RAISONANCE

a KEOLABS brand

RLink & REva for EM6819

Raisonance Tools for C816 family

Getting Started

Document version
7 May 2013

Contents

1. Introduction

This document describes using an RLink and a REva board for an EM6819 with Ride7 and the monitor. If you are not using this type of hardware this document is not relevant to you.

This document assumes that you have already read and understood the C816 Getting Started for Ride7, that you know how to create and use projects for making applications, and how to use the Ride7 debugger. It does not repeat information from these other documents. You can find C816 Getting Started document by clicking in Ride7: **> Help > View documentation** under **C816\Ride7 for C816.**

1.1 Purpose of this manual

This guide can be used by anyone interested in programming and debugging EM6819 targets using Ride7 for C816.

1.2 Scope of this manual

This document describes how to get started using Ride7 for C816 to compile and debug your application or one of the included sample applications. It assumes that you have the prerequisite knowledge of C and C816 assembly languages.

1.1 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, or contact the microcontroller support team.

Microcontroller website: www.raisonance.com

Support extranet site:	support-raisonance.com (software updates, registration, bugs database, etc.)
Support Forum:	forum.raisonance.com/index.php
Support Email:	support@raisonance.com

For information and support about EM68xx chips, contact EM Microelectronic:
http://www.emmicroelectronic.com

1.2 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. Presentation of the tools

Ride7 for C816 comes with the following tools from Raisonance:

- **Ride7**: PC software, designed by Raisonance, that makes applications for the EM6819, and also programs and debugs them using RLink.
- **RLink**: includes specific tools that allow project development from the beginning to the end.
- **EM6819 monitor**: 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).

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

2.1 Ride7

Ride7 is the PC software, designed by Raisonance, that you can use to make applications for the EM6819, and also to program and debug them using RLink. Please read the Getting Started C816 document, *GettingStartedC816_Ride7.pdf*, before continuing.

2.2 RLink

The RLink hardware dongle, designed by Raisonance, programs and debugs the target EM6819 chip from a PC. Technically, it is a bridge between a PC USB and many serial protocols. The GASP protocol used with the EM6819 devices is one of the supported protocols (in certain RLinks).

You must install Ride7, which installs the RLink USB driver, before you connect the RLink to your PC.

The RLink features the standard GASP connector as defined by EM Microelectronic. This GASP connector connects to an adaptor which allows communication with the EM6819 when it is powered in the range 1V to 3.3V.

RLink can be used with commercial demonstration boards like the REva, and also on any custom board featuring the EM6819 chip and a GASP connector.

There are two RLinks that use the GASP protocol RLink-GASP-PRO and RLink-GASP-Standard:

- RLink-GASP-PRO programs and debugs without limits.
- RLink-GASP-Standard has unlimited programming capability, but debugging is limited to applications no larger than 2K instructions (excluding rows 62 and 63 of sector 5). It should be used for evaluation or for programming in production. The RLinks embedded in the REva boards included in the EM6819 starter kits are of this type.
- Other RLinks cannot use the GASP protocol.

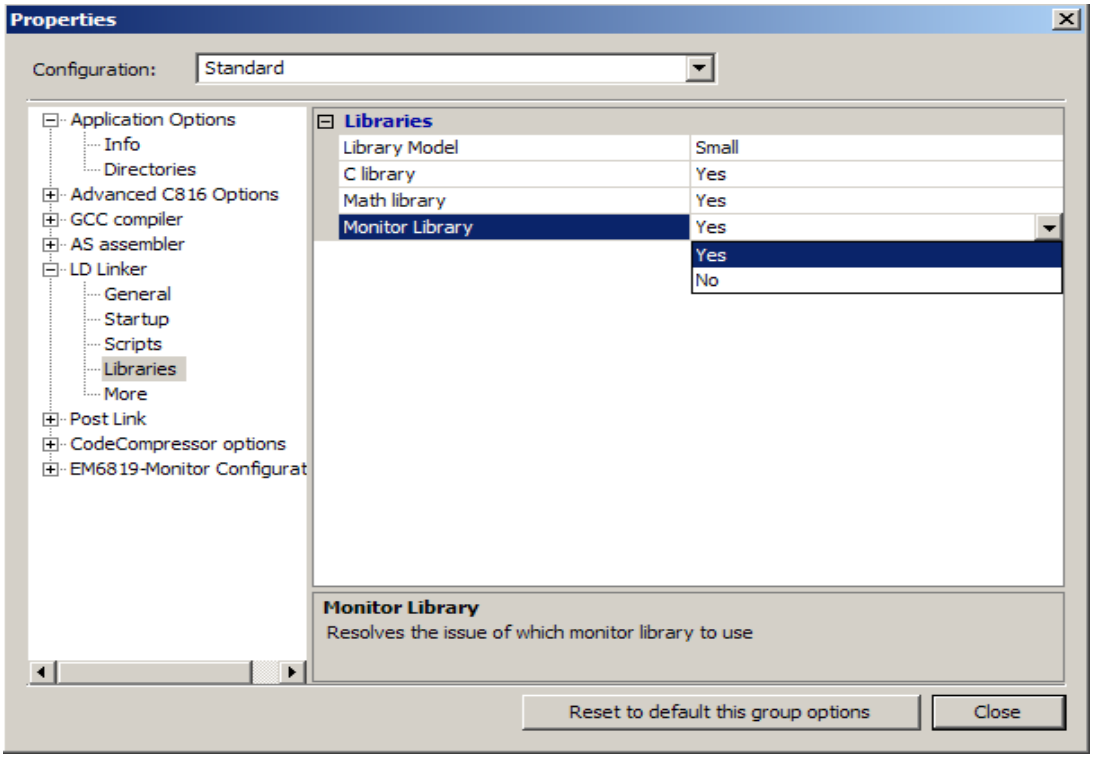
2.3 EM6819 monitor

The monitor is a piece of software that you can include in your project. It communicates with the RLink using the GASP protocol. This allows you to debug your application from your PC while it is running in the chip, using the same graphical interface as with the simulator. The monitor provides you with:

- Unlimited code breakpoints allowing you to stop the execution at defined code locations.
- One data breakpoint (Read, Write or both) allowing you to keep an eye on a defined data location.
- Step by step capability so you can see the execution of your application line by line (of C) or instruction by instruction (assembler).
- Stop while running capability, so you can see what is happening at any moment.
- Ability to read and write registers and memory locations.

