

Drivers

series

SIGNUM SYSTEMS CORPORATION

RDI Drivers for Third Party ARM Debuggers
with Raisonance RLINK-Pro

Installation Instructions

SIGNUM
S Y S T E M S

COPYRIGHT NOTICE

Copyright (c) 2007 by Signum Systems Corporation. All rights are reserved worldwide. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of Signum Systems.

DISCLAIMER

Signum Systems makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Also, Signum Systems reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Signum Systems to notify any person or organization of such revision or changes.

WARRANTY

Signum Systems warrants to the original purchaser that this product is free of defects in material and workmanship and performs to applicable published Signum Systems specifications for a period of SIX MONTHS from the date of shipment. If defective, the product must be returned to Signum Systems, prepaid, within the warranty period, and it will be repaired or replaced (at our option) at no charge. Equipment or parts which have been subject to misuse, abuse, alteration, neglect, accident, unauthorized installation or repair are not covered by warranty. This warranty is in lieu of any other warranty expressed or implied. IN NO EVENT SHALL SIGNUM SYSTEMS BE LIABLE FOR CONSEQUENTIAL DAMAGES OF ANY KIND. It is up to the purchaser to determine the reliability and suitability of this product for his particular application.

SIGNUM
S Y S T E M S
11992 CHALLENGER COURT
MOORPARK, CA 93021, U.S.A
PHONE 805 • 523 • 9774
WWW.SIGNUM.COM

Purpose *This document describes the installation process for the Remote Debug Interface (RDI) driver used with the Raisonance RLink-Pro in-circuit emulator for the STR processors. The driver is incompatible with other RLink emulators. Outlined are the software configuration processes for several popular ARM debuggers. (Other RDI compatible debuggers should be configured similarly.)*

Installing the Driver

1. The Setup program for installing the Signum RDI driver for the RLink emulator from Raisonance is invoked during the RIDE environment installation process. In the opening dialog box of the Setup program, select the destination folder for the RDI driver (Figure 1).

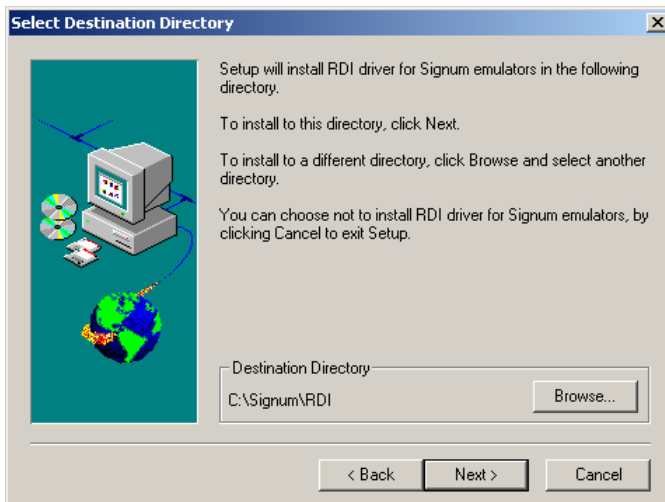


FIGURE 1 Selecting the folder for the driver.

In the confirmation dialog box, press the Next button to copy the driver files to your hard drive:

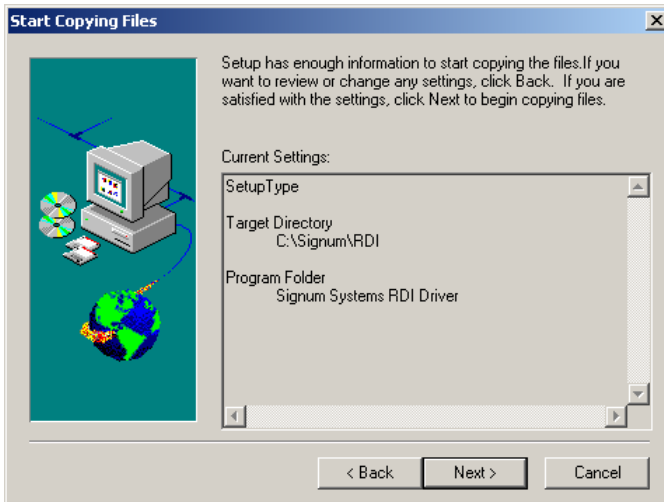


FIGURE 2 Confirming the driver location.

2. Connect the emulator JTAG cable to the JTAG connector on your ARM system or evaluation board. For examples of connecting the emulator with selected target boards, see the Appendix. Turn the emulator on first, and then turn on your target board. Finally, configure your debugger, as described in the next *Configuring the Debugger* section.

Configuring the Debugger

With the ARM RDI Driver installed, you are ready to configure your Debugger. The configuration process varies from debugger to debugger. In the next several sections, you will find configuration details for several popular debuggers:

- ARM Ltd. RealView Debugger
- ARM Ltd. AXD, SDT and ADS Debuggers
- Green Hills Software Multi-2000 Debugger

- IAR Software Embedded Workbench for ARM (EWARM)
- Mentor Graphics XRAY
- Palm Software Universal Debugger (PUD)
- GNU gdb Debugger
- Metrowerks CodeWarrior

ARM RealView Debugger

1. Start RealView Debugger and select the Click the Connect to Target (Figure 3) or the File | Connection | Connect to Target menu option.

