

**RAISONANCE**

a KEOLABS brand



# **Raisonance tools for C816 family**

**Getting Started**

**Document version**  
25 April 2012

# Contents

1. INTRODUCTION.....	4
1.1 Purpose of this manual.....	4
1.2 Scope of this manual.....	4
1.3 Additional help or information.....	4
1.1 Raisonance brand microcontroller application development tools.....	4
2. RAISONANCE TOOLS FOR THE C816 FAMILY.....	5
2.1 Ride7 and RFlasher7.....	5
2.2 RKit-C816.....	6
2.2.1 Simulator (SIM-C816).....	6
2.2.2 Licenses.....	6
2.2.3 Supported derivatives.....	6
2.3 RLink.....	6
3. SETTING UP THE RAISONANCE TOOLS FOR C816.....	7
3.1 Downloading the software.....	7
3.2 Installing the software.....	9
3.3 Registering the software.....	9
3.3.1 Registering using a Serial Key.....	9
3.3.2 Registering using a hardware dongle.....	11
3.3.3 Registering using a Serial Number or old dongle (before 2011).....	11
4. CREATING A PROJECT USING THE GNU TOOLCHAIN.....	12
4.1 Using an example project.....	12
4.2 Creating a new project.....	13
4.3 Configuring the GNU GCC toolchain.....	14
4.3.1 Compiler and Assembler options.....	14

4.3.2 LD linker options.....	14
4.3.3 General.....	14
4.4 Using the new version of GCC: 4.4.....	15
4.4.1 Why use GCC 4.4?.....	15
4.4.2 Switching to GCC 4.4.....	15
4.4.3 Activating dead code removal.....	17
4.5 Using the old version of GCC: 3.2.....	18
4.5.1 Switching back to GCC 3.2.....	18
4.5.2 Why switch back to GCC 3.2?.....	18
<b>5. DEBUGGING WITH THE SIMULATOR.....</b>	<b>19</b>
5.1 Configuring the simulator options.....	20
5.1.1 Miscellaneous.....	20
5.1.2 XTAL Oscillator.....	20
5.2 Launching the simulator.....	20
5.3 Using the simulator.....	22
5.3.1 Viewing a peripheral.....	23
5.3.2 Viewing the stack.....	23
5.3.3 Viewing data.....	24
5.3.4 Using breakpoints.....	25
<b>6. DEBUGGING WITH HARDWARE TOOLS.....</b>	<b>26</b>
<b>7. CONFORMITY.....</b>	<b>27</b>
<b>8. GLOSSARY.....</b>	<b>28</b>
<b>9. INDEX.....</b>	<b>29</b>
<b>10. HISTORY.....</b>	<b>30</b>

## 1. Introduction

The Raisonance CoolRISC 816 development kit, RKit-C816, is a complete solution for creating software for the CoolRISC 816 family of microcontrollers. The development kit includes many tools that allow simple and highly complex projects to be developed with relative ease.

The new Ride7 and RKit-C816 **must** be registered to operate correctly.

### 1.1 Purpose of this manual

This manual introduces the first time user to the Raisonance development kit and guides them through many of the features, so they can quickly understand the tools, how they interact with each other and can start developing their own projects. This manual does not cover the features in great detail or advanced features, but provides a familiarity that will be a good basis for using more complex features.

### 1.2 Scope of this manual

This guide can be used by anyone interested in Ride7 for C816. It describes how to get started using Ride7 and RKit-C816 to compile and debug your application or a sample application. It assumes that you have the prerequisite knowledge of C and C816 assembly languages. The tools specifically addressed in this manual include:

- GCC C compiler toolchain
- Ride7 development environment
- RFlasher7 programming interface
- RLink debugger programmer

### 1.3 Additional help or information

If you want additional help or information, if you find any errors or omissions, or if you have suggestions for improving this manual, go to the KEOLABS' site for Raisonance microcontroller development tools [www.raisonance.com](http://www.raisonance.com), or contact the microcontroller support team.

Microcontroller website: [www.raisonance.com](http://www.raisonance.com)

Support extranet site: [support-raisonance.com](http://support-raisonance.com) (software updates, registration, bugs database, etc.)

Support Forum: [forum.raisonance.com/index.php](http://forum.raisonance.com/index.php)

Support Email: [support@raisonance.com](mailto:support@raisonance.com)

For information and support about XE8000 and SXxxxx chips, contact SEMTECH™ (formerly XEMICS SA): <http://www.semtech.com>

For information and support about EMxxxx chips, contact EM Microelectronic-Marin Ltd.: <http://www.emmicroelectronic.com>

### 1.1 Raisonance brand microcontroller application development tools

January 1, 2012, Raisonance became the brand under which the company KEOLABS sells its microcontroller hardware and software application development tools.

All Raisonance branded products regardless of their date of purchase or distribution are licensed to users, supported and maintained by KEOLABS in accordance with the companies' standard licensing maintenance and support agreements for its microcontroller application development tools. For information about these standard agreements, go to:

Support and Maintenance Agreement: <http://www.raisonance.com/warranty.html>

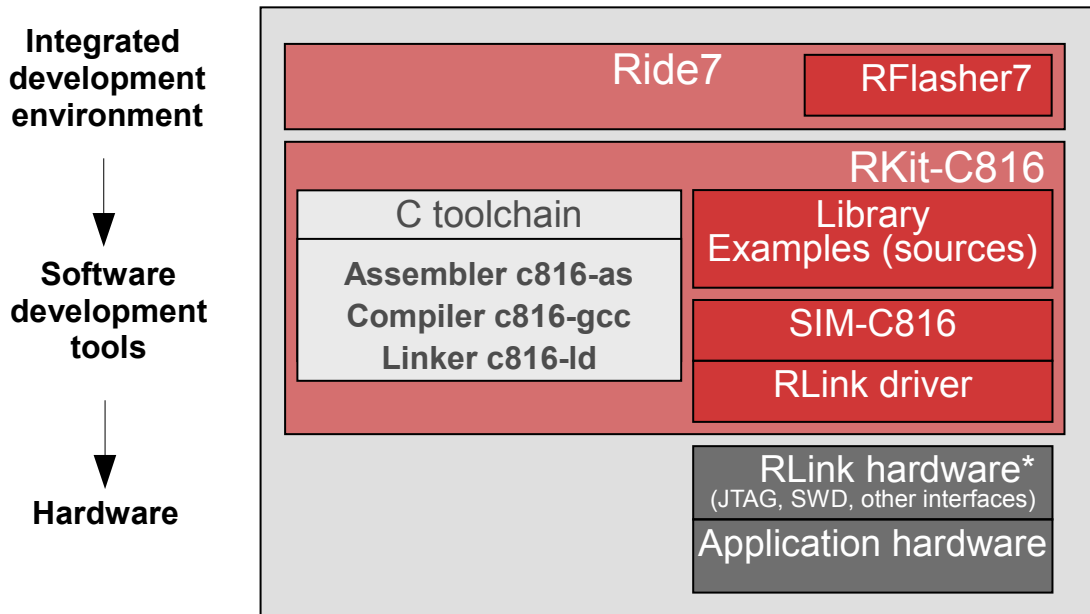
End User License Agreement: <http://www.raisonance.com/software-license.html>

## 2. Raisonance tools for the C816 family

Raisonance's integrated development environment, Ride7 for C816, interfaces with a range of development tools including:

- Editor, debugger and project manager and which integrates the GNU GCC toolchain.
- RKit-C816
- Software-based simulator SIM-C816
- RLink
- Debugging interface to use with the Raisonance SIM-C816 simulator, or to connect to your C816 CPU for programming and debugging with hardware debuggers

Each tool mentioned above has a dedicated user manual that you can refer to for more details. Documentation for Ride7, SIM-C816 simulator and RLink is available on-line from the user interface.



\* Purchased from EM Microelectronic or SEMTECH.

### 2.1 Ride7 and RFlasher7

Ride7 acts as the user interface for all the other tools. It gives you start-to-finish control of application development including code editing, compilation, optimization and debugging.

The associated RFlasher7 interface can program the Flash memory of the target MCU.