To include (or exclude) the monitor in your application, you must check (or uncheck) the “Monitor Library” linker option. Click **Options > Project Properties > LD Linker > Libraries** and set option **Monitor Library** to **Yes**.

Note: If you do not check this option, you cannot debug.

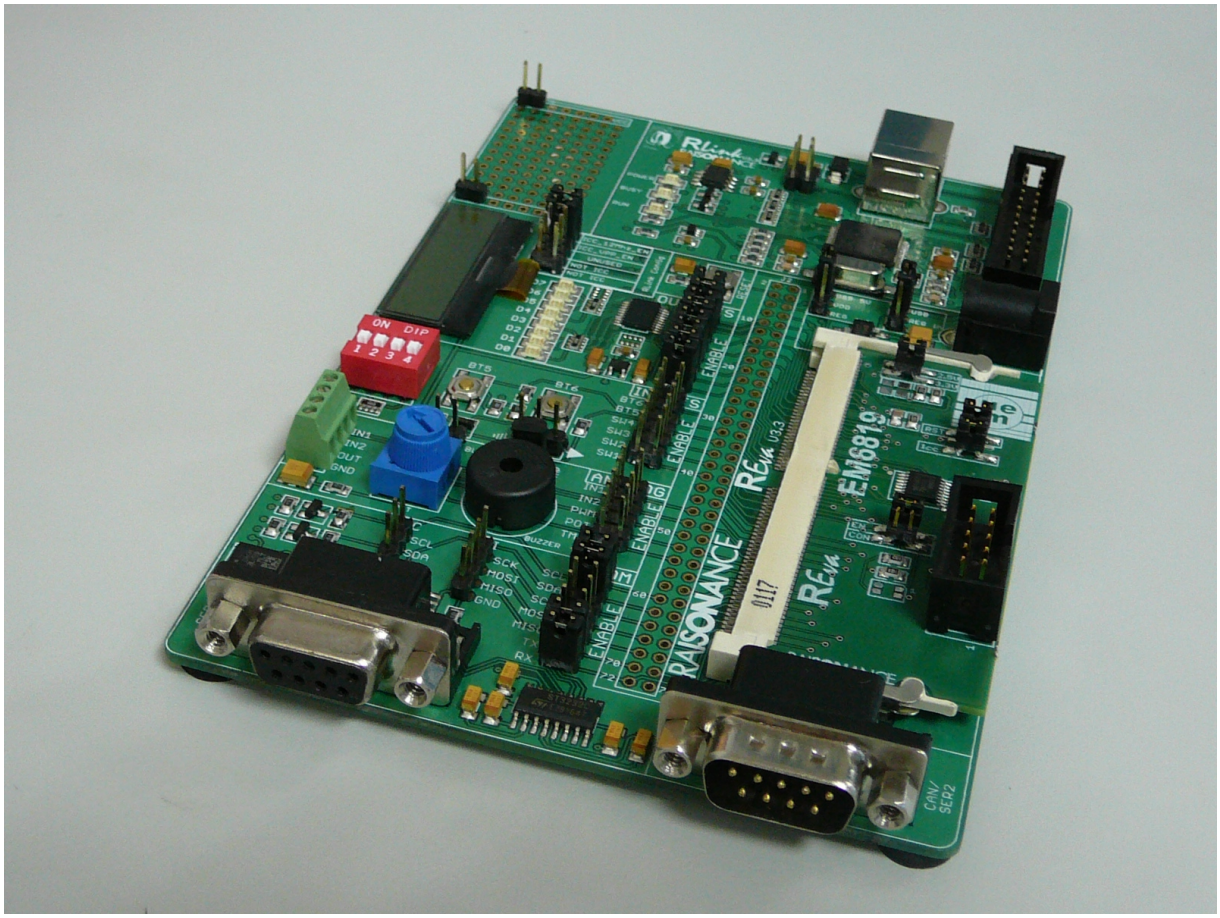


When you have finished designing your application, or if it gets too big, you might want to remove the monitor code from it. Click **Options > Project Properties > LD Linker > Libraries** and set option **Monitor Library** to **No**. This tells the linker not to include the monitor code in the application.
Of course, once you have done this, you cannot debug it anymore, you can just program and execute it. As with any linker option change, you must link the application again for the change to be taken into account.

Note: When you are NOT debugging, the application runs the same way whether or not you included the monitor code in it when you built it. Including the monitor code in the application does not mean you must use Ride7 for executing it. The only reason to remove the monitor code is to gain code space, or if you want to have full control of every byte of code in your application.

2.4 EM6819 REva board

The EM6819 REva board consists of the EM6819 daughterboard plugged on the REva motherboard.



Together they make an EM6819 demonstration board.

- The REva motherboard is a generic demonstration board that provides most of the standard features required of a demonstration board. It also includes an RLink-GASP-Starter that allows debugging of the EM6819 device on the daughterboard (with a 2K instruction limit on code size), and unlimited programming. The whole board can be powered by the USB of the RLink, or by the Jack connector. The programming and debugging can also be done by an external tool (by another RLink without limitation, for example).
- Please refer to the REva documentation for more information that you should find in *C:\Program Files\Raisonance\Ride\Doc*.
- The daughterboard, which includes the target EM6819 device to be programmed and debugged, is plugged into the SO-DIMM connector of the REva board. This daughterboard is designed and maintained by EM Microelectronic. Please contact them directly for more information, datasheets, schematics, etc.