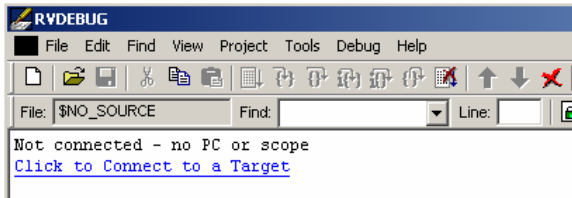


FIGURE 3 RealView Debugger's opening screen (upper left corner).

2. Select the Add/Remove/Edit Devices option from the right-click popup menu (Figure 4).

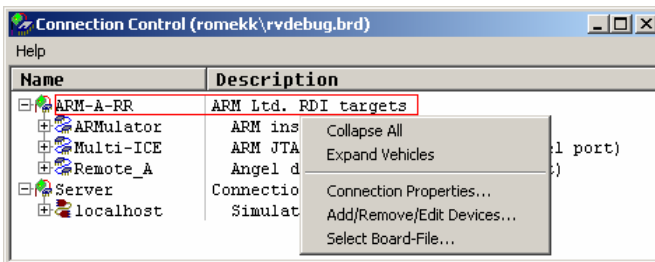


FIGURE 4 The Connection Control dialog box.

3. The RDI Target List dialog box appears. Press the Add DLL button (Figure 5) and browse for the driver file SigJdsDI.dll found in typical installations in the \Signum\RDI folder.

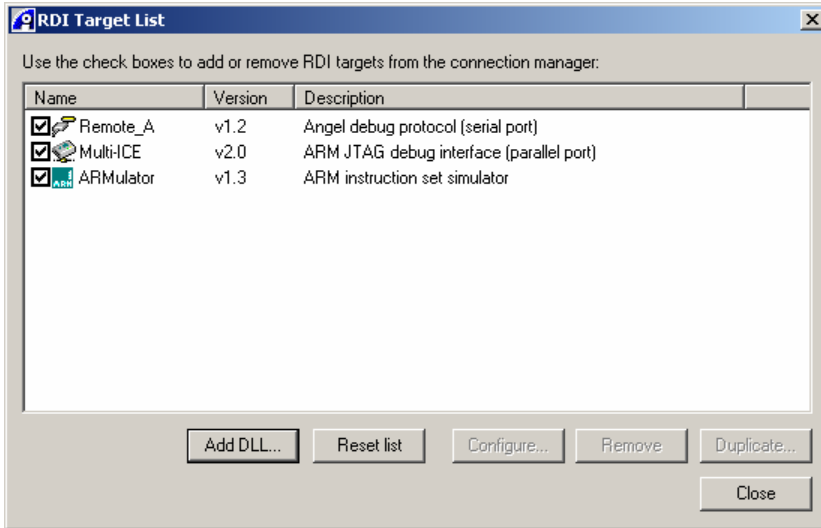


FIGURE 5 The RDI Target List dialog box.

4. Enter a name and description for the new target connection (Figure 6). Click OK.

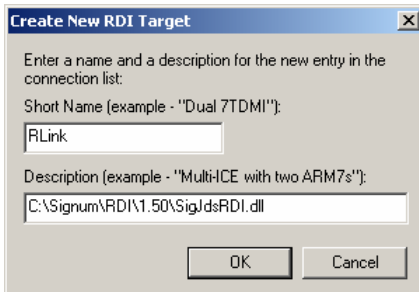


FIGURE 6 Naming and describing the new target connection.

5. Make sure that the new target entry appears in the RDI target list (Figure 7) and close the RDI Target List dialog box.

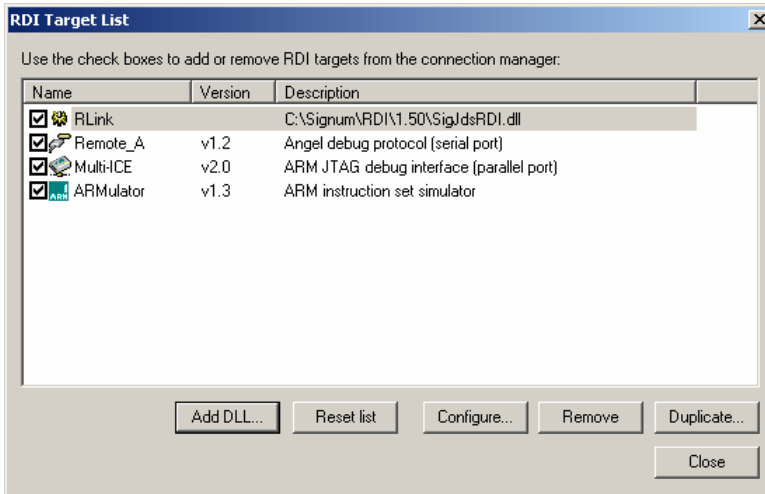


FIGURE 7 The augmented target list.

6. In the Connection Control dialog box, right-click the new connection entry and select Configure Device Info from the popup menu (Figure 8).

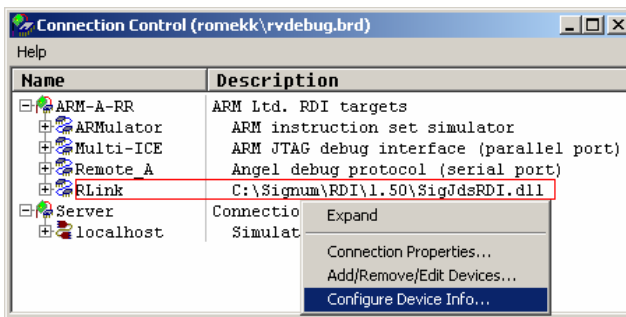


FIGURE 8 The augmented target list.