**Note:** RFlasher7 does NOT work for C816 devices. Do not try to start it, as it will fail.

## 2.2 RKit-C816

The RKit-C816 has 2 license options:

- **Lite** license includes default Raisonance hardware tools, GCC toolchain, and limited software features.
- **Enterprise** license includes script-based controls, integrated CVS and SVN version management, auto C formatter, integrated calculator, comment stripper and Unix end line converter.

RKit-C816 defines the devices, script tools, and debugging and programming interface features. Some of its tools have been developed by CSEM from a GNU core. They are now maintained and supported by Raisonance:

- The assembler c816-as takes source files written in assembler and generates object files. Controls are included to enable features of the microcontroller to be used or controlled.
- The C compiler c816-gcc is an ANSI compliant compiler that takes source files and generates object files. Extensions to the C language are used to enable features of the microcontroller to be used or controlled.
- The linker c816-ld combines the object files generated by the compiler and assembler and produces a binary file ready to be downloaded in the microcontroller. The linker decides where the different blocks of data and code are located in memory.

### 2.2.1 Simulator (SIM-C816)

Simulates the core (including the entire memory space) and most peripherals. Complex peripherals (USB, CAN) and some less common peripherals are not simulated. The same user interface is used for the simulator and the hardware debugging tools (RLink).

### 2.2.2 Licenses

Any files not explicitly mentioned here (below) are under license by Raisonance or other companies. They should not be copied or distributed without written agreement from their owners.

- The GNU toolchain is under the GPL license, which makes it free to use without royalties, as are some of the files written by Raisonance and its partners (EM, Semtech). You can assume that everything under the C816-gcc and GNU subdirectories of the Ride7 installation folder can be freely used, copied and distributed, but is without any warranty and only limited support from Raisonance.

### 2.2.3 Supported derivatives

Most of the existing C816 derivatives from SEMTECH (formerly XEMICS SA) and EM Microelectronic are supported. The up-to-date list of supported derivatives and the software simulation limitations can be seen in the **Target Options** in Ride7.

## 2.3 RLink

RLink is a standard debugger with a USB interface that allows you to program some C816 devices on an application board and debug the application while it runs on the target. It uses one of the standard protocols JTAG, GASP or DoCSPI. For more information, refer to the suppliers documentation (EM Microelectronic-Marin or SEMTECH) and the Getting Started manual specific to your target microcontroller.

**Note:** RLinks have different capabilities for programming and debugging microcontrollers. Your RLink's capability to program and debug any Ride7-supported target microcontroller is shown when Ride7 reads your RLink's serial number.

### 3. Setting up the Raisonance tools for C816

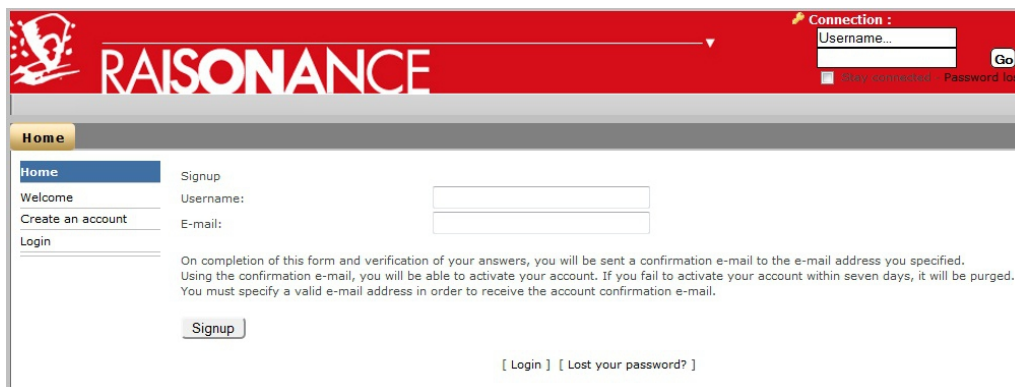
Before using any of the Raisonance tools for C816, the Ride7 and RKit-C816 software must be installed and registered.

#### 3.1 Downloading the software

The most recent Ride7 and RKit-C816 can be downloaded from the Raisonance extranet website: <http://support-raisonance.com/extranet/home/>



First you have to create an account, which is free and only requires a valid email.





Then go to the Download/MCU/C816 section, read the up-to-date information there (more recent than this document) and download the two files.

**RKit-C816**

RKit-C816, is the software package that defines the specific features of the Ride7 integrated development environment to include: GCC compiler toolchain and libraries for CoolRISC core, supported target microcontrollers, available hardware tools and Ride7 advanced features. The latest RKit-C816 release provides support for components from Semtech (includes former Xemics devices) and EM Microelectronic:

**Important Notes about Software Download & Registration**

**Installing the software**

1. Download both **Ride7** and the **RKit-C816** installation below.
2. Install Ride7 and then the RKit

**Registering the software**

**Software must be registered.** If not registered, it can only be used for 7 days. After the 7-day evaluation, the software can not be used in evaluation mode any more. During the 7-day evaluation users have access to full features of the Enterprise version.

Use the "RKit-C816 serial key generator" form below to obtain a **free** RKit-C816-Lite serial key. It allows activation of the software to compile without limitation and debug up to 2 Kbytes instructions of code.

To register the software, the PC you install it on must be connected to the internet. Launch Ride7 and select **Help > License**. In this interface you can register by providing either

- **Software serial key** for the RKit-C816 Lite (obtained using the form below)
- **Software serial key** for unlimited RKit-C816 Enterprise (provided at time of purchase)
- **Hardware dongle number** (purchasers of RKit-C816-Enterprise or RKit-C816-Lite tool sets)

To use CodeCompressor, purchase the RKit-C816 Enterprise (Hardware Dongle or Serial Key).

For unlimited debugging, purchase either the RKit-C816 Enterprise (Hardware Dongle or Serial Key) or a PRO version of your debugger (if available, depending on the target device).

If you have installed this software by accident and need to return to a previous version to continue working, you can download the old (obsolete) versions from the table below.  
(TODO)

**RKit-C816 serial key generator**

Company

Username \*   
(Give your extranet username or define it)

Firstname  Lastname

e-mail \*

Confirm your e-mail \*

**Last release**

<b>RKit-C816 1.08.12.0006</b>	2011-10-24 15:53:58	
<b>Ride 7.36.11.0280</b>	2011-11-04 09:54:59	Raisonance Integrated Debugger Environment

Two green download arrows are circled in red on the right side of the table.

Then you must install and register the software as explained in the following sections.



### 3.2 Installing the software

Perform these steps to install your new software:

1. Remove old versions of Ride7 and RKit (if any).
2. Install the new Ride7 software, then the RKit-C816 software, then validate its operation (test compilation, connection to RLink and to target CPU, etc.).

**Note:** Some components of RKit-C816 (GCC, ...) will not work correctly if installed in a path containing spaces or other complex characters. Be careful when using Windows Vista and Windows 7, especially the 64-bit versions. These versions use a default install directory named "Program Files (x86)", which contains parentheses and will make the tools fail. To be sure that your tools work correctly we recommend you install the tools in a directory with a short and simple name, such as "c:\Raisonance". In the rest of this document, the installation directory of Ride7 will be referenced as "<RideInstallDir>".

3. Launch **Ride7** and select **Help > License**. Register the software as explained in the following section.

### 3.3 Registering the software

The Ride7 and RKit-C816 software must be registered to operate correctly. Registration requires a Raisonance hardware product or software product license that is under a valid support contract (a standard support contract expires one year after the date of purchase).

To activate the software you must register your RKit-C816 using your RKit-C816 Serial Key or Dongle. Unregistered software functions for a 7-day evaluation period with full features of the Enterprise version. After 7 days the software can no longer be used. If this occurs, contact [info@raisonance.com](mailto:info@raisonance.com).