Note: The embedded RLink and some of the REva peripherals do not support voltages lower than 2.5V. Therefore, while using the REva board, you cannot set the power lower than 2.5V. To do this you need to use your own board and a stand-alone RLink with a GASP ADP.

3. Debugging an example project

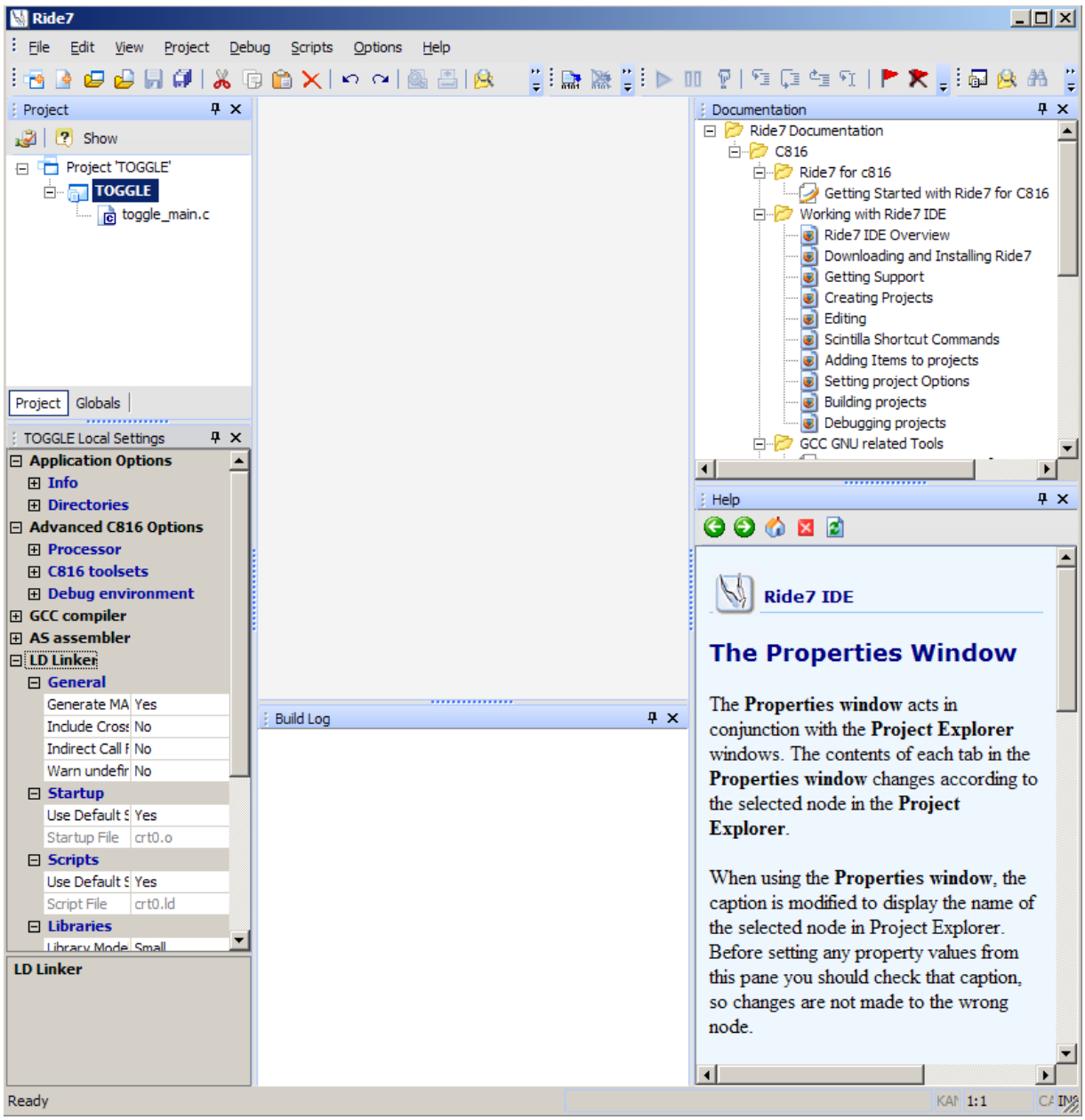
This part of the document shows you the features of the monitor and their use through a quick tutorial. We will debug one of the examples for the REva board provided with Ride7.

3.1 Opening the example project

First, open Ride7 using Windows: **Start > Programs > Raisonance Tools > Ride7 > Ride7**.

Next open the project, using **Project > Open Project** in Ride7 and select file *C:\Program Files\Raisonance\Ride\Examples\C816\EM6819\Toggle_IRQ\Toggle.prj.*

