

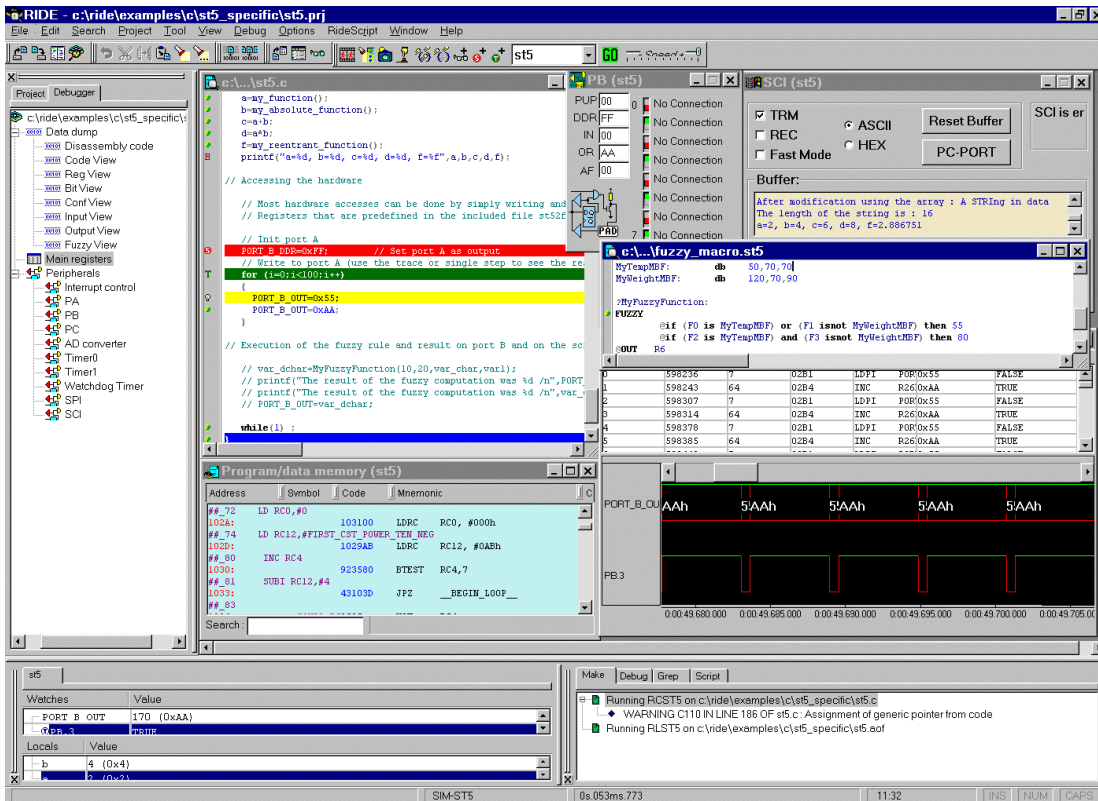
Rkit-ST5 is a complete toolchain for the ST5 family of micros. It includes an optimizing ANSI C Compiler, a Macro-Assembler, a Linker, an accurate Simulator and CodeCompressor post-linker technology, all integrated into the same Development Environment.

Rkit-ST5



www.raisonance.com

RAISONANCE



Integrated into RIDE

Rkit-ST5 is delivered with Raisonance Integrated Development Environment (RIDE), a fully integrated IDE featuring color syntax highlighting editor, multifile search, project manager (with global and local options), on-line help and full control over all the tools of the toolchain.

From editing to compiling, linking and debugging (with the simulator, or real time emulator), RIDE increases productivity by putting all the tools at your fingertips into a coherent and easy to use user interface.

When it comes to debugging, RIDE provides a rich variety of views into your application (Main registers, Peripherals, Symbols, Stack...) and a graphical trace view, as well as the possibility to define waveforms to be used as stimuli during the simulation.

For the most complex projects RIDE allows scripting, multi processor simulation and user-defined peripherals / hardware.

Command-line Version

All tools accessible via RIDE are also available as 32-bit executables that can be integrated in the IDE or Editor of your choice. Tool options can be set via dialog boxes into RIDE, on the command line or directly into the source file with pragmas.

Flexible licensing

Rkit-ST5 comes in a variety of licensing options to fit all needs: from the FREE, limited demo version, to the enterprise suite, including CodeCompressor, Scripting and Multiprocessor Simulation, there is an option for every need and budget. For even more flexibility, Rkit-ST5 offers the choice of Software (serial number) or Hardware (dongle) protection.