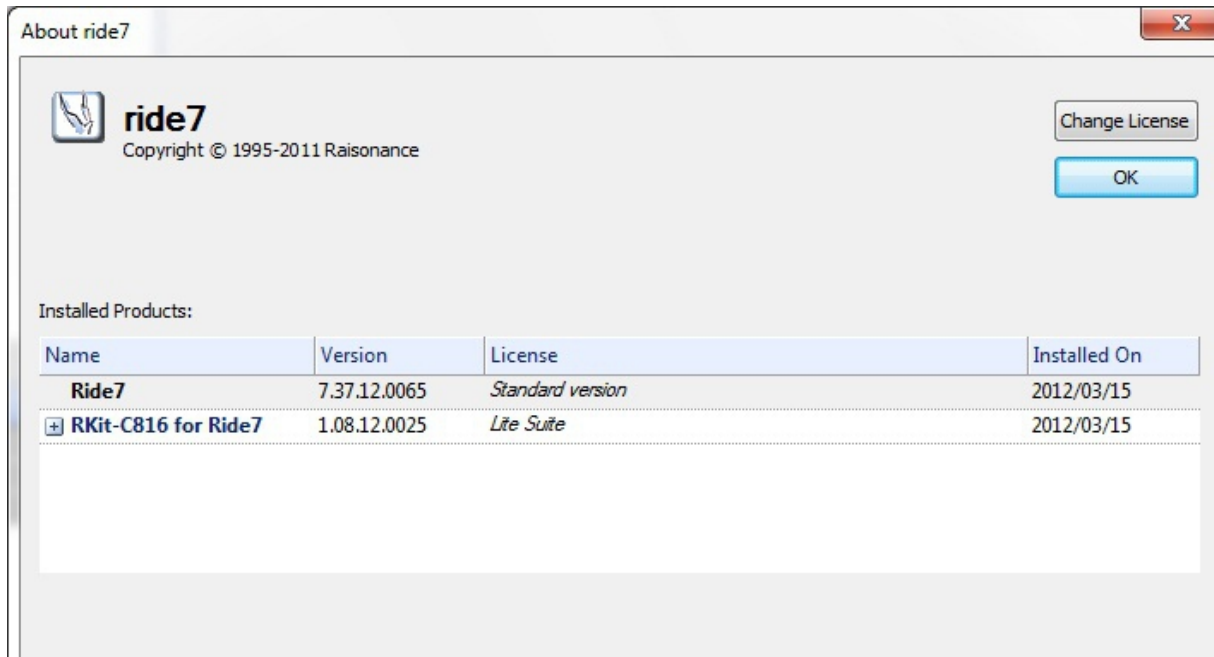
**Note:** RLink activation cannot be used for RKit-C816. Ride7 shows it because it can be used for other RKit (ARM, PPC), but for RKit-C816 you must use one of the other procedures described below...

#### 3.3.1 Registering using a Serial Key

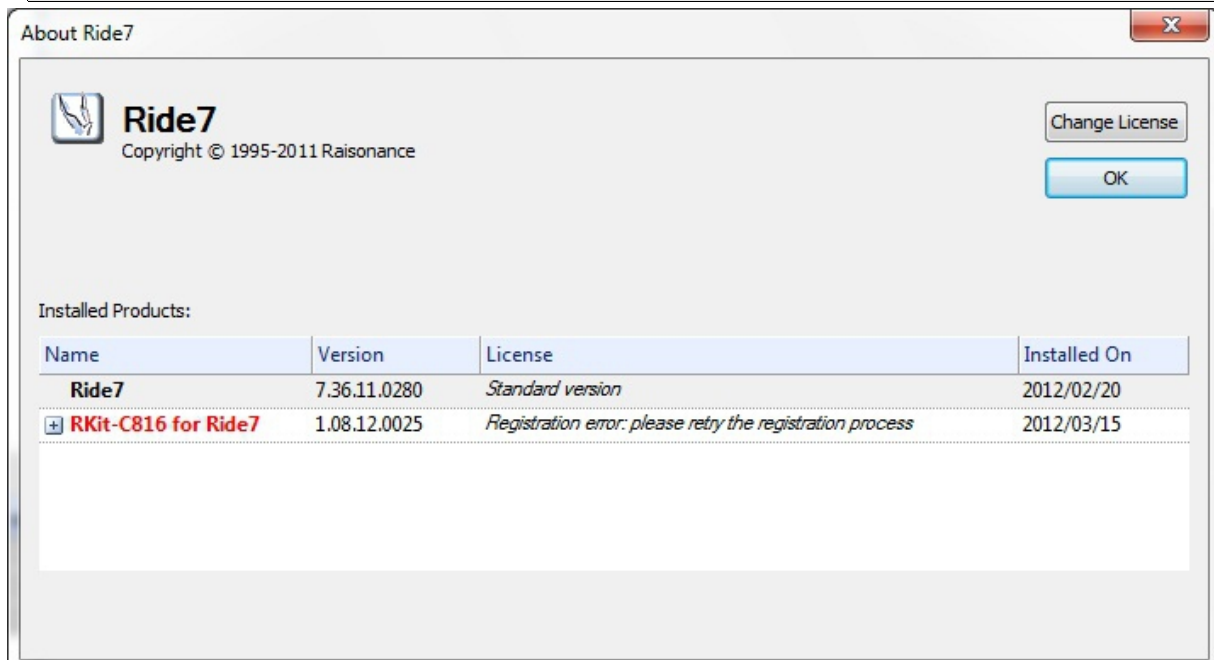
Use this procedure to register using a Serial Key.

1. Obtain a Serial Number using one of the options below:
  - a. Get a free RKit-C816 Lite Serial Key from the Raisonance extranet website (<http://support-raisonance.com/extranet/home/>), in the download section for RKit-C816.
  - b. Purchase an RKit-C816 Enterprise Serial Key (<http://www.raisonance.com/>).
  - c. Obtain a Serial Key corresponding to one of your old products (see section 3.3.3 Register using a Serial Number or old dongle (before 2011)).
2. Log on as admin, ensure you have internet access, a working default browser and a working email address.
3. Open Ride7. If it does not open automatically, click **Help->About Ride7**.
4. Click the **Change License** button. Select the **Serial Activation** method, click **Next**.
5. Enter the **Serial Key** provided to you. Click **Next**.
6. Connect to the internet and click **Get Activation code online** to open your browser on the Raisonance server.
7. Fill in the form and click **Generate and Send Activation code**.
8. Check your server for an email from Raisonance Support team with the **Activation Code**.
9. Copy the **Activation Code** from the email to the Ride7 window. Click **Finish**.
10. Close Ride7 and open it again. Check in the **About Ride7** window that you are in "Lite Suite" for RKit-C816 and "Standard version" for Ride7. Otherwise please contact our technical support ([support@raisonance.com](mailto:support@raisonance.com)).

This is what the About Ride7 window should look like:



**Note:** Version 7.36 of the Ride7 kit (the official version at the time this note was written), does not display the registration status of the C816 kit correctly. If it says "Registration error..." in the Help/About box, but you are still able to use the software, then in fact you are correctly registered. It is just a bug in the display of the status, and does not prevent you from using the tools. It will be corrected in versions 7.37 (currently being validated) and later.



You can ignore this error message if you are using Ride7 version 7.36 (or older) AND if you see the OK button. If the button is "Exit" instead of OK, then the registration really did fail.

### 3.3.2 Registering using a hardware dongle

Use this procedure if you purchased a new hardware dongle (2011 or later).

1. Plug in the dongle and make sure the LED lights up (otherwise you must install the driver).
2. Open Ride7 or RFlasher7. If it does not open automatically click **Help->About Ride7**.
3. Click the **Change License** button. Select **Dongle Activation**, click **Next**.
4. Close Ride7 and open it again. Check in the **About Ride7** window that you are in "Lite Suite" for RKit-C816 and "Standard version" for Ride7, otherwise contact our technical support (support@raisonance.com).

### 3.3.3 Registering using a Serial Number or old dongle (before 2011)

You must use this procedure for RKit-C816-Enterprise software licenses (and old hardware dongles).

If your product was purchased:

- less than one year ago, you will need to provide your proof of purchase.
- more than one year ago, you will need to purchase the (annual) support extension.

To register using a Serial Number, first contact Raisonance (info@raisonance.com) to obtain a **Serial Key**. They will need your **Serial Number**. Then follow the Register Using a Serial Key procedure described previously.

## 4. Creating a project using the GNU toolchain

Assembly and C applications can be written using the free GNU toolchain. This chapter gives an overview of how to create a C816 project with Ride7.

**Notes for users of other GCC toolchains:** You may have to modify the `GCC_EXEC_PREFIX` environment variable to rectify compatibility issues between different GCC toolchains. If you have this kind of problem, look at the GCC documentation to see the usage of this variable.