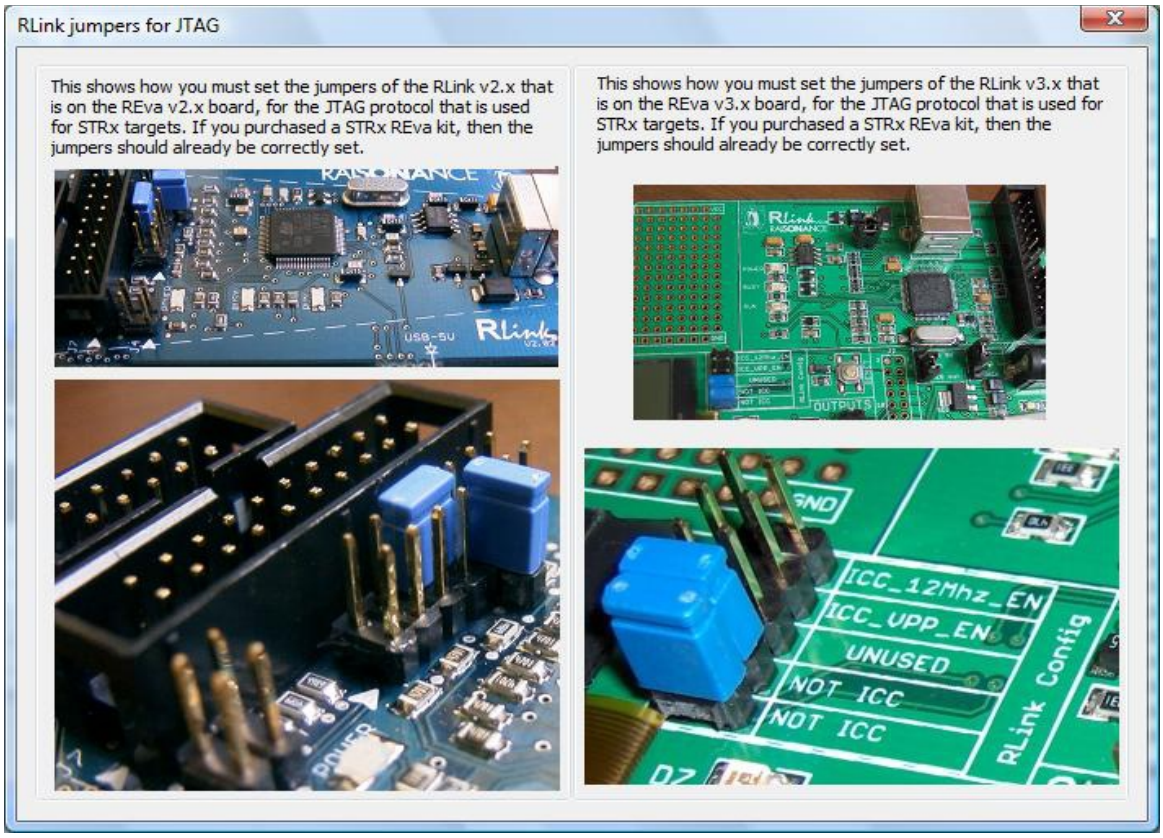
Now the project is opened and is ready to be used.



3.2 Connecting and testing the hardware

Doing all these checks seriously is very important! It is the best way to make sure that your whole system is working and is correctly configured, powered and connected (RLink, USB driver, target CPU, target board, connections, etc.).

1. Connect your stand-alone RLink or REva-embedded RLink to your PC using the USB cable.
2. When using a starter kit's embedded RLink (no plastic casing), ensure that your jumpers are set correctly on the RLink.



3. Check that the POWER and BUSY LEDs of the RLink are lit.
4. After the driver is loaded and Windows has enumerated the RLink correctly, the BUSY LED of the RLink should turn OFF.

Note: The first time you plug this RLink on this PC, loading the driver can take some time (or not, depending on the version of Windows) This driver was installed by the Ride7 installation program. Let Windows search for it automatically, in your Ride7 installation directory and in Windows system directories, but not on the web. The next time(s) you plug in the RLink, it should be much faster.

5. Then, if you are using the stand-alone RLink, connect the RLink to the target board, and power the target board.
6. Check that the POWER LED is lit on the target board (between the Jack and the daughterboard).

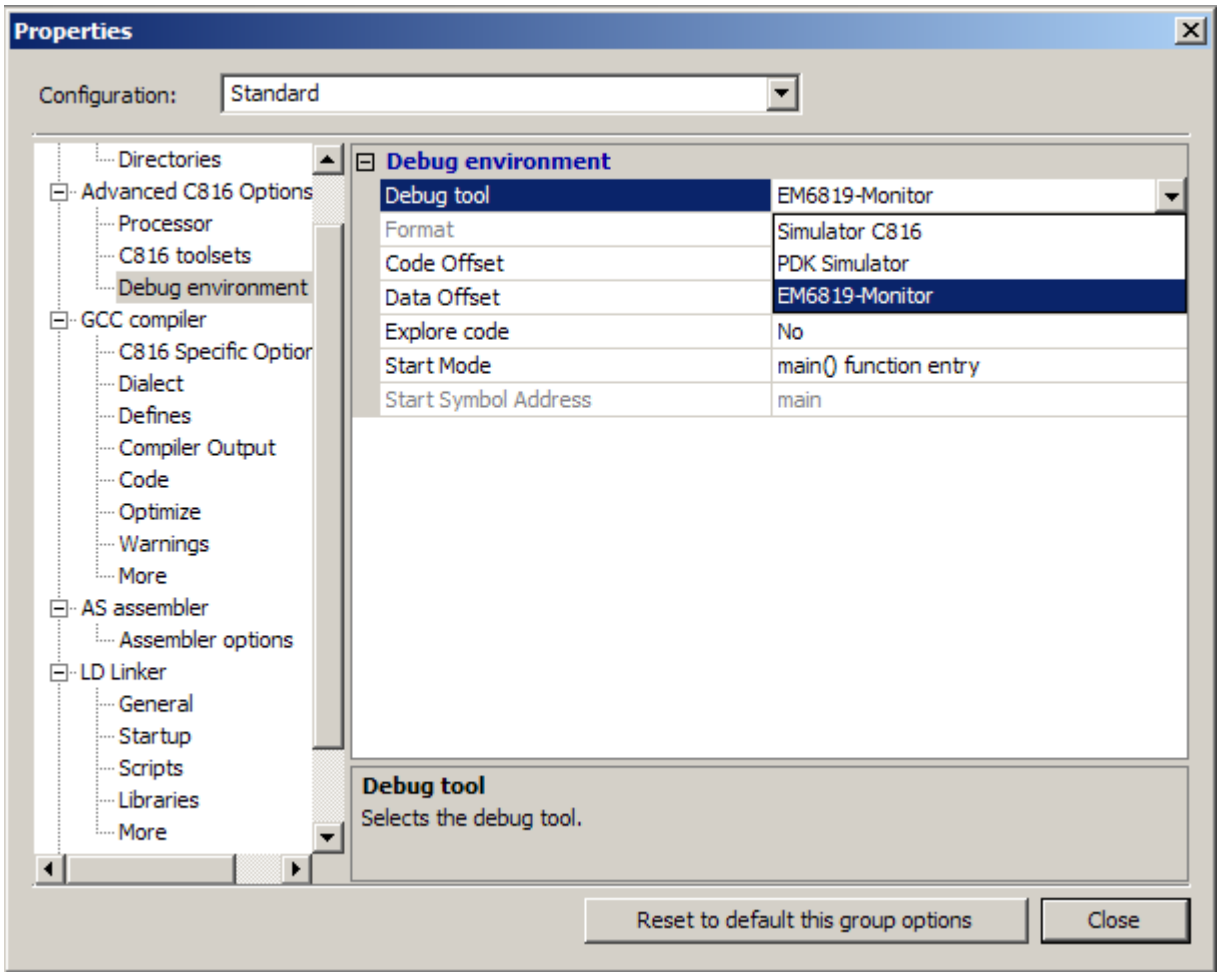
Note: You might need to change the jumpers to select Jack or USB power depending on the power source you choose to use. See the REva documentation for more information.

