SDL-RT white paper

Context

A real time / embedded software is basically a software nobody sees but it is the one that has most constraints in the software industry such as high-level of reliability, small memory footprint, and high performance. In the same time embedded software requirements are getting more and more complex because all devices should now be able to communicate with each other and also because hardware components prices are dropping allowing the usage of powerful processors for small systems. What used to be a few hundred bytes of code in assembly language is now hundreds of thousands of lines in C code. Therefore it is a priority for all real time software developers to make sure the software architecture is good, the interfaces with other modules is well defined, the code is legible and re-usable. A reduced time to market also brings the need to develop software before the hardware is done: simulating and prototyping the software is now a must.

See what is hidden

SDL-RT is a graphical representation that allows to see the key elements of the embedded software such as tasks, messages, timers and semaphores. It offers a way to graphically gather tasks in functions without direct impact on the final code to define the software architecture. At the same time it offers graphical representation of a task's internal states, message exchanges, semaphore manipulations and timers.

Keep the precision

When it comes to the detail of coding, graphical languages show their weaknesses. SDL-RT does not have that problem because it is based on C language. The developer keeps the preciseness of C embedded in a graphical representation. The balance between SDL-RT graphical representations and traditional textual C code is set by the user as it is not always wise to use graphical representation instead of text. Therefore integration with legacy code or third party modules is direct and traditional developers do not get lost with the language. That is a guarantee of continuity with existing development processes.

Re-use what has been done

The basic running element in a real time system is a task. Some complex functionalities need several tasks that can be gathered in SDL-RT functional blocks. SDL-RT is object oriented where a real time developer needs it to be: on blocks and tasks providing inheritance and specialization. Please note SDL-RT has the benefits of object orientation while basic instructions are still written in C.

Write it once

SDL-RT allows full C code generation reducing development time. Since basic instructions are written in C language, effective code generation is limited to support SDL-RT concepts such as finite state machines, timers, semaphores, and tasks manipulations. The generated code is therefore legible making debug on target easy at C level if no host connection is available.

Simulation and validation

SDL-RT has the property of a formal language: it is complete and non ambiguous. An SDL-RT design contains in itself all the necessary information to be executed on host or target. Executing the system on host allows early debug and connection to a graphical user interface for prototyping of the final software. It has been proven early test of the software increases quality and can reduce time to market up to 50%.

Dynamic view

Traditional code is a static representation of a dynamic behaviour. SDL-RT MSC representation is a dynamic view of the system that can be used as specification or as a trace of execution. The key elements are represented (tasks and semaphores) as well as the key events (task creations or deletions, semaphore manipulations, timers, message exchanges). This is a major breakthrough since an interface is not only a static set of messages or function calls but also a sequence of exchange.