### 4.1 Using an example project

Ride7 for C816 provides several example projects that are ready to run. One of them is very simple and you should look at it first. To open this project, use **Project > Open Project** in Ride7 and select the `.prj` file depending on your target. For standard installations of Ride7, this file is found in `<Ride>\Examples\c816`. The example is described in its source file comments.

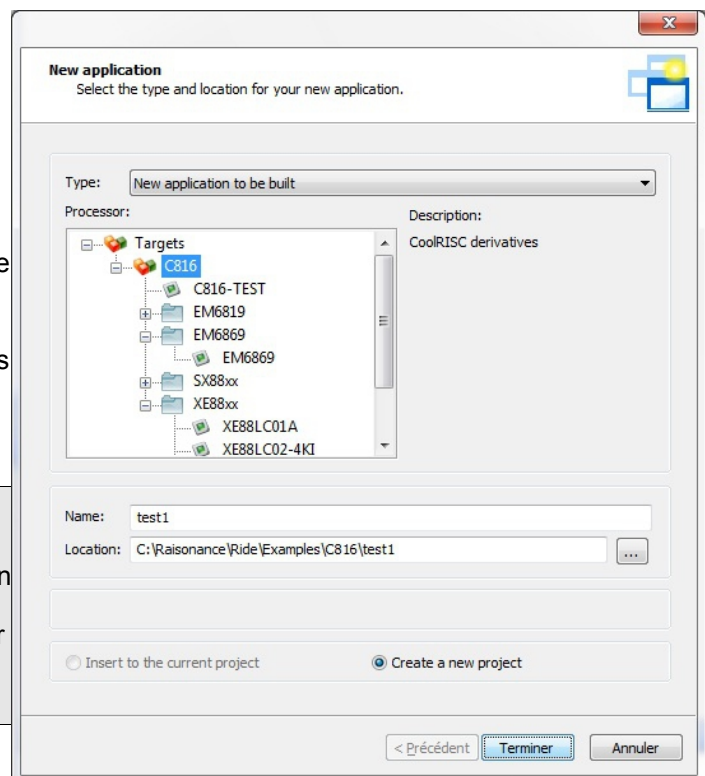
You can also look at the other examples which are run by selecting a virtual target C816-TEST whose properties can be modified. These projects should be used with caution with other targets. These examples are located in the directory: `<RideInstallDir>\Raisonance\Ride\Examples\C816\C`.

**Note:** Opening a project located on a network resource will fail in some situations. Among other things, it cannot work with pure network references. (`\\myserver\...`) It will also fail if the project is located in a compacted or crypted folder or in most similar "virtual folders". Therefore it is recommended to work with projects located on a local hard drive. It should work on Windows-mapped network drives and flash drives, but performance will be significantly impacted so even this is not recommended.

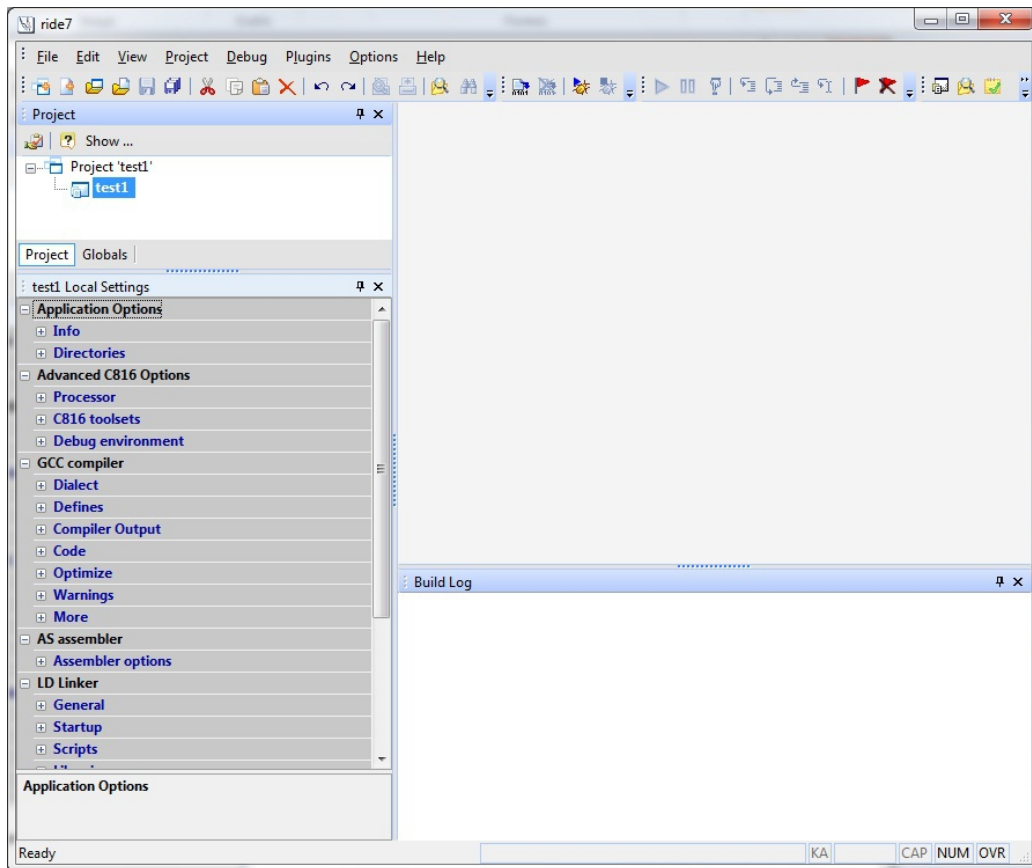
### 4.2 Creating a new project

1. In Ride7, go to the menu **Project > New Project**.
2. Select **New application to be built**, then your target **processor**.
3. Choose the application **Name** and the path to the main application folder.
4. Click on **Finish** to generate your application. Your application project is now created.

**Note:** The derivative called "C816-TEST" allows you to use a virtual device for which you can define space sizes and addresses in the "Advanced C816 Options". If you select this derivative, you must use a custom linker script. See the section describing the linker options for more information.



The Ride7 environment should look like this:



### 4.3 Configuring the GNU GCC toolchain

Ride7 Options Manager wraps the most important options needed to configure the GCC toolchain. The GNU options are split into three sections: GCC compiler, AS assembler and LD linker. Only the sections that apply to the selected project node and its children are displayed. For example:

- If a C source file is selected, only the GCC compiler section is visible
- If the application or project node is selected, all three sections are visible

**Important note:** When you modify options on a child node (most probably a source file), you create a superset of local options for this node and its own children. If you want to globally modify an option (this is the case most of the time), remember to verify that an application node is selected, **not** a child node.

#### 4.3.1 Compiler and Assembler options

Refer to the tools specific documentation for details about GCC compiler and AS assembler options.

#### 4.3.2 LD linker options

Ride7 provides some libraries. You can choose to include them or not by (un)selecting the corresponding options.

### 4.3.3 General

**Generate MAP file:** Tells the linker to generate a map file. The file generated has the same name as the *.elf* application file but with a *.map* extension.

**Include Cross References:** Outputs a cross reference table. If a linker map file is being generated, the cross reference table is printed to the map file. Otherwise, it is printed to standard output (similar to the `--cref` command-line option).

**Indirect Call Files:** Keeps track of references from data to code, which include indirect calls (function pointers, switches, etc.). This is required for CodeCompressor to operate correctly. You can deactivate it if you do not use CodeCompressor, but we recommend leaving it active all the time as it causes no problems.

**Generate Data Init:** Select if you want Ride7 to automatically generate the code for initializing the data section. Note that this option is not available for every target.

**Warn undefined symbols:** Displays only one warning for each undefined symbol, rather than one per module.

LD Linker	
General	
Generate MAP file	Yes
Include Cross References	No
Indirect Call Files	No
Generate Data Init	Yes
Warn undefined symbols	No
Startup	
Use Default Startup	Yes
Startup File	crt0.o
Scripts	
Use Default Script File	Yes
Script File	crt0.ld
Libraries	
Library Model	Small
C library	Yes
Math library	Yes
EM Monitor Library	Yes
More	
More	

#### 4.3.3.1 Startup

**Use Default Startup:** To keep the default startup file, set **Yes**.

If you are familiar with the GNU toolchain, you will probably use your own startup file. In that case set **No** and complete the **Startup File** box (or not if your startup is part of the source files).

