
ctrl + shift + v	add logic flow to closest sibling
ctrl + shift + b	add logic flow to parent
f5	start
ctrl + r	start from selected state
f4	step
f6	step mode
f8	stop
f7	pause
ctrl + shift + k	run only selected state
ctrl + shift + r	run until selected state
🖱	close tabs (when clicked on)
ctrl + w	close current tab

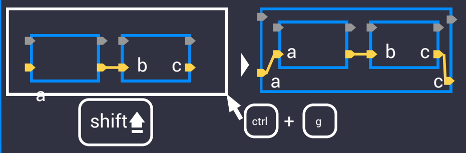
GUI Tipps / Tricks



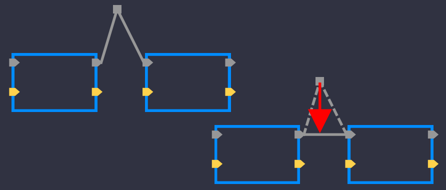
Dragging from a dataport into a state creates the dataport with the same data type and name



Data ports can be copied and pasted when the row is selected

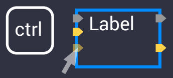


Grouping multiple selected states will add all required dataports to hierarchy state

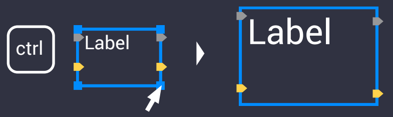


You can make via-points in the connections

To remove via-points again, bring them close to the straight line, this will automatically remove them

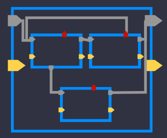


Labels and ports can be moved by clicking strg

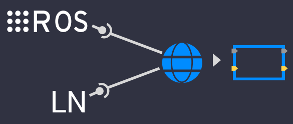


When rescaling pressing strg scales the whole statemachine nicely

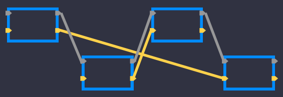
Guidelines / Best Practices



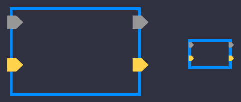
More hierarchy provides more semantic scope and readable local error handling



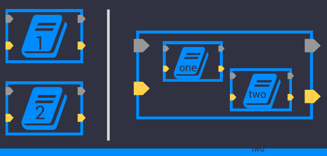
Keeping rafcon stateless and only orchestrating system capabilities (such as service calls) helps in maintaining observability



Organising states in a „snake“ style helps keeping an overview of the execution- and dataflow



Visual state size is important to keep track of relevance. Either try to be consistent with the size or actively adjust the size.



Using libraries allows to reuse functionality. Renaming libraries directly helps to understand the contextual use of the library.

Dev Tipps for speedup

core_config.yaml:

LOAD_SM_WITH_CHECKS: false	loads statemachines way faster
NO_PROGRAMMATIC_CHANGE_OF_LIBRARY_STATES_PERFORMED: true	only use this if you are NOT using dynamic state machine generation

gui_config.yaml:

MAX_VISIBLE_LIBRARY_HIERARCHY: 2	The more hierarchy you display the slower
NO_FULLY_RECURSIVE_LIBRARY_MODEL: true	if set to false models for all states will be loaded

Logging / Execution History

Rafcon logs **inputs** and **outputs** in the **Execution History** . This panel can be opened seperately and **inputs** and **outputs** can be investigated by clicking on those with 🖱 .

Some tipps for the config.yaml

IN_MEMORY_EXECUTION_HISTORY_ENABLE	if set to true, execution history is saved and can be opened with raflog