SCARI Software Suite

A Complete Integrated Development Environment for SDR

OVERVIEW

The SCARI Software Suite is the most comprehensive Integrated Development Environment (IDE) for SDR on the market. Used internationally by platform and radio manufacturers as well as application providers, the Suite reduces development risk and the time-to-market, to obtain top-quality SDR products.

The SCARI Software Suite includes:

A. SCARI++ Core Framework (CF): The CF of choice for commercial off-the-shelf (COTS) platform providers.
B. SCA Architect: The platform-independent, Eclipse-based development environment that lets you create, validate, and debug your components and applications in a single project.
C. Component Development Libraries: This provides a set of generic functions that greatly simplifies code generation and significantly reduces memory requirements.
D. Radio Manager: A platform run-time monitoring tool to install and control applications, as well as visualize deployment strategies.

FEATURES

SCARI++ CORE FRAMEWORK	SCA ARCHITECT ™	COMPONENT DEVELOPMENT LIBRARIES	RADIO MANAGER
• Built for embedded platforms, but can also be used on development hosts • Comes with open source ORB • Ultra-fast and flexible	• Create and assemble your SCA components • Platform independent - Eclipse based • Real-time validation and re-factoring	• Generic component libraries • Reduces lines of code and memory footprint • Simplifies and speeds up development • Core framework independent	• Install, control and debug your applications • Introspect SCA radios • Graphically represents component deployment and their interconnections

A. SCARI++ CORE FRAMEWORK

Used for research projects, for development purposes or for actual fielded radios, SCARI++ is an SCA core framework you can always rely on.

Built from the ground up for embedded platforms, the SCARI++ Core Framework is a new generation core framework. Every design decision was made with real-time performance in mind.

With the lessons learned from the JTRS-certified SCARI-Open core framework, SCARI++ is available for an exceptional number of operating environments and has been used in several system-on-chips solutions.

It is the most widely adopted core framework, and is available pre-integrated with a number of generic SCA platform providers. SCARI++ is the cornerstone of CRC's SCA development solution.

Operating environments

Designed with portability in mind, the SCARI++ Core Framework has been shown to work on a wide number of operating environments. This version is intended for use on Linux/X86 systems with the TAO ORB.

High-speed core framework

The SCARI++ Core Framework provides a full implementation of all the SCA framework interfaces. It implements exceptional real-time features to minimize the boot time of an SCA system. It transparently transforms indirect connections into faster direct connections. It uses a unique caching system to avoid long delays required for finding deployed components.

The SCARI++ DeviceManager even allows SCA devices to be collocated into a single address space, which is significant for accelerating the boot sequence of a node. In fact, SCARI++ components can all be linked into a single address space to further accelerate the boot sequence.

Extended introspection features

SCARI++ supports unique introspection features used by CRC's Radio Manager monitoring tool. Using SCARI++, developers can obtain detailed deployment information, which allows them to determine where components have been deployed and how they have been interconnected.

Debugging support

SCARI++ is provided in two binary forms: Release and Tracer. The Release version is compact and ready for embedded deployment. The Tracer version is instrumented with debugging code that produces several levels of tracing messages that can be selectively turned on or off. SCARI++ can also save the logging messages produced to a file before a log service becomes available.

B. SCA ARCHITECT ™

The SCA Architect is the Integrated Development Environment (IDE) of the SCARI Software Suite. It covers the complete SDR development life cycle, from the creation and validation of components to their assembly into applications or nodes.

SCA Architect is provided as a plug-in for the universally adopted, platform-independent, Eclipse framework. Embedded system developers will benefit from a well-known interface, making it easy to migrate between the different development phases of their project.

Reusable modeling elements

There's no need to repeat every step when creating components. SCA Architect lets you create reusable component properties and ports, allowing you to assemble them into component types to create similar components much more quickly. This feature, unique to SCA Architect, is also extremely useful when applying design modifications to many components. A modification at one place is automatically reflected in every component of similar type – a precious time-saving feature.

Real-time model validation

The real-time model validation feature of SCA Architect eliminates time-consuming retrofits to correct early errors, greatly accelerating the creation of SCA components. CRC's in-depth experience and expertise with the SCA has provided Architect with the largest set of validation rules in the industry.

Reverse engineering and re-factoring

SCA Architect can also be used to model and validate pre-existing SCA domain profiles. Its powerful reverse engineering capabilities will provide a validation assessment of imported profiles. Architect's unique re-factoring feature can also automatically correct errors from a number of suggested fixes.

Unambiguous modeling

SCA Architect's superior modeling capabilities provide unambiguous graphical representations of assemblies, capturing the containment relationship between deployed components and their target – a key concept to enable the graphical representation of all types of indirect connections and host collocation relationships. Architect's modeling capabilities make it the only tool capable of providing a determinist graphical representation of assemblies.

Zero-merge code generation

SCA Architect generates fully functional POSIX C++ SCA components, to be built and used in applications without writing a single line of code. Being template based, SCA Architect can be tailored to support other programming or code conventions.

