Raza Microelectronics, Inc.

**User Manual**

# RMIOS IDE - User Manual

## Preliminary

## Rev 0.05

Dec 2006

Document # 2032UM

Raza Microelectronics, Inc.
18920 Forge Drive
Cupertino, CA 95014
U.S.A.
www.Razamicro.com

Tel: (408) 434 5700
Fax: (408) 434 5777

## Revision History

| Version | Date | Description of Changes Since Last Revision |
|---|---|---|
| 0.01 | 06/01/2006 | Initial version - Subject to changes. |
| 0.02 | 06/01/2006 | Added the environment variable “GDBDEBUGGER” |
| 0.03 | 07/05/2006 | Added changes to the installation wizard. Also, incorporated the changes in IDE. |
| 0.04 | 11/08/2006 | Added a new section for building RMIOS Library. |
| 0.05 | 12/15/2006 | Made changes to the section “Debugging RMIOS Applications.” |

## Related Documents

- XLR™ Processor Family Data Sheet- 2DS001xxx
- XLR™ Processor Programmer’s Manual- 2UM001xxx
- GDB -
    - Richard M. Stallman, Roland Pesch, Stan Shebs, et al., Debugging with GDB (Free Software Foundation, 2002) ISBN 1882114884
    - Norman Matloff, P. J. Salzman, The Art of Debugging with GDB/DDD: For Professionals and Students (No Starch Press, 2003) ISBN 159327002X
- MIPS™ Technologies, "MIPS64™ Architecture for Programmers"
    - Volume I: Introduction to the MIPS64 Architecture - Document Number: MD00083
    - Volume II: The MIPS64 Instruction Set - Document Number: MD00087
    - Volume III: The MIPS64 Privileged Resource Architecture - Document Number: MD00091

# Copyright Information

# Table of Contents

# List of Figures

# 1 Introduction

This guide describes step by step instructions for installing “RMIOS SDK” and using it effectively along with IDEs to develop RMIOS applications.

**Note: In order to install and use this software, you must have a fully executed and valid “RMI Software License Agreement.”**

Updates to the document and its associated RMIOS releases are available via your private RMI ftp site.

## 1.1 What is RMIOS SDK?

RMIOS SDK is a custom software development kit, which consists of the following components:

- BINUTILS-2.15, GCC-3.4.3 and NEWLIB-1.13 - Cross compiler and toolchain, which can be used for firmware builds (Bootloader, RMIOS-applications). This is created for 64-bit ELF targets (default MIPS ABI=O64). It also includes C, C++ front-ends and the target (XLR) specific patches for binutils, gcc and newlib.
- Libraries and Source codes for the libraries - precompiled RMIOS libraries, header files and required scripts are provided as part of the release.

  **Note: If you are not enhancing the functionality of RMIOS, then there is no need for compiling RMIOS source code.**
- Cygwin tools and libraries - certain Cygwin tools and libraries, which are required by the toolchain for building RMIOS applications.
- Reference applications - RMIOS reference applications.

  **Note: RMIOS library and applications are two distinct modules.**
- GDB/Insight debugger and Insight launcher - used to debug RMIOS applications.

## 1.2 How to build RMIOS applications?

RMIOS applications are built using the “Software Development Tool Kit” along with any of the custom IDEs. In this document we have illustrated examples, setup and configurations using Eclipse IDE.

This document explains the process of building/compiling RMIOS applications in the following sections:

- Installing RMI SDK Toolchain for Windows” on page 11
- Installing and configuring Eclipse for RMIOS Development” on page 19
- Creating a New Project/Building a sample application” on page 21
- Debugging RMIOS Applications” on page 31
- Building a Custom RMIOS library” on page 39

These sections are detailed in the subsequent sections of this guide.

# 2 Installing RMI SDK Toolchain for Windows

This chapter provides the complete details of the steps that are to be performed for installing and setting up the RMIOS Software Development Kit.

