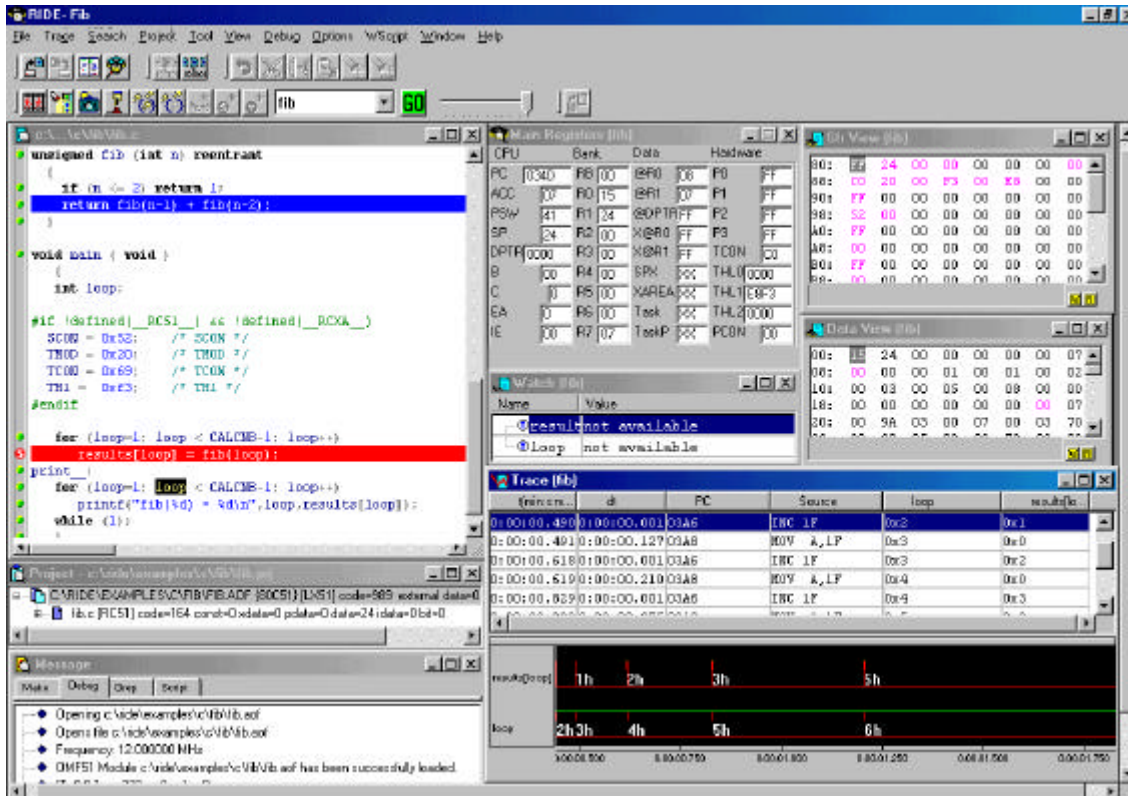


Rkit-51

Rkit-51 is a complete toolchain for the 80C51 family of micros. It includes an optimizing ANSI C Compiler, a Macro-Assembler, a Linker, a Real Time OS, a ROM Monitor, an accurate Simulator and CodeCompressor post-linker technology, all integrated into the same Development Environment. Rkit-51 supports >64k derivatives, including the MX family.



www.raisonance.com



Integrated into RIDE

RKit51 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, ROM monitor 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

Rkit51 comes in a variety of licensing options to fit all needs: from the FREE, 4kb-limited demo version, to the enterprise suite, including CodeCompressor, Kernel, PDK and Multiprocessor Simulation, there is an option for every need and budget. For even more flexibility, Rkit51 offers the choice of Software (serial number) or Hardware (dongle) protection.

RAISONANCE



Optimizing ANSI-C Compiler

A super-set of ANSI-C

RC-51 implements the ANSI standards for the C language, expanded with 8051 specific keywords:

alien	asm	at	bdata
bit	code	data	generic
idata	interrupt	intrinsic	pdata
sbit	sfr	using	xdata

RC-51 is compatible with most existing 8051 applications written in C.

Memory Models

Five 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, the default type for generic pointers and the type of CALL/JMP that are generated.

Model	Default memory	xdata	stack	Call/Jmp
TINY	data	No	idata/pdata	acall/ajmp
SMALL	data	No	idata/pdata	lcall/ljmp
COMPACT	pdata	yes	idata/pdata	lcall/ljmp
LARGE	xdata	yes	idata/pdata	lcall/ljmp
HUGE	xdata	yes	pdata	lcall/ljmp

Data Types

RC51 supports lots of different types, sometimes in both little-endian and big-endian modes

- 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),
BCD float : 32, 48 and 56-bit words.

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-51 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-51 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.

MONITOR-51 ROM Monitor

RKit51 is supplied with a ROM Monitor, directly driven by RIDE via the same user interface as the simulator. The ROM Monitor allows simple and inexpensive debugging with Raisonance or Third Party boards.

Local distributor :

A wide range of supported derivatives

Rkit51 supports a wide range of 8051 derivatives from companies like Philips, Atmel, Infineon, Cygnal, Dallas, Analog Devices and more. For most devices a specific header files are provided that defines the special function registers for that derivative. These headers files are simple to create so additional derivatives can be supported very easily. The compiler supports specific features of some derivatives, like Dual Data Pointers and serial ports. Specific simulators are provided for some devices, but all devices can be simulated by the "default 8052" for the core and the most common peripherals.

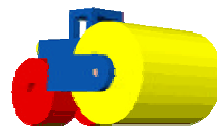
The Philips 8051MX family has a dedicated set of tools that is actually a dedicated toolchain and is included in Rkit51. The Compiler, Assembler, Linker and Simulator are all done in such a way that the new features of the MX such as linear addressing mode (up to 8Mb of code and Data), 16 bit stack and Universal Pointers are available to the user.

SIMICE-51 Simulator

SIMICE-51 simulates in detail the 8051 core and memory spaces, as well as most peripherals of the most common derivatives (including timer, UART, IO Ports, ADC, watchdog and more). Used with RIDE, Simice51 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 and Code Coverage.

Simice51 can run multiprocessor simulations and offers the possibility to define stimuli to be associated to port pins or to SFRs 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 is a fully automatic tool that should be used when your code is already debugged, but it also provides a manual interface to see the action of each of its optimization passes and the possibility for a limited debugging after the compression. CodeCompressor also allows specific control of the optimization steps through the use of pragma statement in the source code effectively defining what sections of your applications are NOT to be optimized. Giving the programmer absolute control at all times.

KR-51 Real Time Kernel

KR-51 offers a variety of services, providing time and memory management, semaphores, synchronizing events, inter-task communications, and more sophisticated functions such as a perpetual calendar. KR-51 is fully integrated into RIDE and debugging is kernel-aware. Source code of KR51 is provided with the most complete version of Rkit51 (RkitE51).

RAISONANCE

755, av. A. Croizat

38920 CROLLES

FRANCE

Tel : (+33) 4 76 08 18 16

Fax : (+33) 4 76 08 09 97

Email: info@raisonance.com

4120 Firebrick Lane

Dallas, TX 75001-5008

USA

Tel : +1 877-315-0792

Fax : +1 972-380-1436

Email: info@amrai.com