Architect breaks new ground by introducing “zero-merge” code generation capability. Developers specialize the behaviour of a component, rather than modify it. Developers can at last remodel existing components, and regenerate code without having to merge two versions of the source code.

Configuration management

SCA Architect pioneers model-level configuration management. Developers no longer have to manually track each individual artifact of a model element. It allows developers to save model elements directly to a repository. Developers don’t have to save incoherent versions of those models. This feature radically simplifies configuration management.

Shared projects

SCA Architect provides a way to reuse common modeling elements without having to duplicate them. SCA Architect supports the Eclipse concept of shared projects. Rather than duplicating a modeling element, developers can reference projects containing shared elements. After all, “reuse” is the SCA’s philosophy.

C. COMPONENT DEVELOPMENT LIBRARIES

The Component Development Libraries (CDL) accelerates the creation of SCA-compliant components. The CDL lowers the learning curve of the SCA and speeds up the development by providing generic SCA components, designed to shield developers from the intricacies of the SCA and CORBA.

Using the CDL-generic components drastically reduces the number of lines of code required to create components, simplifying the generated code and associated testing, thereby reducing the development cycles. Also available as shared libraries, the CDL enables the creation of very small footprint SCA components.

Generic log and event ports

The CDL provides generic implementations of a log port and an event channel port. These are used to report logging messages or to produce events that may be consumed by other components. It implements these through generic ports that transparently handle all the requirements of the SCA. Developers only need to worry about producing messages, not levels and connections. In fact, the CDL offers much more than the SCA-required behaviour, allowing components to save the logging messages produced before a log service becomes available. This unique feature provides precious debugging information that is otherwise lost.

Generic components

The CDL provides a generic implementation of the two types of components SCA developers must create: the resource and the device. CDL components will automatically generate the SCA-required log messages and events, preventing developers from making costly compliance mistakes. They also implement the SCA and CORBA life-cycle, avoiding other potential compliance issues.

The implementation of the state behaviour for an SCA device is notoriously difficult, creating one of the major sources of compliance issues. The CDL generic device takes that burden off the developer's shoulders by handling capacity allocation requests. This further contributes to the reduction in quantity of source code required for implementing a new SCA device. The CDL device automatically handles the three state machines required for all SCA devices, which when combined, lead to 25 states and close to 70 transitions.

Generic PropertySet

The CDL generic components provide a framework for implementing properties which shields developers from the SCA and CORBA. Using the CDL Generic PropertySet, configuration requests are transformed into simple invocations to C++ member functions. Component property values are kept as simple C/C++ native data types. The CDL Generic PropertySet implements all of the SCA required behavior for configuration requests, increasing SCA compliance.

Debugging support

The CDL is provided in two binary forms: Release and Tracer. The Release version is compact and ready for embedded deployment. The Tracer version is instrumented with debugging code that produces tracing messages of varying levels that can be selectively turned on or off.

D. RADIO MANAGER

The Radio Manager is the essential tool for platform integrators and application testers. Through a block diagram representation, the integrator and tester can quickly visualize the SCA platform composition and see where each application resource has been deployed. The Radio Manager can visually represent how the connections between the resources and devices have been established.

In real-time, the Radio Manager introspects the SCA platform and reports the status of the devices and resources, refreshing the block diagram if needed. A new waveform being added, a device failing, a connection broken – all will be shown, providing invaluable information to the integrators and testers to reduce debugging time.

The Radio Manager is to the SCA what a debugger is to source code. In fact, the Radio Manager can be connected to any embedded SCA platform, just like source code debuggers.

In addition, the Radio Manager can be used to serve many other purposes.

Installing and controlling applications

Built into the Radio Manager is an application installer used to upload the required application artifacts onto the platform. Once done, the Radio Manager can instantiate the application, start, configure and stop it, and finally terminate and remove the application.

The Radio Manager provides a generic properties browser that can render every type of SCA property and can change values dynamically or in batch mode.

Debugging features

The Radio Manager is a very useful tool to test performance of the platform and applications. It offers full control over the deployed components. It can therefore be used to modify the values of the application parameters when testing the application under different conditions; it can shut down a complete node while applications are running to analyze how the platform reacts in specific scenarios.

Extended introspection features

The Radio Manager provides two different views for displaying the deployed software components. The hierarchical view uses a tree-like structure, where each node represents a deployed component. The block diagram view uses a block for each deployed component.

When connected to the SCARI++ core framework, the Radio Manager can even show which components have been deployed onto other components, and it graphically displays how components have been inter-connected.

Specialized HCIs

The Radio Manager allows the system integrator to launch its own specialized Human Control Interface (HCI) to control the application. This becomes particularly interesting when specific APIs are required to control the radio.

Using SCA Architect, an SCA application can be packaged with a specialized HCI that will be installed with the application on the target SCA platform. After instantiation of the application, the Radio Manager searches for the existence of the specialized HCI, downloads it, and launches it.