## 2.1 The Installation Process

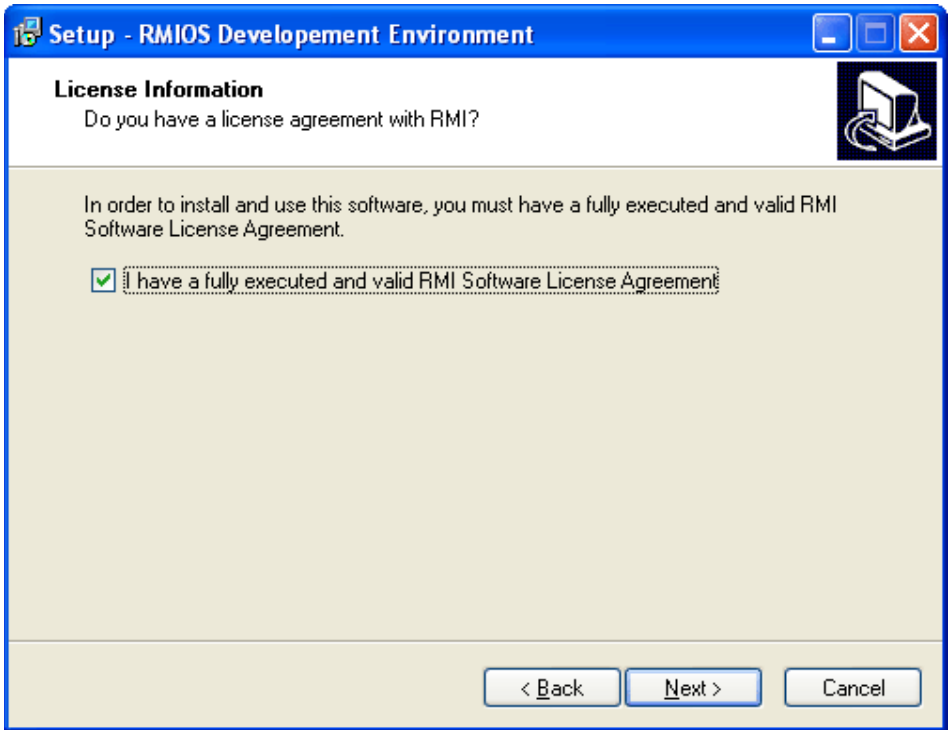
Insert the CD-ROM in the CD drive and run the file SETUP.EXE.

When you execute this file, the installation wizard prepares the system for the installation and a ‘Welcome dialog’ window is displayed as shown below.

**Figure 1: RMIOS Development Tool Kit Welcome Window**


Click the ‘Next’ button to proceed with the installation. The ‘License Information’ window is invoked.

**Figure 2: License Information Window**

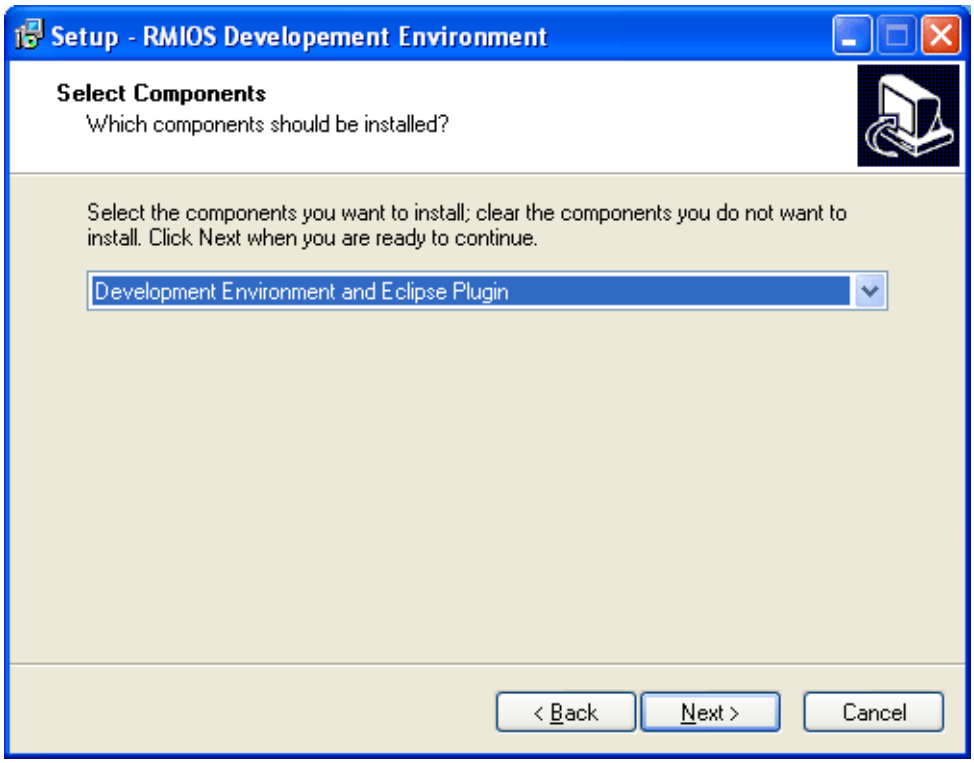



In the 'License Information' window, select the check box "I have fully executed and valid RMI Software License Agreement" to proceed with the installation. The "Select Destination Location" window is invoked as shown below.

**Figure 3: Select Destination Location window**

In this window, you can browse and select the folder wherein the RMIOS development toolkit will be installed. The default location for installation is C:\rmidev. Click the “Next” button to invoke the following window.