When the RDI Configuration dialog box appears, proceed to section *Configuring the Connection* on p. 18 to complete the system configuration process.

ARM AXD, SDT and ADS Debuggers

The ARM RDI driver for Signum emulators is compatible with all the ARM debuggers that conform to the RDI specification, including:

- ARM extended Debugger (AXD) for Windows,
- ARM Debugger for Windows SDT 2.51 and higher,
- ARM Debugger for Windows ADS v1.0.1 and higher.

The following debugger configuration procedure uses the AXD debugger as an example. In most situations, extending this procedure to other debuggers will be straightforward.

1. Run the ARM debugger.
2. From the debugger's Options menu, select Configure Target. The Choose Target dialog appears.

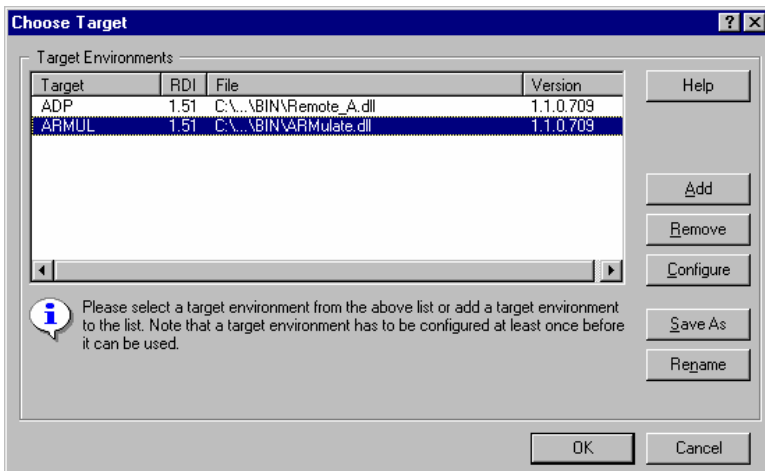


FIGURE 9 Selecting the driver DLL.

3. Press the Add button. In the Windows Open dialog, navigate to the ARM RDI driver file. If you accepted the defaults from the Installing the Driver section, the path to the driver file should be C:\Signum\RDI\SigJdsRDI.dll. Click the Open button. The driver is now listed in the Choose Target dialog box.

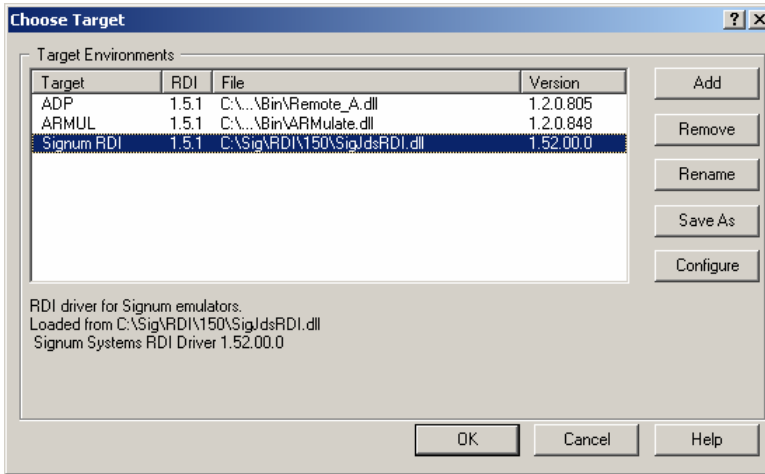


FIGURE 10 The RDI driver for the Signum emulator is now available.

Press the Configure button to configure the connection between your debugger and the Signum emulator. Proceed to section *Configuring the Connection* on p. 18.

GreenHills Multi2000 Debugger

► To configure the GreenHills Multi2000 Debugger:

1. Start the MULTI debugger. In the Target menu, select Show Connection Organizer (Figure 11).

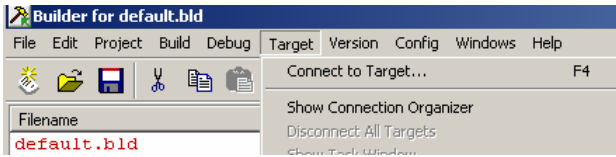


FIGURE 11 Activating the Connection Organizer.

2. In the Connection Organizer dialog box (Figure 12), select Open from the File menu and navigate to the RLink.con file. In a standard installation, the file is found in the C:\Signum\RDI\Config\Multi directory.

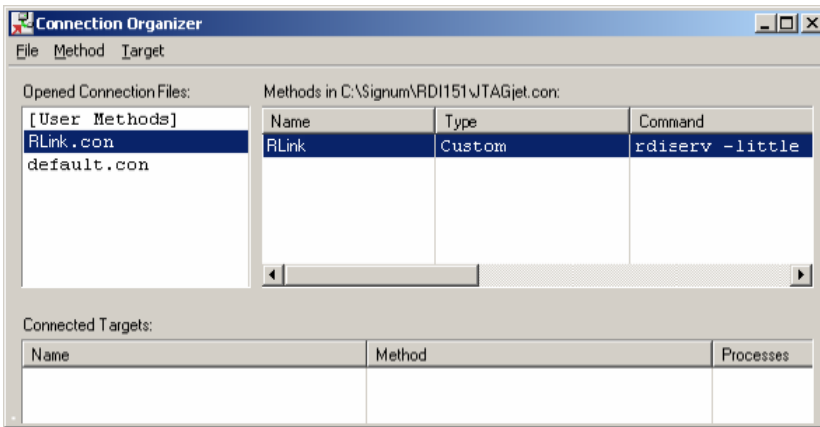


FIGURE 12 Choosing the RLink.con file in the Connection Organizer.