**Startup file:** If **No** has been set for the **Use Default Startup** option, indicate the path of the startup file that you want to use. You can also see the default startup and linker script files provided by Ride7, which you can copy and modify. For standard installations of Ride7, these files are in `<RideInstallDir>\Raisonance\Ride\lib\C816\templates`.

#### 4.3.3.2 Scripts

**Use Default Script File:** To keep the default script file select **Yes**. If you are familiar with the GNU toolchain, you will probably use your own script file. In that case select **No** and complete the **Script File** box (you have to use a linker script).

**Script File:** If **No** has been selected for the **Use Default Script File** option, indicate the path of the script file that you want to use. You can also see the default startup and linker script files provided by Ride7, which you can copy and modify. For standard installations of Ride7, these files are in `<RideInstallDir>\Raisonance\Ride\lib\C816\templates`.

Please also note that there is no default linker script for the C816-TEST device. Therefore, when you use this virtual device, you must use a custom linker script.

#### 4.3.3.3 Libraries

**Library Model:** This option specifies where the library variables are located:

- Huge model: the variables are in page 0.
- Small model: they are in the other pages.

**C library:** Select if you want Ride7 to include the C library. (recommended)

**Math library:** Select if you want Ride7 to include the mathematical library. (recommended)

**EM Monitor library:** Select if you want Ride7 to include the monitor library in the link. This is required for debugging using some debuggers. This should be deselected for production. Please note that this option is not available for every target. See the device-specific documentation for more information.

#### 4.3.3.4 More

This option lets you specify options to be added to the command line (just before the linker configuration file). See LD documentation for available commands and switches.



## 4.4 Using the new version of GCC: 4.4

Version 1.08 of RKit-C816 includes a test version of the next release of GCC for C816: GCC4.4. Of course it also includes the previous version GCC3.2, which is the one that is active by default.

This section provides some tips and tricks concerning this major tool change.

### 4.4.1 Why use GCC 4.4?

#### 4.4.1.1 Performance

This new version produces much more compact code, partly (but not only) if you activate automatic removal of "dead code" (if some code in your application is never called, with the new version and this option it will not be linked and use no memory). See the Activating Dead Code removal section for details.

#### 4.4.1.2 Maintenance

In the future, all bug fixes and new features will only appear in upgrades of version 4.4 (4.x, 5.x, etc.). Version 3.2 will not be supported or maintained in the future. Problems reported on version 3.2 will not even be analyzed.

### 4.4.2 Switching to GCC 4.4

Go to the Ride\GNU directory and run the **setGCC4.bat** batch file (some systems require that you are signed in as Administrator). After this, Ride7 will compile using GCC 4.4.

After switching to GCC4, there are a few things that you will need or want to change in your environment, project options, and configuration files. You must follow the instructions below carefully.

#### 4.4.2.1 Avoid using GCC standard headers

Historically, the C816 CPUs have always used C816-specific standard headers. (stdio.h, stddef.h, etc.) However, the new GCC4 provides its own headers before the specific ones, which can lead to problems when porting projects that have been written using the old GCC3.2.

To avoid these problems, you should deactivate the use of standard includes in the Dialect section of the compiler options. (equivalent to the `-nostdinc` compiler command-line option)

#### 4.4.2.2 Removing Program Headers Alignment

To use the **Datalnit** process (device dependant) you must deactivate the **Align Headers** linker option (equivalent to the `-n` linker command-line option).

#### 4.4.2.3 Linking page0 data first

To avoid allocation problems by the linker with **page0**, you must modify your linker script and move all ".page0\*" (**.page0\_bss**, **.page0\_data**, etc.) sections to the first position in the SECTIONS part of the file, before all non-page0 data (**.bss**, **.data**, etc.).

**Note:** This has already been done in the **default linker scripts**. So if you are using the default linker script, you do not need to modify anything.  
If you are using a custom linker script, you can look at the default for an example of what should be done.

#### 4.4.2.4 Changing the standard optimization options

Optimization level **O1** produced best code size with previous GCC versions, but it now produces best execution speed with significantly larger code.

With GCC4.4, optimization level **Os** produces best code size, with the **Ignore Inline** (equivalent to `-fnoinline` command-line option) compiler option ON. This higher level of optimization in the new compiler introduces difficulties when debugging optimized code. So it is recommended to...



- Activate the **Ignore Inline** option all the time.
- Select **optimization level O0** for debugging
- Select **optimization level Os** for production, unless you want to give higher priority to execution speed over code size, in which case you should select **O1**.

**Note:** You can use the **Local Options** to optimize some source files using a different level than the rest of the project. This allows you to debug some critical functions in a project that would not fit in the device's memory if it was all compiled without optimization. After all the functions in the file are validated, you can activate optimization for this file using "Reset Local Options", which makes it use the same options as the rest of the project.

### 4.4.2.5 Avoiding the warning about DOS paths

If you compile from outside Ride (using Makefiles for example), after switching to GCC 4.4 you will receive a warning concerning DOS paths.

You can simply ignore them, or you can follow the instructions in the message to make them disappear: Create an environment variable called **CYGWIN** and give it this value: **nodosfilewarning**.

**Note:** When compiling using **Ride** you don't need to modify your environment at all, as the variable is defined locally to Ride's processes (and sub-processes).

### 4.4.2.6 Avoiding weird characters in GCC messages

If you compile from outside Ride (using Makefiles for example), after switching to GCC 4.4 you may see weird characters in GCC messages.

You can simply ignore them, or you can follow the instructions in the message to make them disappear: Create an environment variable called **LC\_ALL** and give it this value: **C**.

**Note:** When compiling using **Ride** you don't need to modify your environment at all, as the variable is defined locally to Ride's processes (and sub-processes).

### 4.4.3 Activating dead code removal

This is one of the major improvements of the new version: If you request it, the code in your application that is never called will not be linked and so it will not use any code memory. Here is the procedure for activating this feature:

1. First make sure you are using GCC4.4 as described previously.
2. Make sure you have adapted your project (linker script for page 0, optimization options, etc.) as described previously.
3. If you are using a **custom linker script**, you have to modify your linker script:

In section `.text` after this line...:

```
* (.text)
```

...add these two lines:

```
* (.text*)
```

```
* (.text.*)
```

**Note:** This has already been done in the **default linker scripts**. If your project is using the default linker script for your device, you do not need to modify anything. If you are using a custom linker script, you can look at the default for an example of what should be done.

4. Activate the **per function section** compiler option (equivalent to the `-ffunction-sections` command-line option). This places each function in its own dedicated section named `.text.<function_name>`. Without this option, and also if you use GCC3, all functions are placed in section `.text` unless explicitly placed in another section.
5. If you are using a **custom linker script** and if you need to keep some specific user-defined sections, like for example Row 62 of EM6819, then you must use the **KEEP** directive in the linker script.

**Note:** This has been done in the **default linker scripts** that need it (EM6819). So if you are using the default linker script, you do not need to modify anything. If you are using a custom linker script, you can look at the default for an example of what should be done.

6. Then activate the **remove unused sections** linker option (equivalent to the `--gc-sections` command-line option). This will tell the linker to take only the sections that are actually referenced by the application.

**Note:** The **per function section** and **remove unused sections** options did not exist with previous versions of GCC, which will not accept them and produce errors. The rest of this procedure (linker scripts modifications) can be done even if you use GCC3.3. They will not cause any problems.

7. After all this you will need to rebuild the project. To be sure use **Build**, not **Make**.

**Note:** If you are not compiling from Ride, or if you are not using the default startup, you must make sure that the linker keeps the vectors, otherwise it will remove everything, thinking it is dead code, and your final application will be empty.

The recommended way to do this is like Ride does with its default startup: add an option on the command-line of the link telling the linker to take and keep a symbol that is defined in your startup. The default startups contain the `_start` symbol, and Ride adds the `-u _start` option to the linker command line. Depending on your build environment and startup file, you can do the same or just get ideas from this.

You can also do this by placing the startup code in a dedicated section, and using the **KEEP** directive on this section in your linker script.