**Figure 4: Select Components Window**



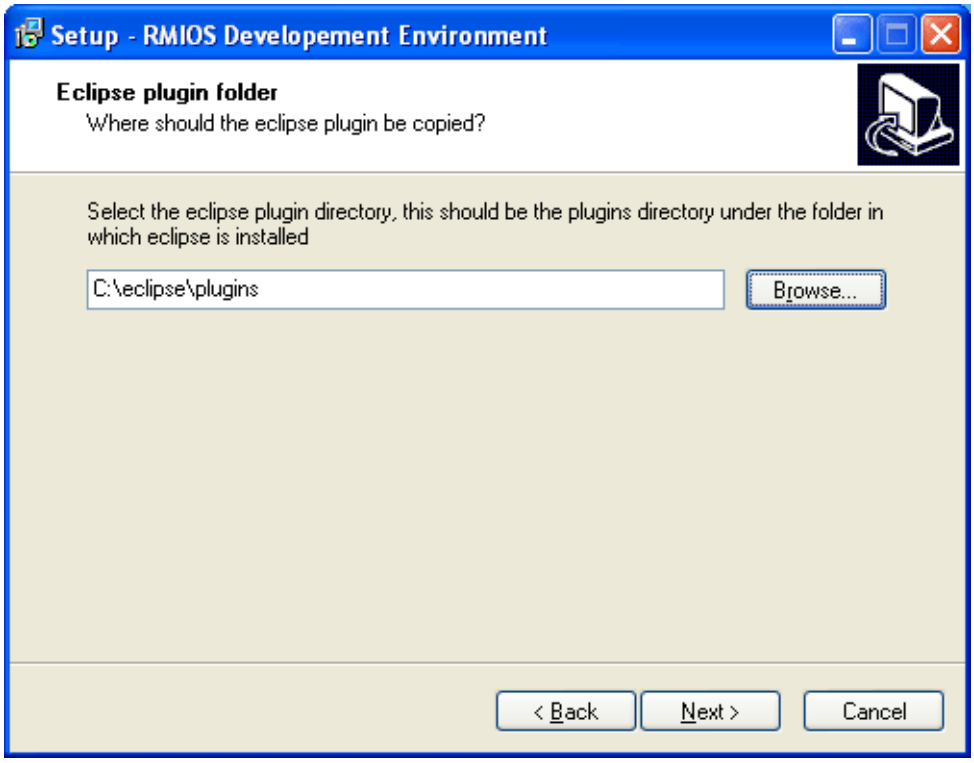

In the “Select Components” window, select the components that are to be installed with the SDK. You are provided with the following options:

- Development Environment and Eclipse Plugin: This option installs the Eclipse plug-in along with RMIOS SDK. It is recommended to select this option if you are planning to use “Eclipse” as the IDE.

- Development Environment: This option installs only the RMIOS SDK. You can choose this option if you intend to use any IDE other than Eclipse.

Click the “Next” button to invoke the following window.

**Figure 5: Eclipse Plugin folder window**

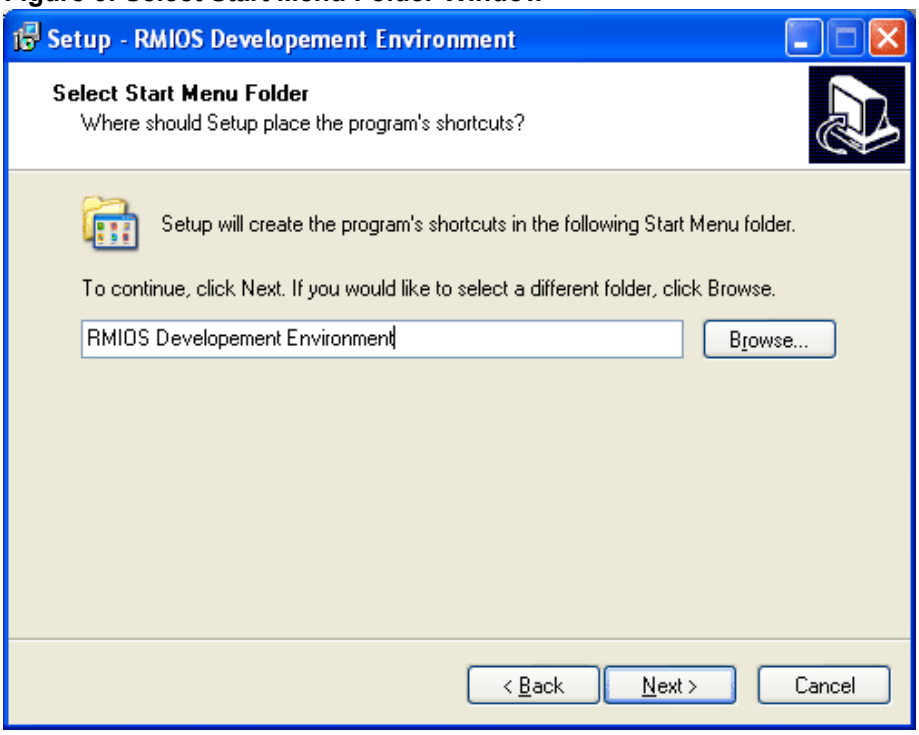



In the “Eclipse plugin folder” window, select the directory for plug-in, in the Eclipse installation folder. If the default settings are used, this is C:\eclipse\plugins.