3. Right-click in the dialog box. From the popup menu, select Copy. Choose [User Methods] from the drop-down list Copy Selected Connections To. Click OK. Close the Connection Organizer dialog box.
4. In the Target menu, press Connect To Target. Choose the RLink target and press Connect. RDI Configuration dialog appears.

Proceed to section *Configuring the Connection* on p. 18. MULTI will display a server timeout message if the RDI Configuration dialog remains opened

longer than the server timeout period. Dismiss this message after completing the RDI Configuration process.

The next time you start the debugger, directly select Connect To Target from the Target menu to connect to the RLink target.

IAR Embedded Workbench

► To configure the IAR Embedded Workbench:

1. Choose Options from the Project menu. The Options for Target dialog box appears. Select the Debugger category. In the Setup tab, choose RDI as the driver, as in Figure 13.

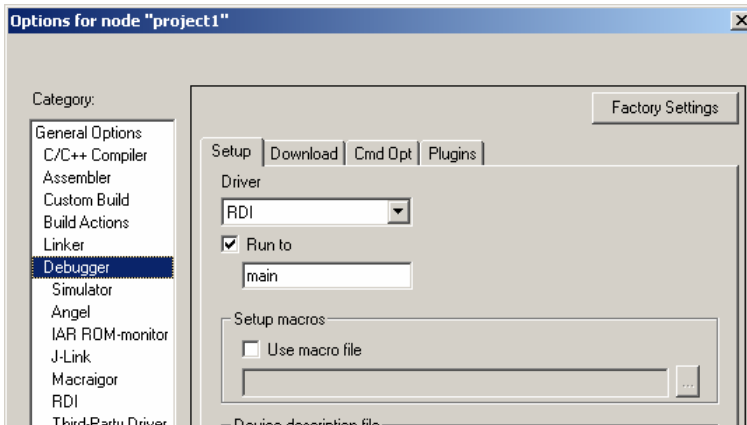


FIGURE 13 Selecting the driver type for C-Spy.

2. Select the RDI tab. In the Manufacturer RDI Driver edit box, navigate to the SigJdsRDI.dll file in the C:\Signum\RDI directory, as in Figure 14.

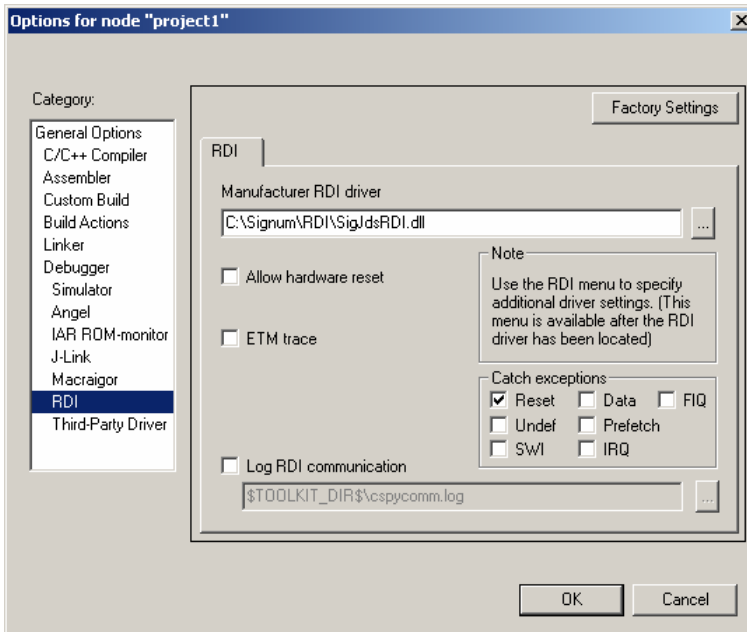


FIGURE 14 Selecting the RDI driver DLL file for C-Spy.

Press OK to close the dialog box.

In the RDI menu, select Configure to open the RDI Configuration dialog. Proceed to section *Configuring the Connection* on p. 18.

Mentor Graphics XRAY Debugger

► To configure the Mentor Graphics XRAY Debugger:

1. Using a plain text editor, such as Notepad, append the Signum RDI driver configuration record to the XRAY Debugger board file %XRAY_HOME%\etc\aro.brd (in default installations, the file is c:\MGC\embedded\xrayose\etc\aro.brd):

```
+ RLink 0 "Raisonance RLink" "RDI 1.5" \  
{ type=multi-ice:endian=little:rdilib=  
  "C:\Signum\RDI\SigJdsRDI.dll" }
```

To avoid mistakes, we suggest you cut and paste this line from the C:\Signum\RDI\Config\XRAY\signum_aro.brd.

The leading + character instructs the debugger to connect automatically to the emulator. Make sure that the aro.brd file contains no more than one entry preceded by +.

2. Start the XRAY OSE for RDI (ARM Freezemode)

The debugger attempts to connect to the emulator. If the emulator has not yet been configured, the RDI Configuration dialog appears. Proceed to section *Configuring the Connection* on p. 18.