Optimizing ANSI-C Compiler

A super-set of ANSI-C

RC-ST5 implements the ANSI standards for the C language, expanded with ST5 specific keywords that allows to access the ST5 hardware (interrupts, memory spaces) efficiently and directly from the C language.

Memory Models

Two different memory models are implemented to provide a code generation appropriate to the size of your application. Choosing the smaller memory model your application fits in means making sure that no hardware resources are wasted.

Technically, the memory model specifies default location for variable declarations as follows:

Model	Default memory
SMALL	Registers
LARGE	Data

Data Types

RCST5 supports all common data types needed for embedded applications:

- Bit type: single bit.

- Integer Types: 8 bits : "signed char" and "unsigned char",
16 bits : "signed int" and "unsigned int",
32 bits : "signed long" and "unsigned long",

- Real Types: 32 bits : "float" (IEEE754 single precision),

All types requiring more than 1 byte are stored in memory according to the "Big Endian" convention (Most Significant Byte first, that is, at the lowest address)

Pointers Types

Both Generic and Space qualified pointers are supported. Generic pointers are the default options because they can point anywhere; with them you don't need to worry about possible problems in pointer assignments and conversions. Space qualified pointers must be managed more carefully, but they produce smaller code.

Code Optimization

RC-ST5 produces very compact code by using optimization techniques such as Constant Propagation, Jump Optimization, Dead Code Removal and use of registers for variables and parameters passing. Whenever possible the code is optimized to be both compact and fast to execute; in some cases you can choose your priority by optimizing for SIZE or for SPEED.

Libraries

RC-ST5 is supplied with ANSI C standard libraries as described in: math.h, stdio.h, string.h, ctype.h, and stdlib.h header files. All libraries are written in Assembler for the best efficiency and are memory-model specific in some cases.

Macro Assembler

A powerful macro assembler is provided together with the compiler. Assembler programmers can take advantage of absolute and relocatable modules, and the linker can mix up modules coming from the assembler and the compiler. The macro assembler understands both ST VisualFive former syntax and Raisonance new, richer syntax, where addressing modes and operands can be specified either in the opcode or in the operands.

Decision Processor Unit Support

The Decision Processor Unit feature of the ST5 (which is basically a coprocessor that allows to apply fuzzy algorithms and rules to control situations that are hardly described in procedural language) is fully supported throughout the toolchain. The compiler and assembler understand a special syntax used to declare and apply fuzzy rules; such syntax can be either typed in or taken from a visual modelization tool provided by ST. The simulator will simulate the internals of the DPU and show some of its internal registers during execution of the fuzzy rules.

Supported derivatives

Rkit-ST5 supports all derivatives of the ST5-508 family. For every derivative a specific header file is provided that defines the Control Registers for that derivative. The toolchain supports specific features of some derivatives, like Flash memory. Specific simulators are provided for most devices, although, for some of them, some of the most complex peripherals are not simulated (typically I2C and CAN).

SIMICE-ST5 Simulator

SIMICE-ST5 simulates in detail the ST5 core and memory spaces, as well as most peripherals, including timer, UART, IO Ports, ADC, watchdog and more. Used with RIDE, SimiceST5 is a powerful debugger and offers a variety of views into your application (Inspect, Watches, Graphical Trace, Memory spaces) and effective debugging tools, such as complex breakpoints (including read/write breakpoints on memory locations), trace with timestamp, Code Coverage and stack consumption analysis.

SimiceST5 can run multiprocessor simulations and offers the possibility to define stimuli to be associated to port pins or to Control Registers directly.

CodeCompressor



CodeCompressor is a new, post-linker optimizing technology from Raisonance. By applying known optimizing techniques such as inlining, factorization and peephole, **after the linker** (and therefore

on the whole application, including libraries, assembler files and cross-module function calls), CodeCompressor achieves a code size saving of 5% to 20% and more depending on the application. CodeCompressor allows detailed control of the optimization steps; through the use of pragma statements in the source code you can define what sections of your applications are NOT to be optimized, thus giving the programmer absolute control at all times.

Supported Hardware

RKit-ST5 is able to drive most existing hardware tools for the architecture (emulators, starter kits, development boards) directly from RIDE, thus offering the user the same, familiar, user interface as the simulator. Raisonance policy of readily making available RIDE's Application Program Interface to third party ensures that new hardware tools can be easily supported.

RAISONANCE

17, av. Jean Kuntzmann
38330 MONTBONNOT
FRANCE
Tel : (+33) 4 76 61 02 30

PO Box 1784
Addison, TX 75001-1784
USA
Tel : +1 877-315-0792
Fax : +1 972-818-0996

Email: info@raisonance.com

Email: info@amrai.com

Local distributor :