**Note: This dialog is displayed only when the option “Development Environment and Eclipse Plugin” is selected in the “Select Components” window.**

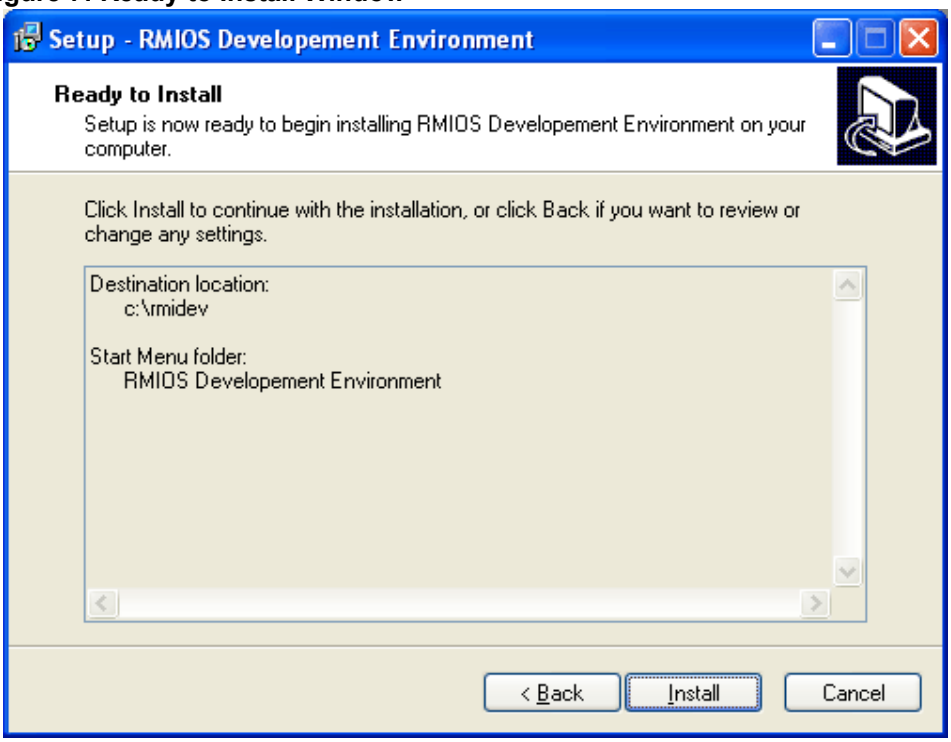
Click the “Next” button to invoke the following window.