If the debugger does not connect to the emulator automatically, select Connection Manager from the Managers menu to choose RLink as the connection target. In the Manager dialog box, press Connect.

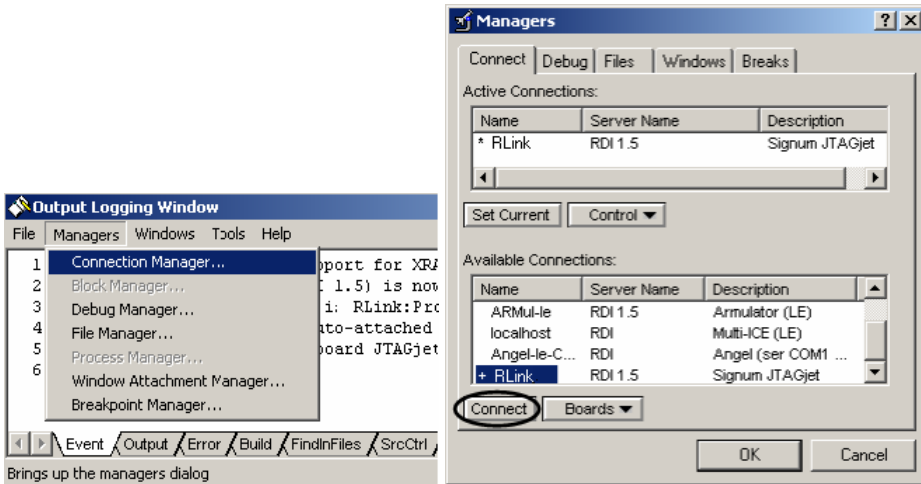


FIGURE 15 Setting up the connection between XRAY and RLink manually. The RLink RDI 1.5 server should appear in the Connect tab.

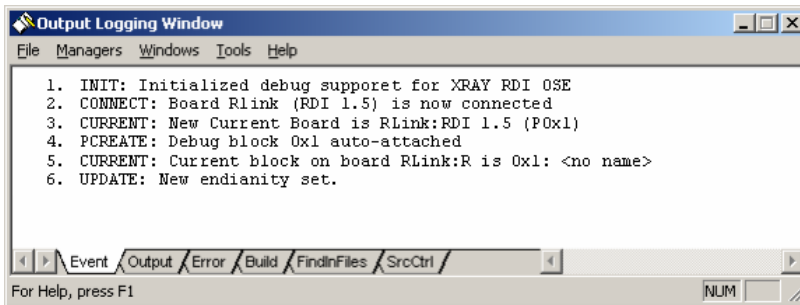


FIGURE 16 A log of a successful connection established between the target and the emulator.

Note: To modify the emulator configuration at a later time, make sure that the debugger is not running. Press the system Start button and select Signum Systems RDI from the Programs (or All Programs under Windows XP) menu. This will execute the RDIconfig configuration program.

Palm Universal Debugger (PUD)

► To configure the Palm Universal Debugger:

1. Select Preferences from the Edit menu. The Debugger Preferences dialog box appears.
2. In the Category pane, select ARM under the Debugger Plugins entry. In the Debugger Plugins/ARM Settings pane, select ARM RDI1.5 Protocol Plugin as the protocol. See Figure 17.

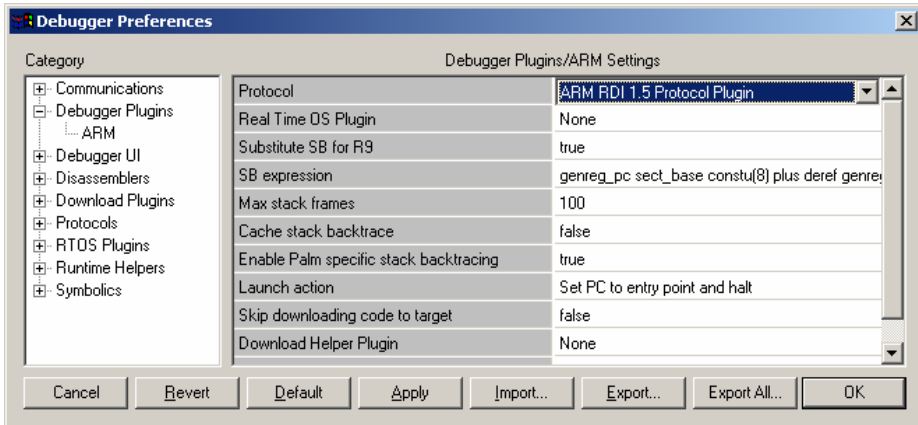


FIGURE 17 Selecting the protocol for the ARM Debugger Plugin.

3. In the Category pane, select ARM RDI 1.5 under the Protocols entry. In the Debugger Plugins/ARM Settings pane, navigate to the SigJdsRDI.dll file to set the Path to RDI DLL entry. In a standard installation, the SigJdsRDI.dll file is located in the C:\Signum\RDI folder. See Figure 18.

