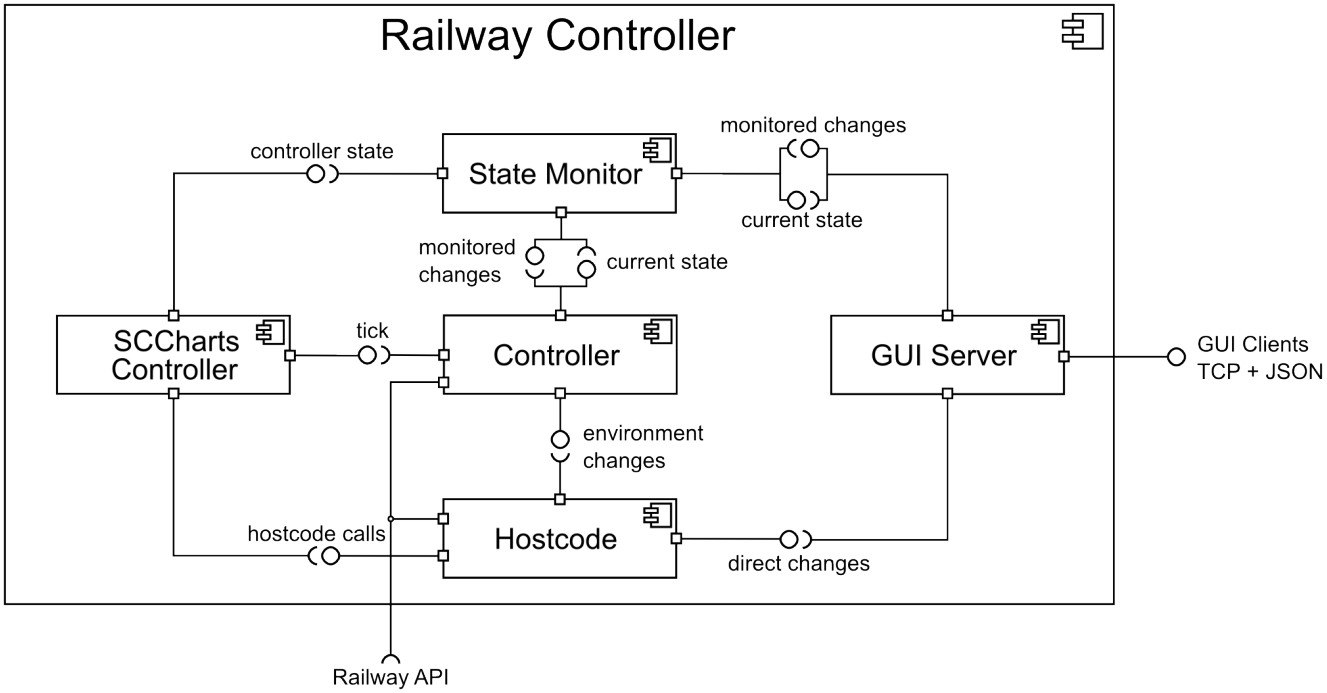
C Controller

Components

The component diagram below shows the architecture of generated and created Railway Controller.



SCCharts Contoroller

```
scccontroller.h <compiled-sccchart>.c
```

This component is the compiled Controller-SCChart.

The interface offers an the sequential synchronous tick function and additionally access to some internal variables to determine the internal state of the SCChart during execution. Furthermore the component uses the provided hostcode-interface to control the railway.

Hostcode

```
hostcode.h hostcode.c
```

This component provides an interface to affect the railway and request events.

The interface provides shorter and simpler access to most of the railway API. To prevent conflicts with existing function names all functions in our interface are prefixed with *rail*.

Most controls and requests are forwarded to the controller maintaining the synchronous environment of the tick-function. Some controls which do not require this maintenance are directly forwarded to the railway API.

The hostcode header contains the general documentation for any hostcode call used in the SCChart.

Controller

```
controller.h defaults.h controller.c
```

The core controller component sets up a connection to the railway, maintains an environment for the synchronous tick-function and invokes the tick-function in a loop.

Additionally some time tracking is performed to determine time consumption of the tick function and its environment.

To increase the robustness of the system the controller checks internal permissions of the SCChart-Controller against detected trains on the railway.

State Monitor

```
statemonitor.h statemonitor.c
```

The state monitor component allows a thread-safe communication between connected GUIs and the running controller. This communication is also tick-safe meaning that changes are only applied at tick-borders. In addition to that the communication is mutual excluded without blocking the controller thread and thus the controller remains a reactive system.

Furthermore it allows the generation of the current state of the controller in JSON format.

GUI Server

```
guiserver.h guiserver.c
```

The GUI server allows multiple GUI clients concurrently connecting to the controller. The number of available slots is defined in the guiserver header.

The communication is established via TCP and the message format is JSON. Each client can request the current status of the controller and interact with some commands changing most of the important properties of each train, especially the schedule. The detailed communication format and capabilities are documented in the TCP Communication section.

Most controls are performed delayed and thread-safe with the state monitor, some are directly performed via the hostcode component. Changes of the same property from different clients is processed with first-come-first-serve-principle which may cause lost update writes but will not affect the integrity of the controller.

Interaction

GUI

The controller allows multiple GUIs to connect and interact with the controller.

Interrupts

Quit, Pause and Cleanup

While running the console the following user inputs are provided by the controller:

- **CTRL+C** (SIGINT) quits the controller
- **CTRL+** (SIGQUIT) cleans up
- **CTRL+Z** (SIGSTP) pauses and continues