**Figure 6: Select Start Menu Folder Window**



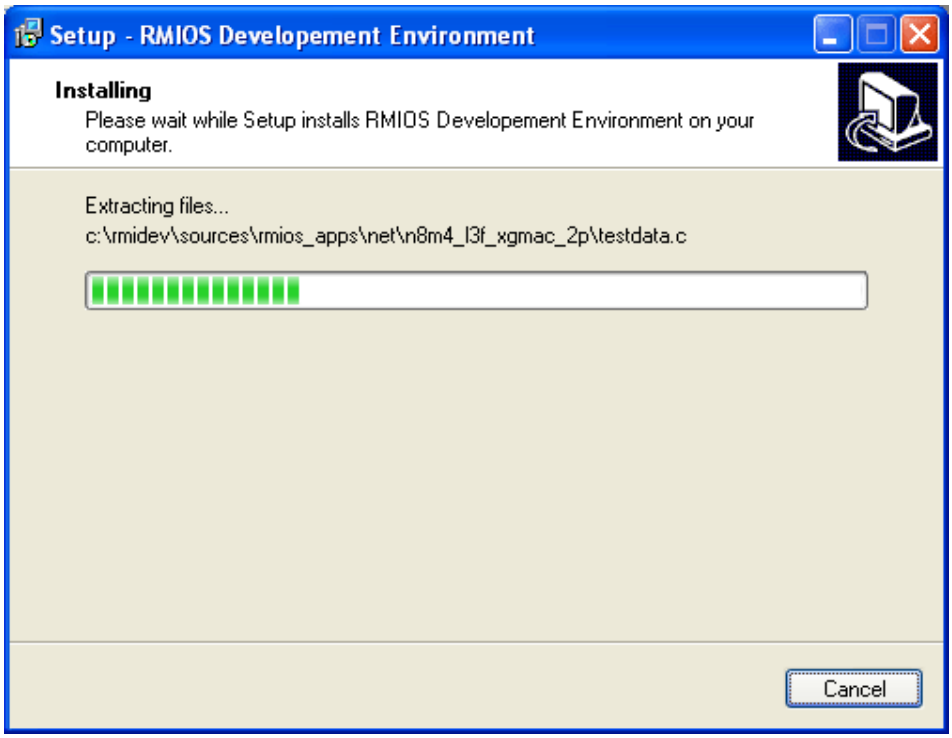

The “Select Start Menu Folder” window gives you the choice to select a folder for placing the programme’s shortcuts. “RMIOS Development Environment” is the default option. Click the ‘Next’ button to proceed with the installation and invoke the following window.

**Figure 7: Ready to Install Window**



The “Ready to Install” window indicates the initiation of the installation. Click the ‘Install’ button to proceed with the installation. The ‘Installation Status’ window is invoked as shown below.

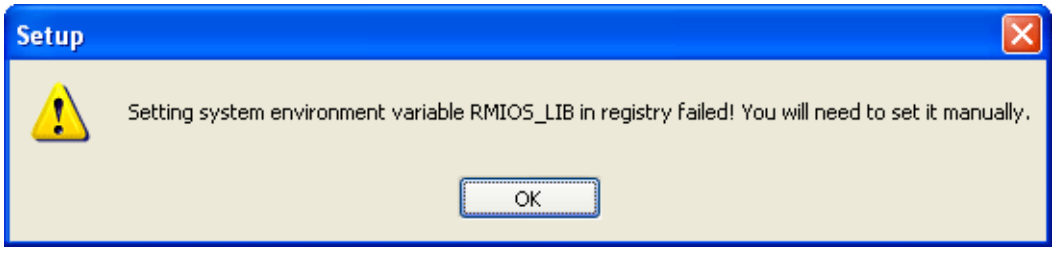
**Figure 8: Installation Status window**




You can terminate the installation at any point by clicking the “Cancel” button.

**Note: if you are installing the tool kit in a regular user profile, you will get the following window. It displays a warning message stating that “Setting System environment variable RMIOS_LIB in registry failed! You will need to set it manually.” Click the OK button, the installation will get completed. Refer to the section “Creating a New Project/Building a sample application” on page 21 for setting up this variable.**

**Figure 9: System environment error message.**




Once the installation is complete, the “Installation Complete” window is displayed as shown below.

**Figure 10: Installation Complete Window**


Click the “Finish” button. The short cut to this program is added in Start -> Programs menu of your windows system.

**Note: Log out and login again to this system in order to set all the environment variables.**

# 3 Installing and configuring Eclipse for RMIOS Development

Download and install the latest version of Eclipse and CDT from http://www.eclipse.org/.

Eclipse[1] is an open source platform-independent software framework, whose projects provide development platform and application frameworks for building software applications.

The CDT (C/C++ Development Tools) package provides a fully functional C and C++ Integrated Development Environment (IDE) for the Eclipse platform.

**Note: Eclipse/CDT is the recommended software framework for building RMIOS applications. However there are various other IDEs that can be customized and tweaked to develop RMIOS applications.**

**Note: You should have the latest version of Java Development Kit installed on your system. If you do not have this kit, you can download from http://java.sun.com/javase/downloads/index.html and install the same.**

Once the installation is complete, you can open Eclipse and compile RMIOS applications.