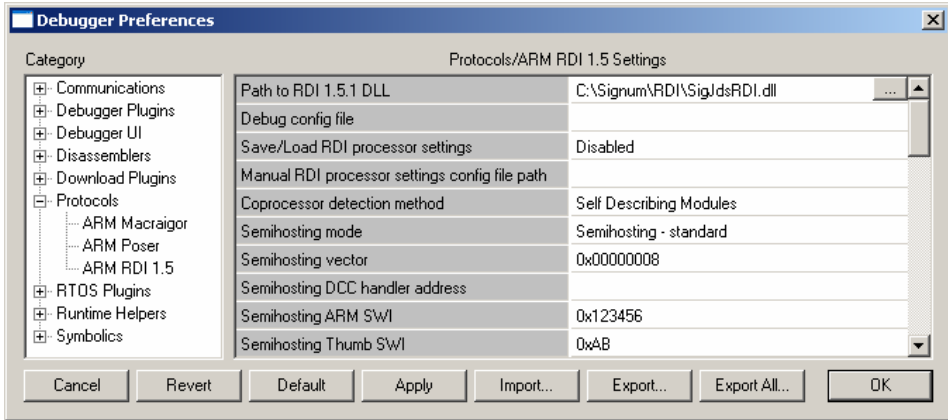


FIGURE 18 Selecting the path to the RDI 1.5 DLL file for the ARM RDI 1.5 protocol.

4. Select Connect in the menu Target.

The debugger attempts to connect to the emulator. If the RLink has not yet been configured, the RDI Configuration dialog appears. Proceed to section *Configuring the Connection* on p. 18.

GNU gdb Debugger

The GNU gdb debugger connects to the emulator through the Signum GDB Server. The server must run on an MS Windows computer, whereas the debugger can run on a local computer (the same as the PC on which the server executes) or on a remote computer under either Linux/UNIX or DOS/Windows operating system.

► To connect the gdb debugger to the emulator via the GDB Server:

Execute the GDBserver.bat file found in the RDI driver folder to start the server. You may want to modify the `--remote-port` parameter in GDBserver.bat order to select the desired communication port. Note that when running the server for the first time, it is necessary to use the `-config-`

Note: The server folder contains the `cygwin1.dll` library file. You may want to rename this file if it is in conflict with your version of `cygwin1.dll`.

dialog option that invokes the RLink configuration dialog box. Once configured, the server can be run without the `--config-dialog` option. Proceed to section *Configuring the Connection* on p. 18

to set up the connection parameters, and then complete this procedure.

5. Start the gdb debugger. To establish connection with the GDBserver, run “target remote <host>:<port>” command, as shown in the following examples.
 - To connect to the server from a remote host computer named “celeronti” via port 9000, enter:

```
target remote celeronti:9000
```

- To connect to the server on the local machine, enter:

```
target remote localhost:9000
```

As another example, the beginning of a remote debug session might look like this:

```
$ arm-elf-gdb x.elf
GNU gdb 5.3
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General
Public License, and you are
welcome to change it and/or distribute copies of it
under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type
"show warranty" for details.
This GDB was configured as "--host=i686-pc-cygwin -
-target=arm-elf"...
(gdb) target remote celeronti:9000
Remote debugging using celeronti:9000
main () at x.c:34
34      a = 0;
```

SIGNAL SYSTEMS

```
(gdb) load x.elf
Loading section .text, size 0x1f10 lma 0x8000
Loading section .rodata, size 0x29 lma 0x9f10
Loading section .data, size 0x850 lma 0xa03c
Loading section .ctors, size 0x8 lma 0xa88c
Loading section .dtors, size 0x8 lma 0xa894
Start address 0x8000, load size 10137
Transfer rate: 81096 bits in <1 sec, 151
bytes/write.
(gdb) p /x $pc
$1 = 0x8000
(gdb) break main
Breakpoint 1 at 0x8220: file x.c, line 34.
(gdb) c
Continuing.

Breakpoint 1, main () at x.c:34
34          a = 0;
.
.
.
And so on
```

Metrowerks CodeWarrior

► To configure the Metrowerks CodeWarrior debugger:

1. Select Preferences from the Edit menu. The IDE Preferences dialog appears (Figure 19). Press the Add button to add (create) a new connection.

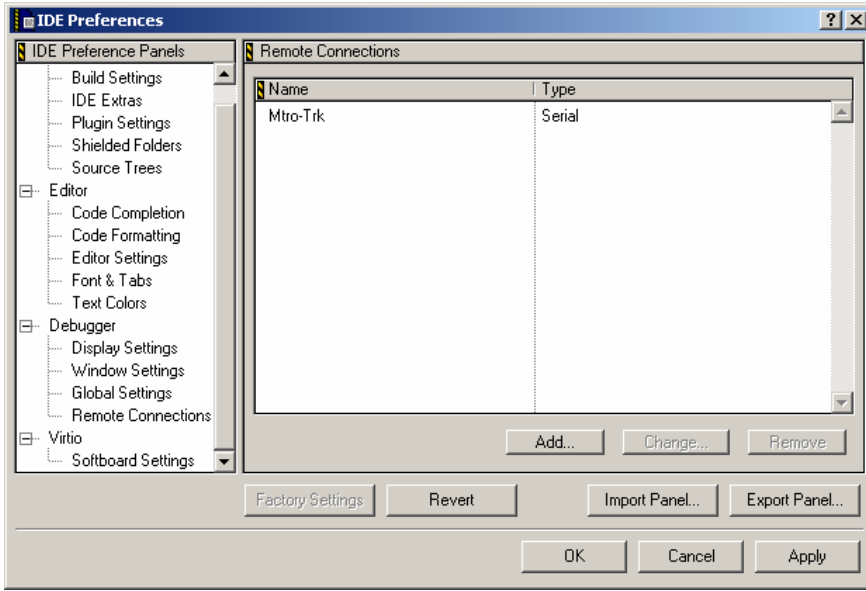


FIGURE 19 The IDE Preferences dialog box.