### 4.5 Using the old version of GCC: 3.2

#### 4.5.1 Switching back to GCC 3.2

Go to the Ride\GNU directory and run the **setGCC3.bat** batch file (as administrator in some systems). After this Ride will compile using GCC 3.2.

Then you must deactivate the **Dead Code Removal** options that are only valid for GCC4 and would generate errors if fed to GCC 3.2.

**Note:** In theory, most linker script and option modifications described above, required for working with GCC4, should also work with GCC 3.2. The exceptions are dead code removal options, which will be rejected by old versions of GCC. However, it can be interesting to start by applying these changes on your project while using GCC 3.2, to validate your application after these modifications are made, and only then switch to GCC 4.4.

#### 4.5.2 Why switch back to GCC 3.2?

Some projects might not work the same with the new version.

For example if you forgot a volatile keyword, or if your application relies on the execution time of some C code, it is possible that the optimizations performed by the new compiler will make the application fail. In these cases you will have to correct your code, and to find out how to correct it, it can be useful to recompile it using the old compiler.

**Note:** As mentioned before, the old GCC 3.2 will cease to be officially supported and maintained. So it is highly recommended for you to switch to GCC 4.4 as soon as possible. You can still use GCC 3.2, but we recommend you only do that to help you in the early phases of the switching process, or for end-of-life maintenance of mature projects.

## 5. Debugging with the simulator

Ride7 for C816 provides a simulator that can simulate the C816 cores and most of their common peripherals. The simulator lets you check your code and the interaction between your code and the peripherals before going on to debug with a hardware debugger.

This section shows the basic use of the simulator for C816 microcontrollers. However, the interface is the same for all the targets. The complete documentation about the simulator is found in the Ride7 general documentation.

Ride7 for C816 supports most of the existing C816 derivatives to various degrees. For the rest of this section, we will use the example project that we can call “toggle”.

### 5.1 Configuring the simulator options

#### 5.1.1 Miscellaneous

##### Code Exploration

If this option is set to **Yes**, the code will be explored from loading to recognize and to flag the first byte of any instruction (this is a call-tree exploration). Because the instructions have different sizes, this is needed to display an exact disassembly to the user. If it is set to **No**, the loading is faster and exploration is not complete; the code would be displayed as *db 0xxh,....* In simulation, you should explore progressively (explore command).

This option needs to be checked to allow the trace to explore the disassembly code.

##### Value Record

This option allows to record (or not) additional information at each instruction. When you unselect **No information** you can choose:

- SP: Value of the Stack Pointer after the execution of the current instruction.
- Expression: Enter the expression you want to evaluate in the Expression box.

#### 5.1.2 XTAL Oscillator

##### Crystal Frequency

Set the crystal frequency you want to simulate the program.

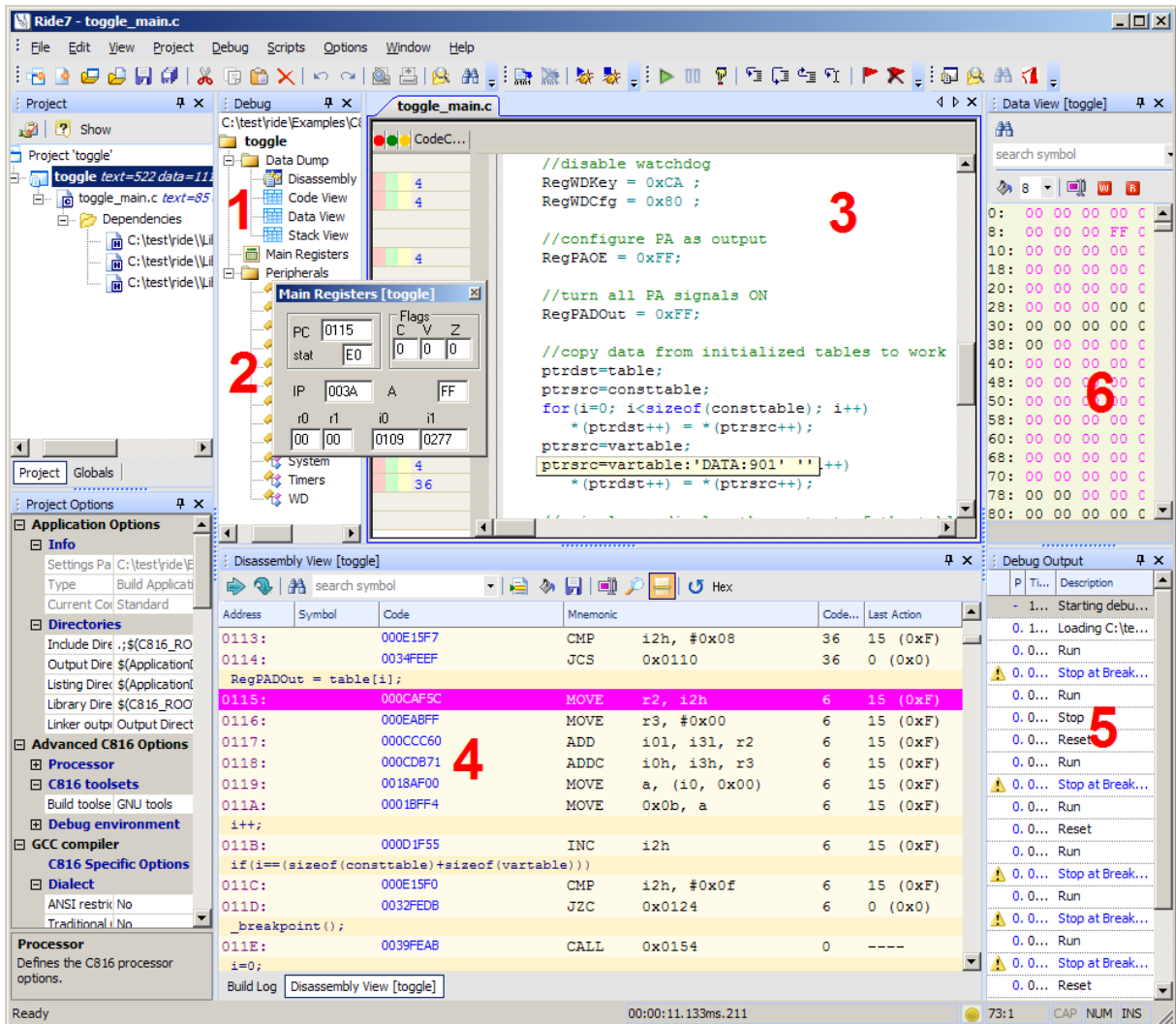
<b>C816 Simulator</b>	
<b>Miscellaneous</b>	
Code Exploration	No
Value Record	No information
Expression	
<b>Space Size</b>	
Code Size	64
Data Size	64000
Stack Size	16
<b>RC Oscillator</b>	
<b>XTAL Oscillator</b>	

### 5.2 Launching the simulator

Before launching the simulator, check the **Debug environment** options and in particular check the corresponding **Simulator C816** is selected for **Debug tool**.

To launch the simulator, type **CTRL-D** or select **Debug > Start** in the main menu. If your project has not been built, this will be done automatically before launching the simulator. Otherwise the simulator is launched directly.

<b>Debug environment</b>	
Debug tool	Simulator C816
Format	Simulator C816
Code Offset	PDK Simulator
Data Offset	RLink
Explore code	No
Start Mode	main() function entry
Start Symbol Address	main



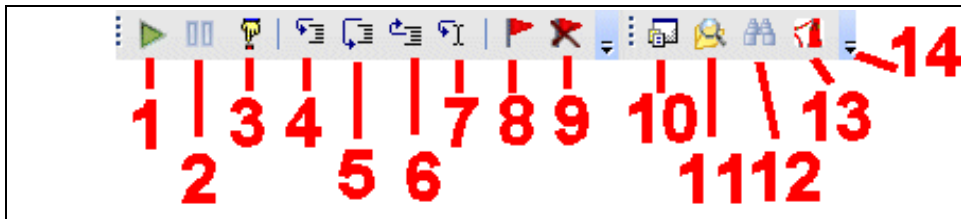
You are now in the simulator. Your Ride7 window looks like the above image:

1. The upper part of the **Debug** tab, which shows the different Data views available on a given microcontroller. To see a specific view, double-click on its name.
2. The lower part of the **Debug** tab, which shows the peripherals available on a given microcontroller. Most of these peripherals are NOT simulated. To see a specific peripheral, double-click on its name.
3. The source file as edited in C or in assembly language. The green circles on the left indicate lines that contain instructions where you can place breakpoints. The line highlighted in blue indicates the current PC value which is executed.
4. The Disassembly view window, which displays the instruction to be executed by the simulator. It is a dump of the memory where the code is located (see below for more information). Note the blue arrow at the beginning of the line indicates the current PC, as in the source window. The following columns are available in the Code window:
  - **Address:** Address where the instruction is located.
  - **Symbol:** Name of the symbol, if a symbol is located at this address.
  - **Code:** Byte-code located at this address.
  - **Mnemonic:** Mnemonic corresponding to the byte-code.
  - **Code Coverage:** Number of times the byte-code at this address was executed.
  - **Last action:** Most significant effect of the instruction during its last execution.