1. Refer to link http://www.eclipse.org/org/ for more information on origin and history of Eclipse software.

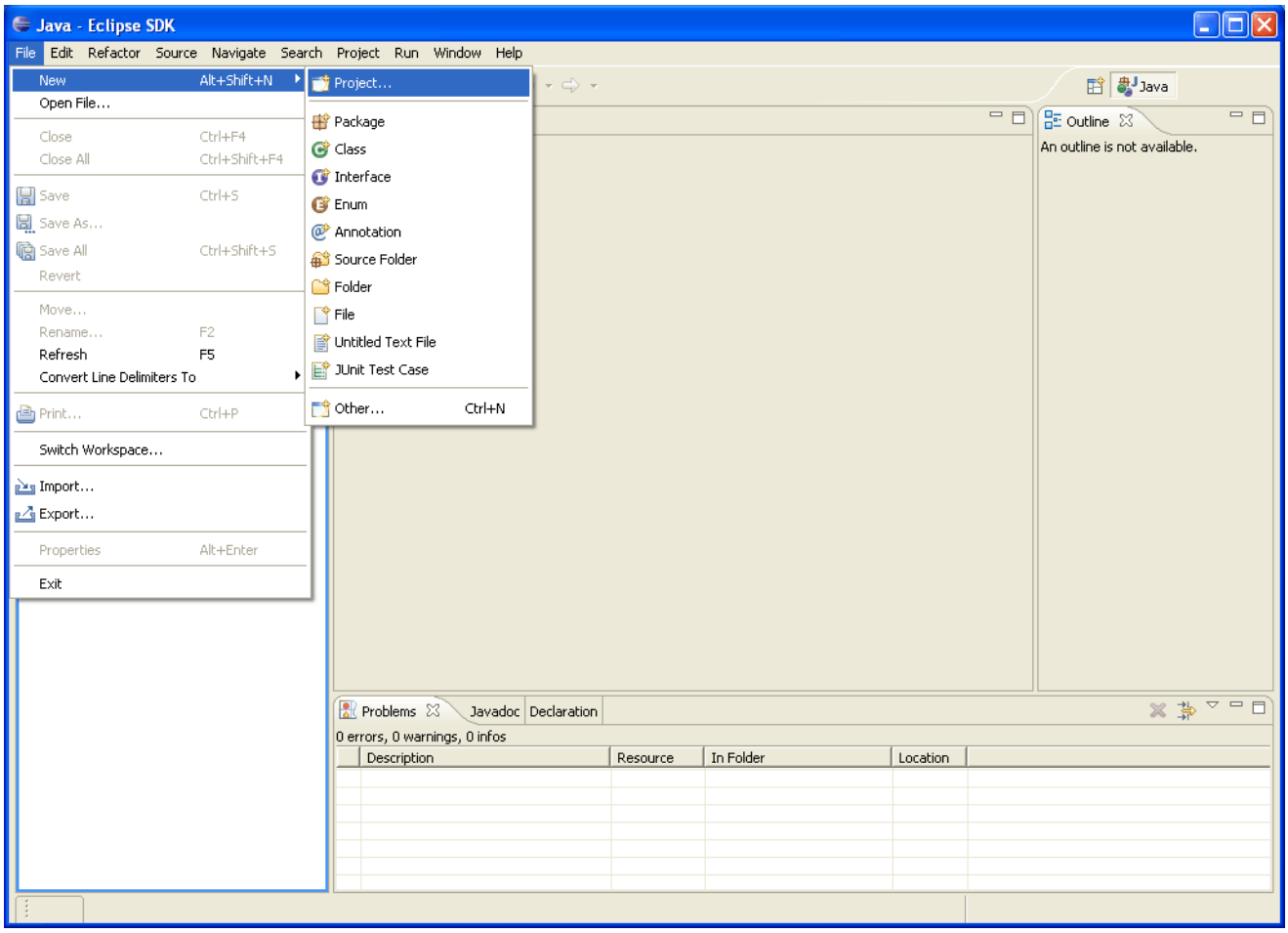
# 4 Creating a New Project/Building a sample application

This chapter explains how to build applications using RMIOS SDK with Eclipse IDE. It also details the essential settings required for building these applications.

## Creating a New Project

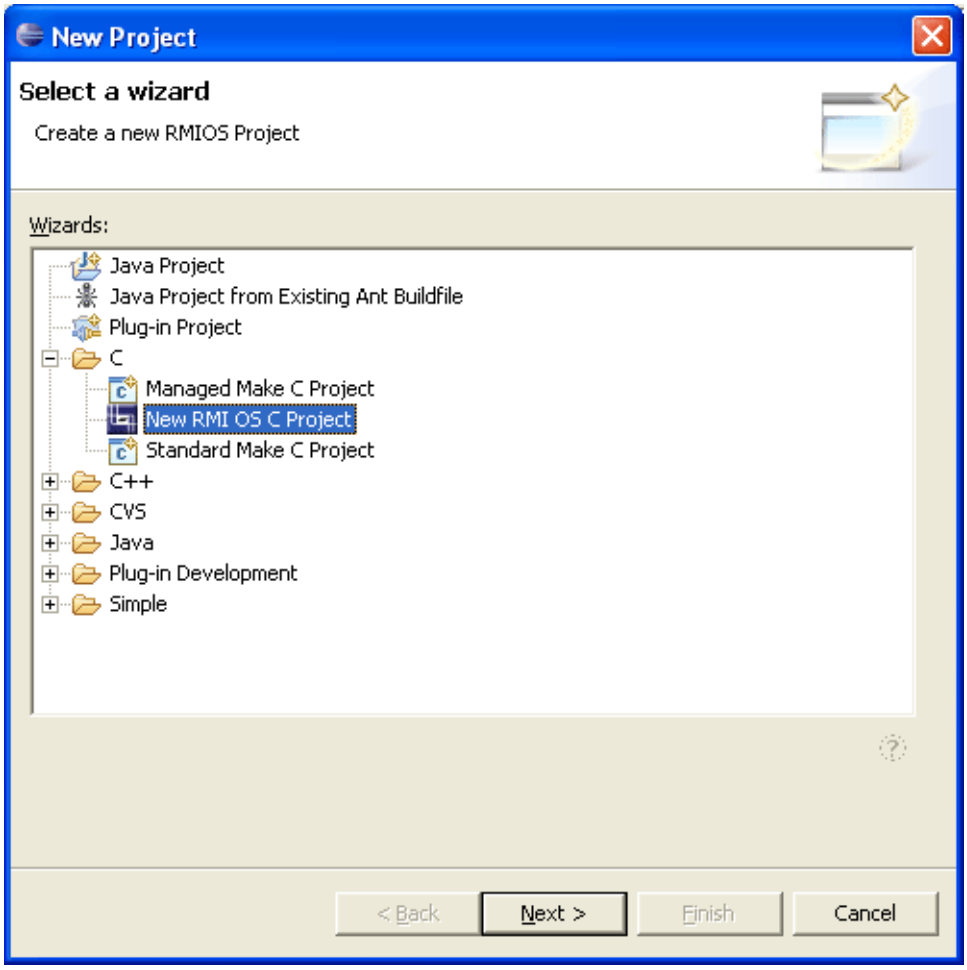
In the Eclipse software, open the menu item **File ⏩ New ⏩ Project**, to create a new C project.

