# Frequently Asked Questions (Chrona Validator)

**What is conceptually the Chrona Validator?**

A platform-aware software-in-the-loop simulation of a controller, in which in addition to the functionality the timing and platform behavior are considered. You can either do an open-loop or a closed-loop simulation. In an open-loop simulation no controlled entity model (= plant model) is necessary, only recorded sensor values or manual inputs. The Validator also allows a closed-loop simulation of controller and plant in the form of a co-simulation. In both open- and closed- loop simulations, the actual source code of the controller is executed.

**How does the Chrona Validator work?**

The Chrona Validator relies on a separate 'supervisory component' that is attached to the controller code. At certain spots during the code execution that 'supervisory component' gets the control and decides, for example, whether a task function needs to be interrupted. The 'supervisory component' is implemented as a discrete event simulation. In the Validator set-up callbacks to the 'supervisory component' are inserted at certain spots in the controller source code. Examples of such spots are operating system calls or accesses to global variables. The 'supervisory component' decides, based on a platform model, what to do. A more detailed the platform model results in a more accurate correspondence between simulated an actual target platform behavior.

**How much effort is it to prepare a system for a validator simulation?**

The preparation of the C code is highly automated: utility tools analyze the source code to find, for example, global variables and calls to operating system functions. At these spots, respectively, a return to the discrete event simulation of the Validator, the 'supervisory component' is inserted. The annotation of the source code with execution times is fully automated. The execution times can be determined by static code analysis tools or by measurements. In addition to the preparation of the controller source code the execution platform needs to be modeled. For this, the Chrona Validator provides the visual/interactive tool Ovid, together with libraries for common platforms, such as the operating system OSEK or the CAN bus.

**Which simulation environments and which programming languages are supported?**

The controller must be available as C code. The model of the controlled entity (= plant model) must be a MATLAB®/Simulink® model. The plant model is only necessary for a closed-loop simulation. If the controller or parts thereof have been developed in MATLAB®/Simulink®, C code needs to be generated from the models, for example, with the Embedded Coder® or Simulink Coder®.

**How do I model the execution platform?**

With a visual/interactive tool called Ovid. Typical platform aspects such as the operating system scheduling, interrupt handling or bus communication can be modeled by configuring pre-defined library blocks. For example, you change the frequency of a periodic function call by configuring that model component in Ovid.

**How do you obtain the execution times of code fragments?**

With analysis tools, or by measurements, for example, on an integrated circuit emulator.

**How accurate is a Validator simulation?**

The accuracy depends on the precision of the execution times and the granularity of the platform model. The platform model can be incremenatlly refined. For example, in the platform model the CAN communication with other control units can initially be neglected and modeled later.

**Does a Validator simulation always require a plant model?**

No, in the case of an open-loop simulation. Yes, in the case of a closed-loop simulation. In the current version, the Validator then needs a MATLAB®/Simulink® model.

**Can I use the Chrona Validator when I developed new controller functions in MATLAB®/Simulink®?**

Yes, but C code needs to be generated from the models, for example, with the Embedded Coder® or Simulink Coder®.

**Do I need the Validator even if I defined a "clean" scheduling?**

Yes. The validation by testing becomes only superfluous if a so-called correct-by-construction approach is used. For example Chrona's Timing Definition Language (TDL) guarantees that the modeled timing behavior is observed on a particular platform under all circumstances.

**Why do I need co-simulation? Is it not sufficient to put the C code of the controller in a MATLAB®/Simulink® S-Function?**

Experience has shown that existing C code can not simply be placed in a MATLAB®/Simulink® S-function, as it can not be compiled. The reason for this is that in MATLAB®/Simulink® the corresponding compiler options are not available.
Even if the compilation would be feasible, no time-accurate simulation of the overall system could be accomplished, as no 'supervisory component' is avaliable. Without that component and the corresponding timing information and platform model no realictic SIL simulation can be expected.

**What are the advantages of the Validator compared to other tools which provide similar objectives? What are the disadvantages?**

The advantages of Chronas Validator are significant and can be summarized by two characteristics: a much more accurate platform-aware SIL simulation, so that errors are actually found much earlier during testing, as well as a seamless integration with the popular development tools MATLAB®/Simulink® and Eclipse. The advantages can be summarized as follows:

1) the control flow must not be modeled separately because the actual C code is executed, thus the simulated behavior and the functional behavior on the target platform (= target) are equivalent

2) the validation is more effective because, in addition to 1) also
a) the execution times are defined at the granularity of basic blocks, whereas other tools allow only task-level granularity
b) because the Validator offers both an open-loop and a closed-loop simulation

3) the Validator is seamlessly integrated with MATLAB®/Simulink® and Eclipse.

At the moment the Validator does not support the import of UML diagrams with annotated timing behavior.
Chrona's Validator can be supplemented by tools that enable the management of test cases.

**Does Chrona's Validator support the fulfillment of ISO 26262?**

Yes, even optimally. The Chrona Validator provides the most effective validation of functionality and real-time behavior of all software-in-the-loop (SIL) tools on the market, thus contributing significantly to the fulfillment of the ISO 26262 standard.