From this point, you should not need to touch the board until the end of the tutorial.

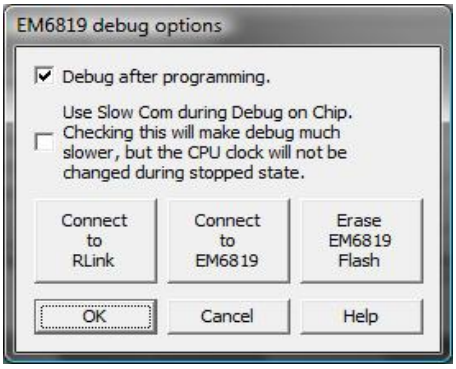
3.3 Setting debug options

Before debugging check that the project options are correctly set.

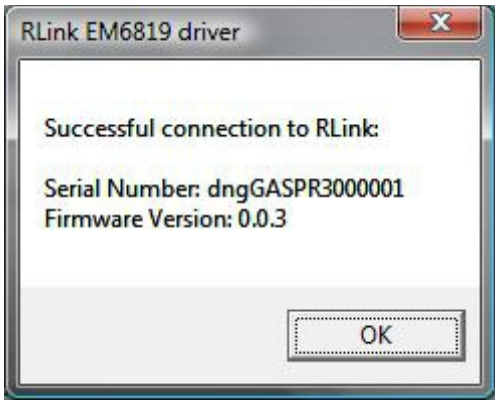
- Open the debugging options window, in which you can tell Ride7 to use the RLink instead of the software simulator for debugging (**Options > Project Properties > Advanced C816 Options > Debug environment**).



- When **EM6819-Monitor** is selected you then select **EM6819-Monitor Configuration** and click on **Click here to open options dialogue box** to open the **EM6819 debug options** box. This window allows you to test and configure the debugger, the target EM6819 and their connections. It also allows you to erase the Flash. Click **Connect to RLink** to check the connection between the PC and the RLink.

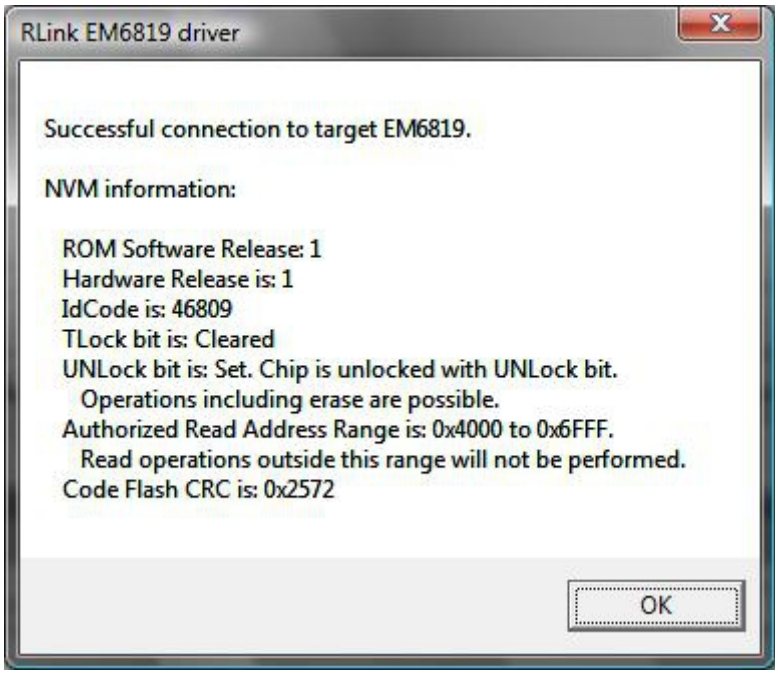


- If successful, this also displays the RLink Serial Number, which Raisonance will need for any support request (unless of course, reading the Serial Number fails…).



Note: This Serial Number is NOT written on the RLink. The only way to find it is with this test.