2. In the New Connection dialog box (Figure 20):
 - Choose a name for your connection, e.g., RLink.
 - Select ARM RDI from the Debugger drop-down list.
 - Make sure that the Connection Type is set to RDI.
 - Use the Browse button to locate the SigJdsRDI.dll file.
 - Select Configure to open the RDI Configuration dialog and proceed to section *Configuring the Connection* further in the text.

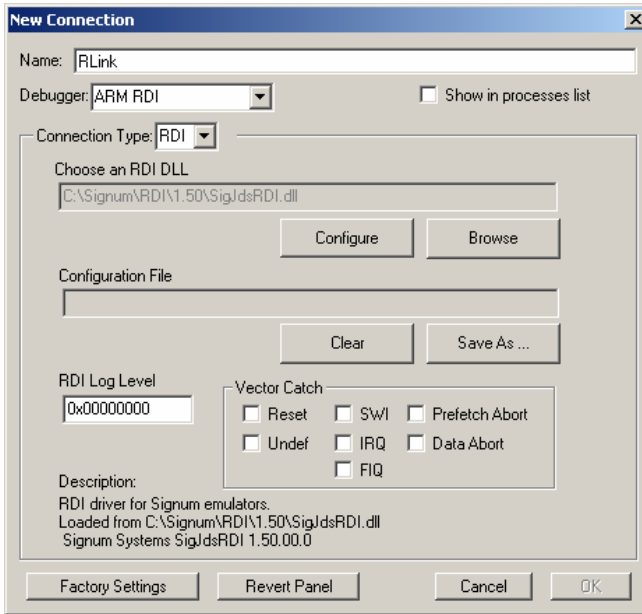


FIGURE 20 The IDE New Connection dialog box.

Configuring the Connection

1. In the Configuration dialog that appears, make sure that the Connection tab is selected.

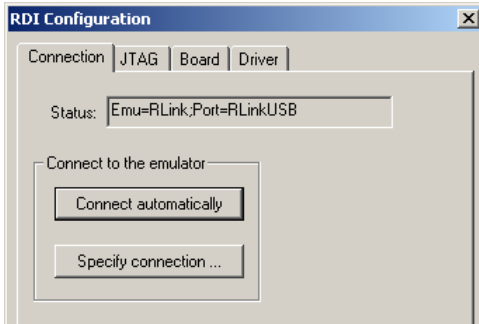


FIGURE 21 Configuring the emulator: the Connection tab.

- To establish a debugger-emulator connection with minimum effort, press the Connect Automatically button. The connection parameters appear in the Status box.
 - For added control over the connection, press the Specify Connection button to set up the communication port and its parameters manually.
2. Once the connection has been established, select the JTAG tab in the JTAG configuration dialog. Choose your target device from the CPU drop-down list as follows:
 - From the CPU list, select the Single ARM7TDMI core option. The name (ID) and device type, along with a short description, are displayed in the JTAG Chain group box (Figure 22):

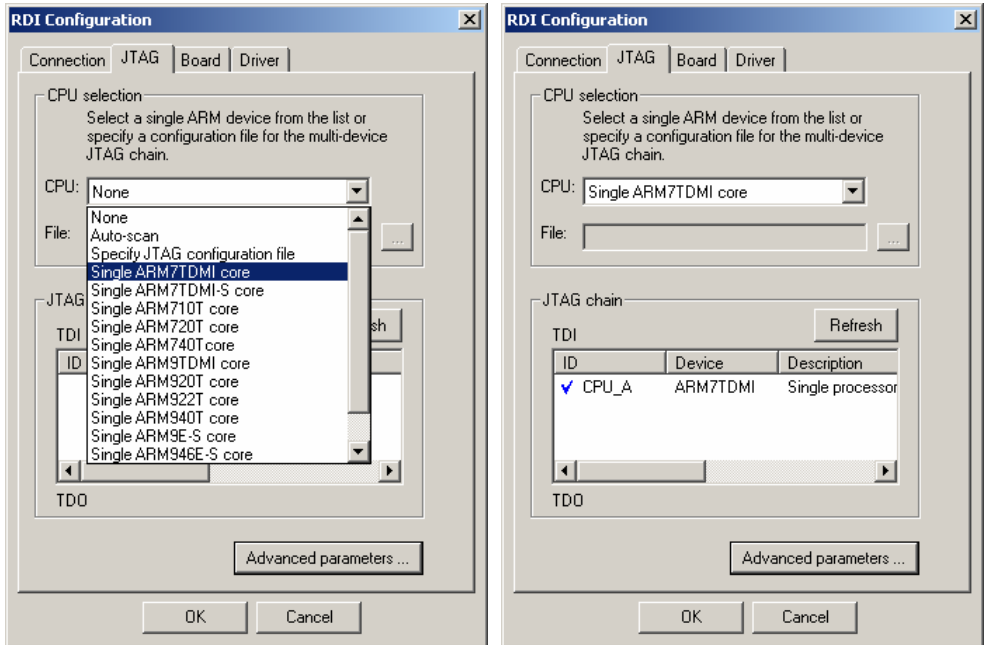


FIGURE 22 Selecting a single target CPU.

- Make sure that in the JTAG Advanced Parameters dialog box, the Emulation Mode is set to ARM (standard), while the JTAG clock field is either empty or displays the correct clock frequency (Figure 22).