**Figure 1: Eclipse SDK Window**




The “New Project” Wizard is invoked as shown below.

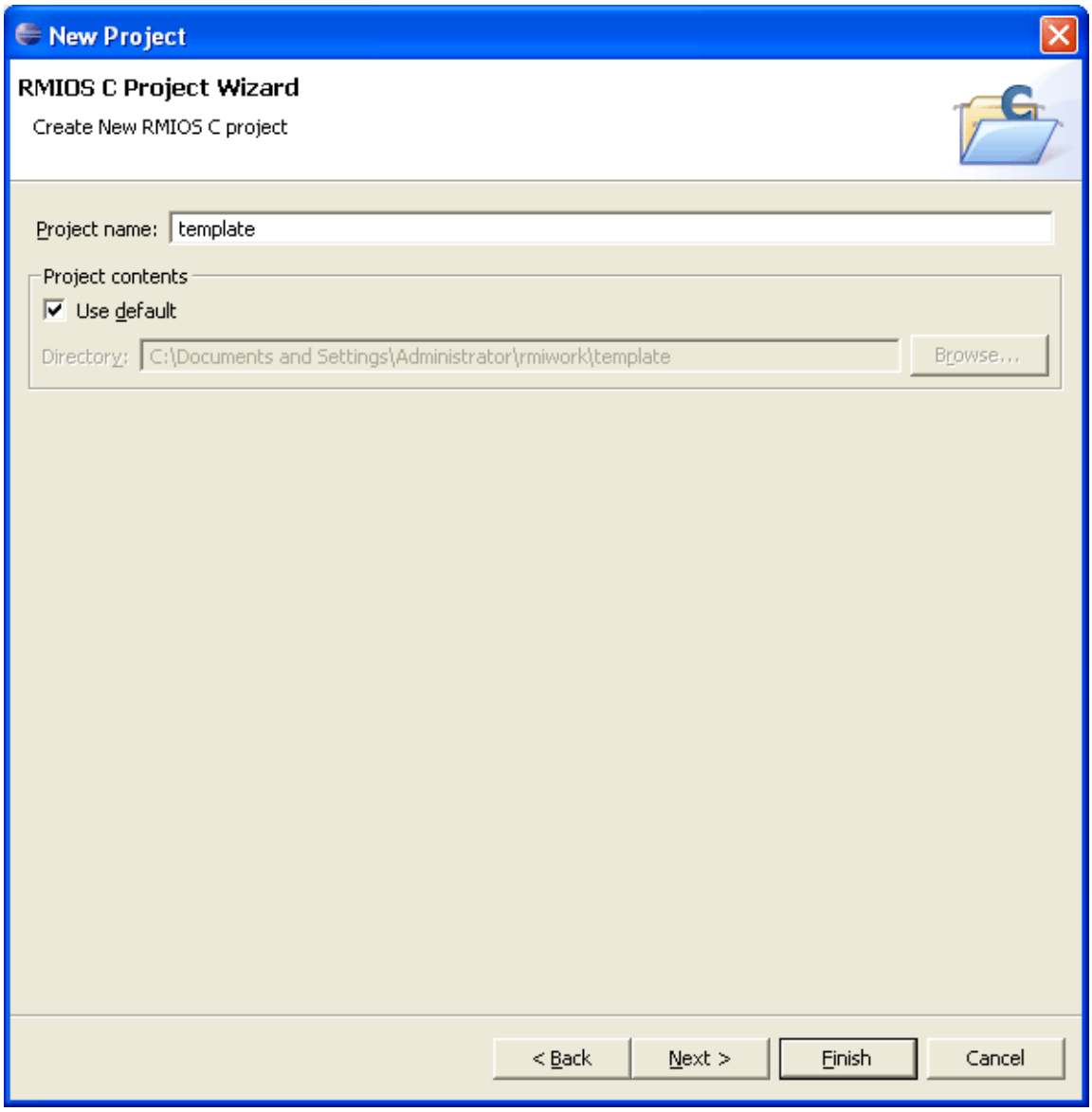
**Figure 2: New Project Window**




In the “Select a wizard” window, choose the option “New RMIOS C project”.