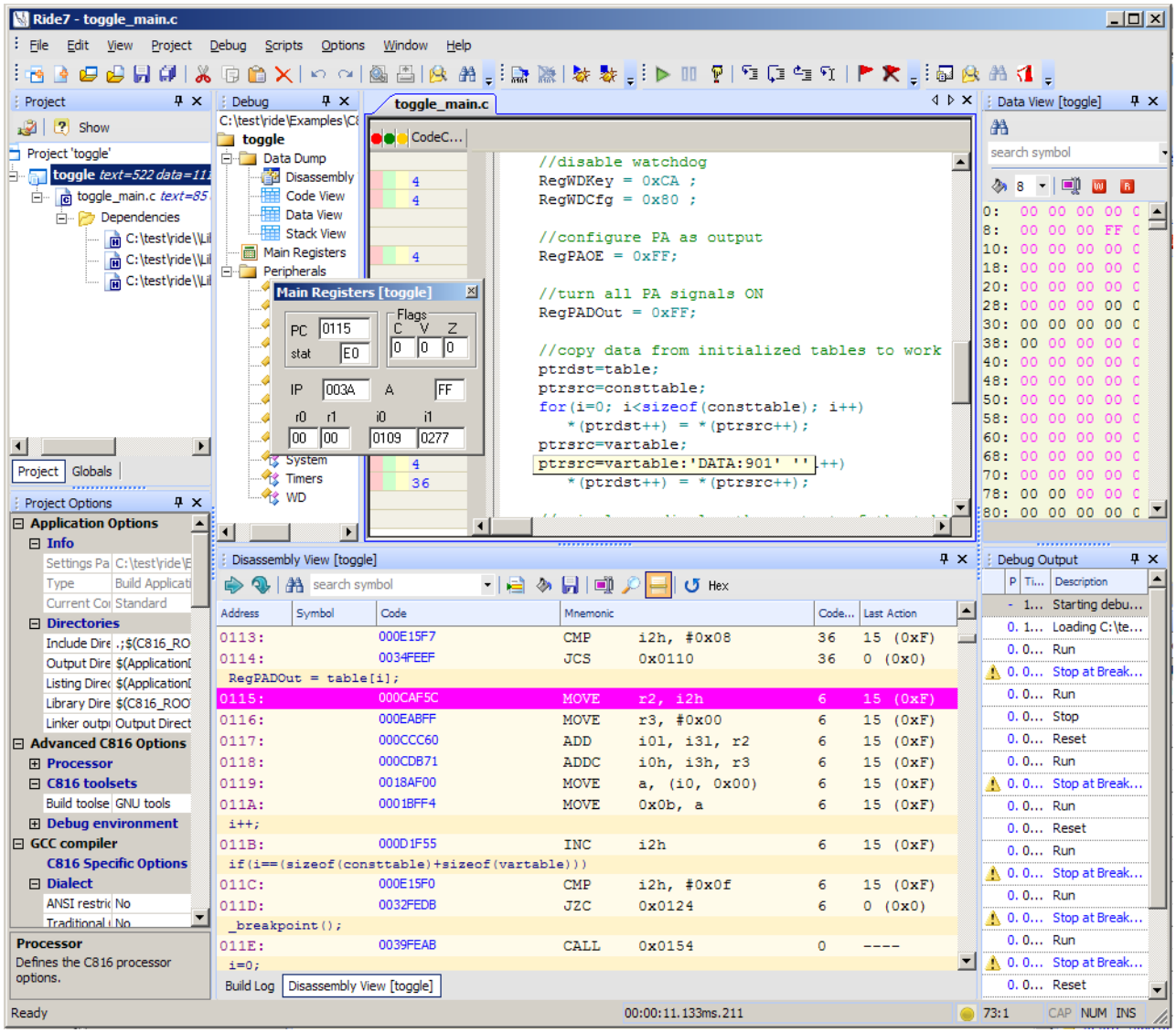
- Click **Connect to EM6819** to check the connection between the RLink and the EM6819, and the EM6819 power. If successful, this also displays information about the target EM6819, like for example the CRC of the program currently loaded in the Flash.
  See the EM6819 documentation from EM Microelectronics for more information on these figures.



Note: If you get any errors while doing this, you must understand and solve them before going on, because the next steps cannot work if these checks fail.
The error messages are usually clear enough for you to understand and solve any problem. If not, please contact our support team.

3.4 Starting the debug session

To start the debug session, select **Debug > Start**,
Your application will be loaded, run and stopped on `main()` function.
The Ride7 environment should look similar to the one shown below:



Note: From a user interface point of view, basic debugging functions (stopping and resuming CPU execution, setting a breakpoint, single-stepping, checking memory and registers, etc.) are identical, whether you are using the simulator or a hardware debug tool.
Refer to the *Getting Started with Ride7* to get familiar with the simulator before starting to work with the hardware debuggers.

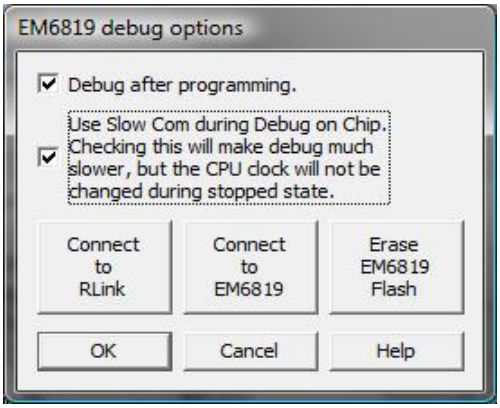
3.5 Using Slow Com mode

The GASP protocol requires that the EM6819 CPU is not too slow compared with the host (RLink). Therefore, the EM6819 monitor in normal mode must reconfigure the clock configuration whenever it communicates with Ride7. (See the monitor limitations, in the last section of this document).

This means that during Stopped state, the CPU clock will be reconfigured at 2MHz (internal 2MHz clock divided by one). This does not usually have much impact on your application's execution, since the original clock configuration is restored whenever execution resumes. However, with certain configurations of peripherals, (PWM, motor control, etc.) this might be a problem. This is why we provide a Slow Com mode.

To use Slow Com mode:

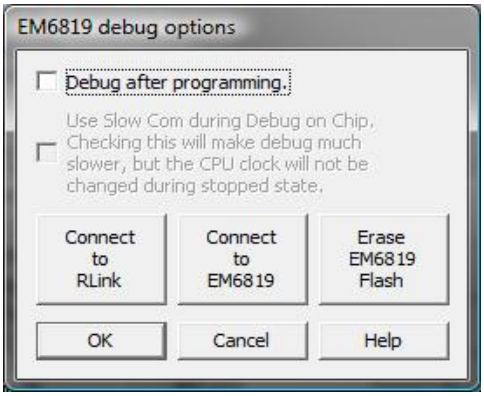
1. Close the debug session.
2. Open the advanced debug options **Options->Debug->Advanced Options**).
3. Select the **Slow Com mode** option.
4. Start the debug again, which will work the same as before, except that it will be slower. (This example does not use any features that are affected by the CPU clock during Stopped state.)