5. The **Debug Output** window provides feedback on the status of debugging process. Status information can include errors as well as debugging event logs. Some message lines are hyperlinked with a PC address, clicking on the PC item will display the Disassembly view at the PC address where the event occurred.
6. **Data Dumps Views** (here **Memory View**) is only available during a debug session, and allows you to observe the content of the different memory dumps. The addresses associated with symbols are highlighted with pink color. A status bar at the bottom of this view displays the symbols indicates a read or write access. You can modify the content of the dump from this view. Select the value you want to alter, type the new value and press Enter. Double clicking the ASCII value will also let you type directly the value in ASCII format.
7. The toolbar, which allows you to control the simulation (see the next section for more information).

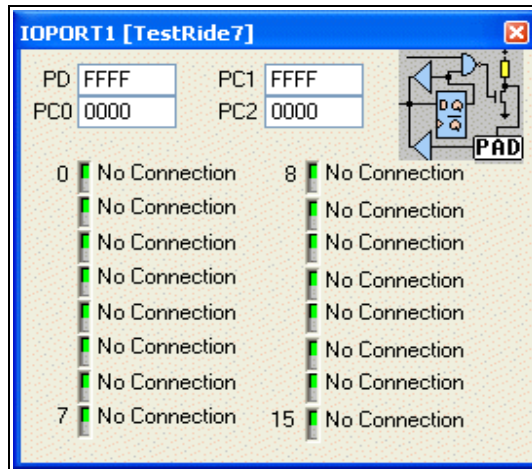
### 5.3 Using the simulator

The simulation is controlled by the simulator **toolbar**:



1. **Run**: Pressing this button launches the application. When the application is running, this button is gray, and the Pause button becomes available.
2. **Pause**: Pressing this button stops the application.
3. **Reset**: Pressing this button resets the application.
4. **Step Into**: On a function call in a line of the C source code, this button steps into the called function. If it is not a function call, it goes to the next line in the source code.
5. **Step Over**: On a function call in a line of the C source code, this button steps over the called function.
6. **Step Out**: In a function this button gets out of the function.
7. **Run To**: Runs the program until the cursor.
8. **Toggle breakpoint**: If there is no breakpoint on the current line, Ride7 sets a breakpoint on it. If there is one, the breakpoint will be removed.
9. **Clear All breakpoints**: Clears all the breakpoints set.
10. **Project**: Opens the **Project** window.
11. **Find in files**: Pressing this button opens the **Find in files** window allowing you to search an expression in project directory files.
12. **Binoculars**: This icon opens the search window.
13. **Documentation**: Pressing this button opens the **Documentation** window allowing you to open *Help html* files.
14. **Toolbar Options**: Allows you to **Add** or **Remove** buttons in the menu.

## 5.3.1 Viewing a peripheral





To view a peripheral, you must open it by clicking on the corresponding item in the peripheral tree. For example, to view Port 1, double click on the **IOPORT1** icon. The Port 1 view will appear:

This view shows the state of each of the port's pins:

- Green indicates a value of one.
- Red indicates a value of zero.

It is possible to connect each pin of the port to a Net, to VCC, to Ground or no connection. This is done by clicking on the **LED**. The registers also let you control the peripheral.

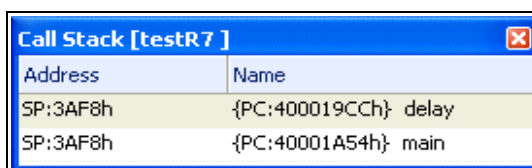
With the test application described above, click on the **Run** button  to launch the execution and then click on the **Pause** button .

You will then see the LEDs counting.

**Note:** Currently, only UART is fully simulated to provide `putchar/printf` capabilities.

For other peripherals, the views provide a comprehensive presentation of the internal registers, but the dynamic behavior of these registers exists only when running the program on real hardware via a hardware debugger (see *Debugging with Hardware Tools*).

## 5.3.2 Viewing the stack



You can view the stack by clicking **View > Debug Windows > View Call Stack**.

This operator opens the Call Stack window.

The window shows the list of functions currently in the stack, allowing you to trace the calls up to the main function or the current **Interrupt Service Routine**. Double-click on a function in the stack view to place the cursor in the associated source file.

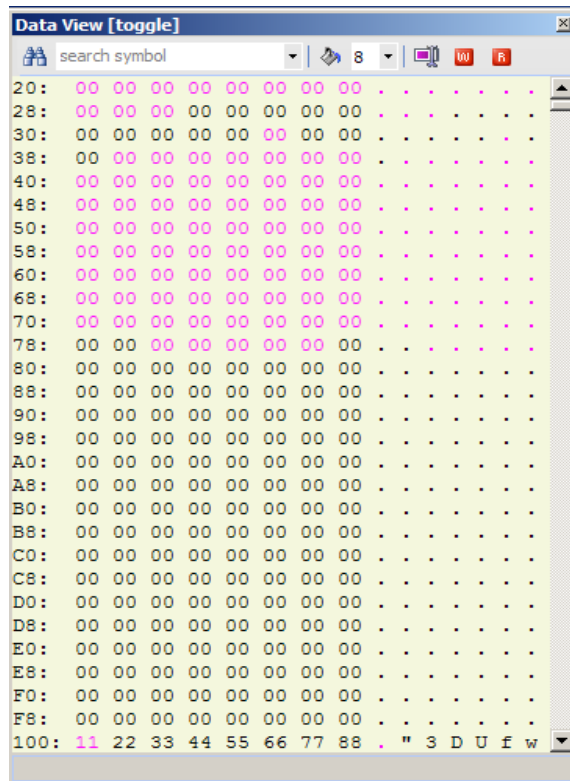
There are a few restrictions for using this view:

- It can only be used without optimization (level zero).
- It needs debugging information for all the functions in the stack.
- It does not work during functions prologue and epilogue (i.e. the very beginning and end of functions).
- It does not work properly when the current PC is not on the beginning of a C line (after a stop or assembler step).



### 5.3.3 Viewing data

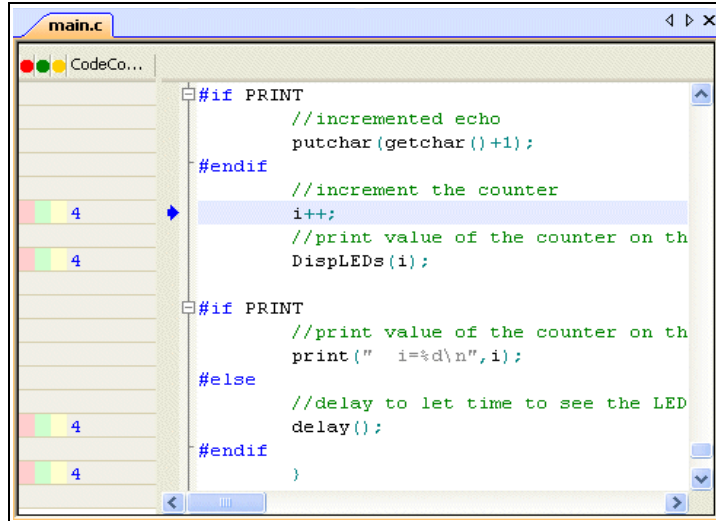
You can also see and modify the content of the Data by double clicking **Data View**. This operator will open the following window:



- The addresses in pink are those that have a symbol associated with them.
- This view is also used for setting and clearing data breakpoints, using the red "W" and "R" buttons at the bottom-right hand corner.