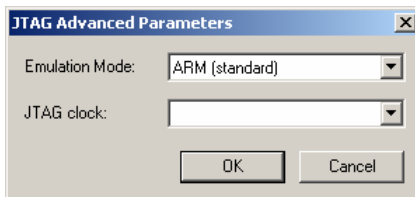


FIGURE 23 Selecting the emulation mode and JTAG clock frequency.

- Click the Board tab and select the endianness of your target board (Figure 24), and then browse for a board startup macro file or leave the File field empty.

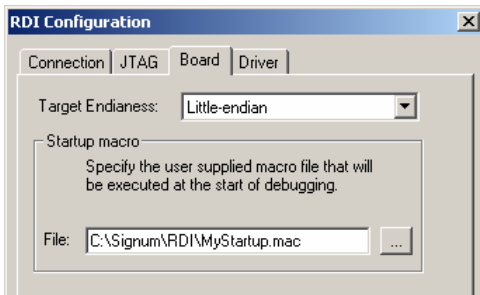


FIGURE 24 Configuring the emulator: the Board tab.

- Click the Driver tab to configure driver protocol logs and error reports.
 - To display the driver protocol log in the Log window, select Log Enable. If you also need to store the log in a file, enter the file name in the File text box.
 - To enable the driver to generate descriptive error messages, select Show Error Messages. This option does not affect the way your debugger displays its own error messages; it is designed simply to augment and clarify those debugger messages that tend to be cryptic or are limited to error codes only.

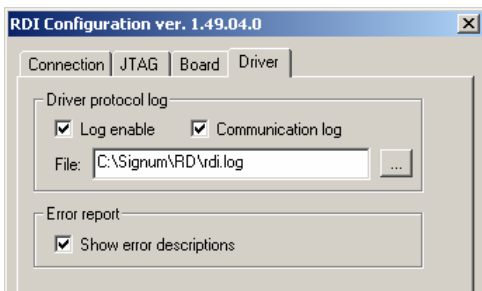


FIGURE 25 Configuring the emulator: the Driver tab.

5. Click OK. In the Choose Target dialog box, click OK again to accept your settings and connect to the target processor.

Multi Device Configuration

Note: JTAG chain with a single device in it does not require any configuration file.

A multi-device JTAG daisy chain is configured using an ASCII text JTAG chain configuration file with extension `.cfg`. With the exception of comment lines, each line in a `.cfg` file refers to a separate device. Thus in general, the file format is as follows:

```
ID1   Device1   Description1
ID2   Device2   Description2
.
.
.
etc.,
```

where

<i>ID</i>	Is a unique name identifying the device, e.g., "CPU_1", including the double quotes.
<i>Device</i>	Is the type of the device, such as ARM7TDMI. Non-ARM devices should be bypassed, and specified as BYPASS <code>xx</code> , where <code>xx</code> denotes the length of the instruction register in a two-digit format. For instance, BYPASS0A denotes a bypassed device with a 10-bit (0a hex) instruction register.
<i>Description</i>	Is a comment text.

In general, enclose in double quotes all names, words or phrases that contain non-alphanumeric characters. Lines that begin with the semicolon (;) are treated as comments and are ignored. The order in which the JTAG devices are specified

in the configuration file is significant: the first line corresponds to the device closest to the TDI, the second one to the next device in the chain, and so on. Finally, the last line describes the device on the TDO side of the chain.

Board Startup Macro File

Some application boards may require to be properly set up before a debug session can begin. Use a startup macro to enable or configure on-board memory before the debugger attempts to access that memory. This macro is also a good place for disabling the watchdog that otherwise may reset the CPU soon after the debugger starts running.

Table 1 lists the commands that can be executed from within the board's ASCII .mac strtp file.

sd <address> = <value>	Write a DWORD (32bit) <value> to memory <address>.
sw <address> = <value>	Write a WORD (16bit) <value> to memory <address>.
sb <address> = <value>	Write a BYTE (8bit) <value> to memory <address>.
dd <address>	Read a DWORD (32bit) from memory <address>.
dw <address>	Read a WORD (16bit) from memory <address>.
db <address>	Read a BYTE (8bit) from memory <address>.
reset /halt	Reset the CPU. The /halt option stops the CPU after the reset.

pause <msec>	Pause for the <msec> number of milliseconds.
emu <parameter> = <value>	Set one of the following emulation parameters: <ul style="list-style-type: none"> • cmdline (argument string for the debugger) • semihosting_enabled (0 – disabled, 1 – enabled)

TABLE 1 Configuration file commands.

Empty lines, spaces and comments starting with a semicolon are ignored.

The following is an example of a typical startup macro file.

```

; MyTarget.mac - RDI Startup macro for MyTarget board.
; Version 1.00 7/10/02 - Initial version

reset /halt                ; reset and stop the CPU
sd 0xFFFFE4000 = 0x032F0102 ; Setup PLL register
pause 100                  ; Wait for clock setup

; Initialize memory
sd 0xFFFECC10 = 0x00203339 ; CS0 configuration
sd 0xFFFECC14 = 0x00001139 ; CS1 configuration
sd 0xFFFECC18 = 0x00001139 ; CS2 configuration
sd 0xFFFECC1C = 0x00001139 ; CS3 configuration

; Disable ARM watchdog
sw 0xFFFECC808 = 0x00F5
sw 0xFFFECC808 = 0x00A0

; Configure the emulator
emu semihosting_enabled=0 ; disable virtual I/O

; End of file 'MyTarget.mac'

```