Slow Com mode tells the EM6819 monitor not to reconfigure the clocks when entering Stopped state. For the debug to work in these conditions, Ride7 has to slow down its communication a lot. Consequently, debugging is much slower, especially stepping. Therefore, we recommend you use this option only if the change of the clock configuration during Stopped state is a problem for your application.

3.6 Programming without debugging

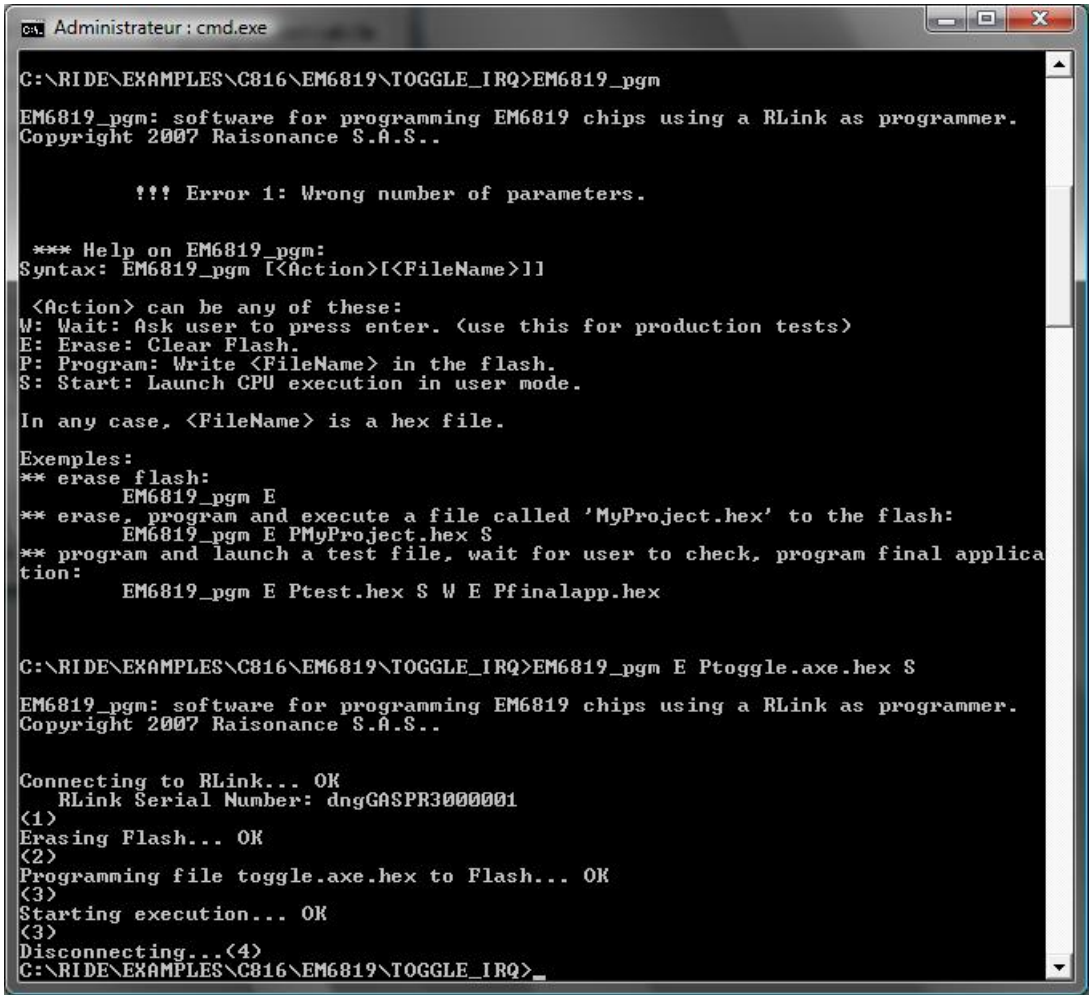
During application development it is sometimes preferable, instead of debugging, to simply program the Flash and let the application run. To do this, terminate the debug session. Go back to the advanced debug options and uncheck the **Debug after programming** checkbox.



3.7 Using EM6819_pgm.exe

Once your application is designed, debugged and validated, you will probably want to mass produce it. This means you will need to program a lot of EM6819 devices with the same code very quickly. If the operator is a subcontractor from another company, you might not want him to have your source code. For these situations, we provide an executable called *EM6819_pgm.exe* that allows you to directly program hex files (generated by Ride7 during the link of applications) to the EM6819 devices without using Ride7. It can easily be called by a higher-end production and test programs, or just a batch file.

- Open a command prompt using Windows **Start** menu and use CD command to go to the example project's folder.
- Execute `EM6819_pgm`, to see the most recent Help for using it.
- Execute `EM6819_pgm E Ptoggle.axe.hex S`. This erases the EM6819 Flash, programs it with the example's code from the hex file and starts its execution.



3.8 Other examples

Ride7 is installed with at least two other examples that deserve your attention, they can be found in *C:\Program Files\Raisonance\Ride\C816\EM6819*

- The Toggle example is the simplest possible application for EM6819. You can use it as a starting point for your new applications by duplicating it and modifying the copy.
- The Toggle_Custom example is similar to Toggle, but it also shows how to use custom startup and linker script files. Use it as a starting point for your applications if you need to modify one of these template files.