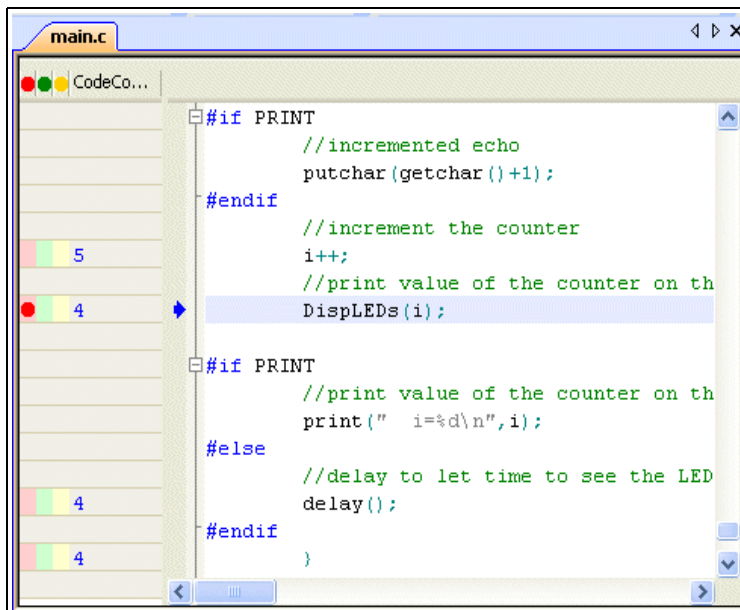
5.3.4 Using breakpoints

You can set a breakpoint either in the source file, or in the code view. Make sure that the application is not running (click on **PAUSE**). You can see in the source code, on the left hand side, lines with boxes (red, green and yellow) indicate the code lines where a breakpoint can be set.



When you click on the red box **Toggle Breakpoint** a red point appears on this line, which means that a breakpoint has been set on this line:

Then, click on the **RUN** button and the application will stop running when the line is reached:



Refer to the Ride7 general documentation for more information about the simulator user interface.

## 6. Debugging with hardware tools

In addition to the Raisonance simulator, Ride7 for C816 can be used with a number of hardware debugging tools that are provided by SEMTECH™ and EM Microelectronic-Marin Ltd.:

- RLink
- RLinkPro
- REva

Documentation for the hardware tools (programmers, debuggers, demonstration boards) that can be used with Ride7 can be found in the documentation files of each target device. These can be found in <RideInstallDir>\Raisonance\Ride\Doc.

These tools have the same user interface as the simulator described above, therefore it is recommended that you read and understand this document before reading the one that is specific to your hardware.

Within Ride7, you can choose your target hardware debugger either in the Project Options window if it is opened or with the **Options > Project Properties** menu, by selecting **Advanced C816 Options > Debug Environment**. From the **Debug Tool** option you can choose between a list of available tools.

Select the tool corresponding to your debugging hardware (Ride7 only shows you the debugging tools that are available for your target selection):

- EM6819-Monitor (using RLink GASP)
- EM6869-Monitor (using RLink GASP)
- SX144x-Monitor (using RLink SX)
- XE143x-Monitor (using RLink dSPI)
- XE-Prostart2x-Monitor (using Prostart2 board)



## 7. Conformity



### **ROHS Compliance (Restriction of Hazardous Substances)**

KEOLABS products are certified to comply with the European Union RoHS Directive (2002/95/EC) which restricts the use of six hazardous chemicals in its products for the protection of human health and the environment.

The restricted substances are as follows: lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls (PBB), and polybrominated diphenyl ethers (PBDE).



### **CE Compliance (Conformité Européenne)**

**KEOLABS products are certified to comply with the European Union CE Directive.**

In a domestic environment, the user is responsible for taking protective measures from possible radio interference the products may cause.



### **FCC Compliance (Federal Communications Commission)**

KEOLABS products are certified as Class A products in compliance with the American FCC requirements. In a domestic environment, the user is responsible for taking protective measures from possible radio interference the products may cause.



### **WEEE Compliance (The Waste Electrical & Electronic Equipment Directive)**

KEOLABS disposes of its electrical equipment according to the WEEE Directive (2002/96/EC).

Upon request, KEOLABS can recycle customer's redundant products.

For more information on conformity and recycling, please visit the KEOLABS website [www.keolabs.com](http://www.keolabs.com)

## 8. Glossary

Term	Description
C816	CoolRisc 816
Ride7	Raisonnance Integrated Development Environment

## 9. Index

7-day evaluation period.....	9	Lead.....	27
Activating dead code removal.....	17	Library Model.....	15
Activation Code.....	9	Local Options.....	16
Additional help or information.....	4	Math library.....	15
Align Headers.....	16	Memory View.....	22
allocation problems.....	16	Mnemonic.....	21
application node.....	14	new hardware dongle.....	11
C 16		nodosfilewarning.....	16
C library.....	15	O0.....	16
C816-TEST.....	12	O1.....	16
CE.....	27	old hardware dongle.....	11
child node.....	14	Os.....	16
Code Coverage.....	21	Purpose of this manual.....	4
Code Exploration.....	20	Raisonance tools for the C816 family.....	5
Compliance.....	27	Registration error.....	10
Configuring the GNU GCC toolchain.....	14	Reset Local Options".....	16
Configuring the simulator options.....	20	REva.....	26
Conformity.....	27	Ride7 version 7.36.....	10
Creating a new project.....	13	Ride7 window.....	20
Creating a project using the GNU toolchain.....	12	RLink.....	26
Crystal Frequency.....	20	RLinkPro.....	26
CYGWIN.....	16	ROHS.....	27
Data Dumps Views.....	22	Scope of this manual.....	4
Data View.....	24	Script File.....	15
DataInit.....	16	SECTIONS.....	16
Debug > Start.....	20	Serial Key.....	9
Debug Output.....	22	simulator toolbar.....	22
Debugging with hardware tools.....	26	start.....	17
Debugging with the simulator.....	19	Startup file.....	14
Directive.....	27	Supported derivatives.....	6
EM Monitor library.....	15	symbols.....	14
Enterprise version.....	9	Use Default Script File.....	14
FCC.....	27	Use Default Startup: T.....	14
ffunction-sections.....	17	Using an example project.....	12
gc-sections.....	17	Using breakpoints.....	25
GCC 3.2.....	18	Using the simulator.....	22
GCC_EXEC_PREFIX.....	12	Value Record.....	20
GCC4.4.....	15	View Call Stack.....	23
Generate Data Init.....	14	Viewing a peripheral.....	23
Generate MAP file.....	14	Viewing data.....	24
globally modify an option.....	14	Viewing the stack.....	23
Include Cross References.....	14	Warn undefined symbols.....	14
Indirect Call Files.....	14	WEEE.....	27
Install new Ride7/kit.....	9	Windows 7.....	9
Introduction.....	4	Windows Vista.....	9
IOPORT1.....	23	_start.....	17
KEEP.....	17	--gc-sections.....	17
Launching the simulator.....	20	-ffunction-sections.....	17
LC_ALL.....	16	-u_start.....	17

## 10. History

Date	Description
17 Mar 09	Initial version
21 Sep 10	Template updated
12 July 11	Added new registration procedure
06 Dec 11	Modified new registration procedure
27 Mar 2012	Correct and clarify
26 Apr 2012	Avoiding weird characters in GCC messages





#### **Disclaimer**

Information in this document is subject to change without notice and does not represent a commitment on the part of the manufacturer. The software described in this document is provided under license and may only be used or copied in accordance with the terms of the agreement. It is illegal to copy the software onto any medium, except as specifically allowed in the licence or nondisclosure agreement.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without prior written permission.

Every effort has been made to ensure the accuracy of this manual and to give appropriate credit to persons, companies and trademarks referenced herein.

This manual exists both in paper and electronic form (pdf).

Please check the printed version against the .pdf installed on the computer in the installation directory of the most recent version of the software, for the most up-to-date version.

The examples of code used in this document are for illustration purposes only and accuracy is not guaranteed. Please check the code before use.

**Copyright © KEOLABS 2013 All rights reserved**