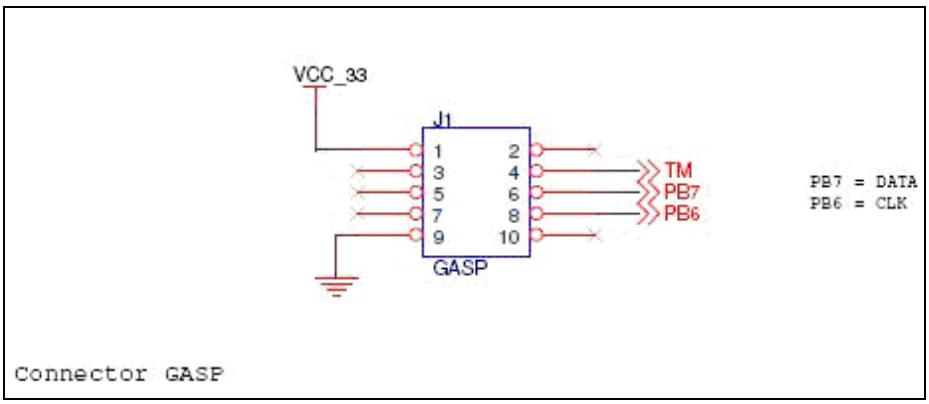
4. Debugging your own application

This chapter describes using the monitor with your own application - what you need to do and what you must be careful about. We cannot guarantee the debugging will work unless you follow this advice.

4.1 Including the GASP connector on the board

When you design your system's PCB, you must take care that it includes the GASP connector, and that this connector is correctly connected to the EM6819 (3 signals), ground, and power (for a total of 5 signals)

The lines must not be too long in order to avoid disturbances. We cannot guarantee that the RLink and EM6819 will be able to communicate using GASP if the lines are longer than 10cm between the GASP ADP and the EM6819 CPU.



4.2 Monitor limitations

The monitor is a level-0 IRQ

- It enables some level-0 interrupts and GIE and needs nested interrupts. It will not work properly if the application disables these features.
- If your application uses level-0 ITs but does not enable GIE, it will work in debug but not in normal execution.
- Other interrupts of any level will not be triggered while the monitor has stopped the execution.
- If the application (more precisely, its Startup file) does not handle nested interrupts, you will not be able to stop in ISRs of any level. The standard Startup does handle nested interrupts.
- If you do not use the default Startup and linker script from Raisonance, we cannot guarantee that the monitor will work. However, it will work in most cases if you do not modify the sensitive parts, which are identified as such by comments.
- Pressing the STOP button in Ride7 while the CPU is in `HALT` instruction will make it exit Halt mode, altering the normal execution of the application. Ride7 has no way to detect it and resume exec properly in Halt mode.
- You cannot set breakpoints on the level-0 IRQ vector, or at the beginning of the level-0 IRQ decoding code in the Startup. Ride7 will refuse it. You can set breakpoints in your ISRs of any level. See the Startup code for more information.
- It will not work if the watchdog is activated. The monitor disables the watchdog whenever it enters Stop state and never enables it again. Even then, your application should deactivate the watchdog if you plan to debug.
- It cannot Stop the execution if the CPU is in Sleep or PowerDown mode.

It needs about 20 bytes of software stack

- Beware of stack overflow. If the SW stack is full, the debugger will break the application data. Ride7 has no way to detect it and warn you properly.
- The values shown in the data view just below i3 are not correct, you cannot modify them when stopped, and you should not set the data breakpoint in this area.
- You cannot modify i3 using the monitor. (and you must be careful when modifying the PC)
- If the application modifies i3 to some inconsistent value, the monitor will crash. Especially, when stopping in or stepping over the modification of i3, it will often crash between the modification of the first byte of i3 and the second. Ride7 has no way to detect it and exit properly.
- It initializes i3 to some consistent value at reset. So if your application uses a custom startup that does not initialize i3, it might work in debug but not in normal execution mode (power-up without RLink).

It needs about 220 instructions of code

- If your application is larger than (<flash size> - 220), you cannot include the monitor in it.
- If you are using CodeCompressor, you must make sure you configure it to not modify the monitor.
- You cannot set breakpoints in the monitor code. Ride7 will refuse it.

It needs to access the Flash during the execution of the application

- Do not activate any protection or lock features if you plan to debug. Design and debug the application without any protection, and then activate protections just before going in production. This means you must have a way to test the application without debugging.

It uses the GASP, DoC peripherals and their SFRs, it is not allowed to modify the other peripherals configuration

- If the application modifies the GASP or DoC registers, the monitor will crash. Ride7 has no way to detect it and exit properly.
- The values displayed in the data view for these registers are not consistent and cannot be modified.
- The other peripherals remain active while in the Stop state. PWMs go on toggling IOs, which is good; Timers go on counting time, which could be bad, etc.

It needs a CPU clock faster than the internal GASP clock of RLink, and fast wake up mode activated for communicating with the PC through the RLink during stopped state, and it constantly communicates with the target

- In normal mode, the monitor changes the clock configuration to 2MHz during Stopped state. It also activates the fast wake up mode. The application's clock is restored when the execution resumes.
- Some peripherals, like PWMs, might be affected by the Stopped state.
- The “Slow Com” option tells the monitor to leave the clock configuration as it is. Then, debugging will be much slower (Fast wake up mode is still required and automatically activated by the monitor).
- The power consumption measurements made during debug (including during Running state) are not consistent.

5. 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*

6. Glossary

Term	Description
ARM	Advanced RISC Machine
C816	Coolrisc 816
DoC	Debug on Chip
GASP	EM Microelectronic serial protocol
RBuilder	Application builder that allows users to configure device peripherals and out put the required C code automatically for their applications. Code is based on libraries provided by the manufacturer.
REva	Raisonance evaluation platform – modular evaluation boards with main evaluation board (motherboard) and daughter boards featuring different microcontrollers
RFlasher	Raisonance Flasher: Programming interface for user-friendly flash programming
Ride7	Raisonance Integrated Development Environment
RLink	Raisonance's versatile in-circuit debugger and programmer for 8-bit and 32-bit microcontrollers
SFR	Special Function Register

7. Index

8. History

Date	Description
17 Mar 09	Initial version
30 Sep 10	Layout reformat
7 May 2013	Modified cover page, final page and section 1.3 Additional help or information for KEOLABS

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.