The first window displayed in this wizard is the “Make Project” window, which is shown below:

**Figure 3: Make Project Window**

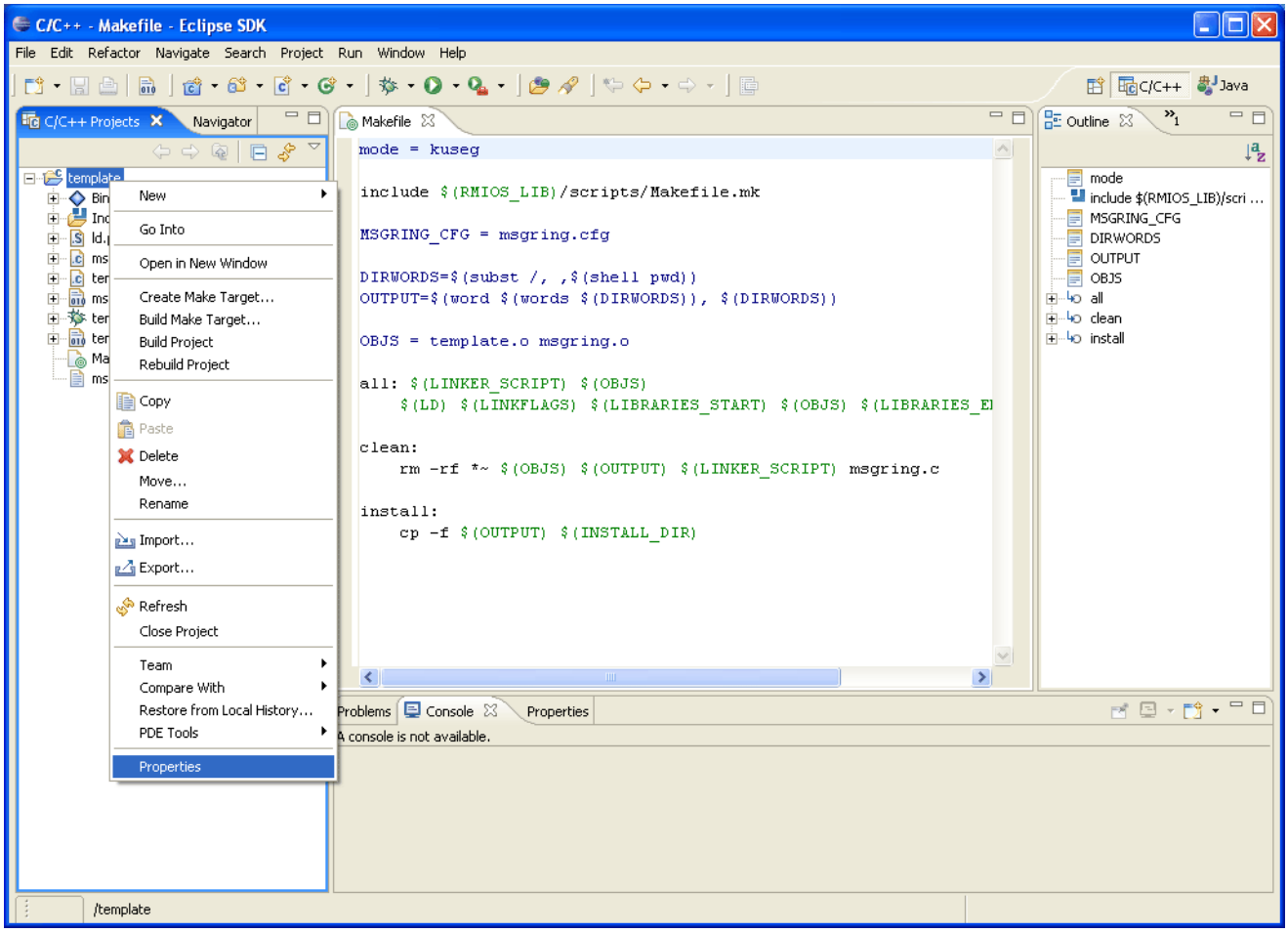



In this window, enter the name of the new project. Click the “Finish” button.

## 4.1 Settings for the Project

In the main window of Eclipse application, select the “project” in the left pane. Right click on the “project” and select “properties” option as shown below.

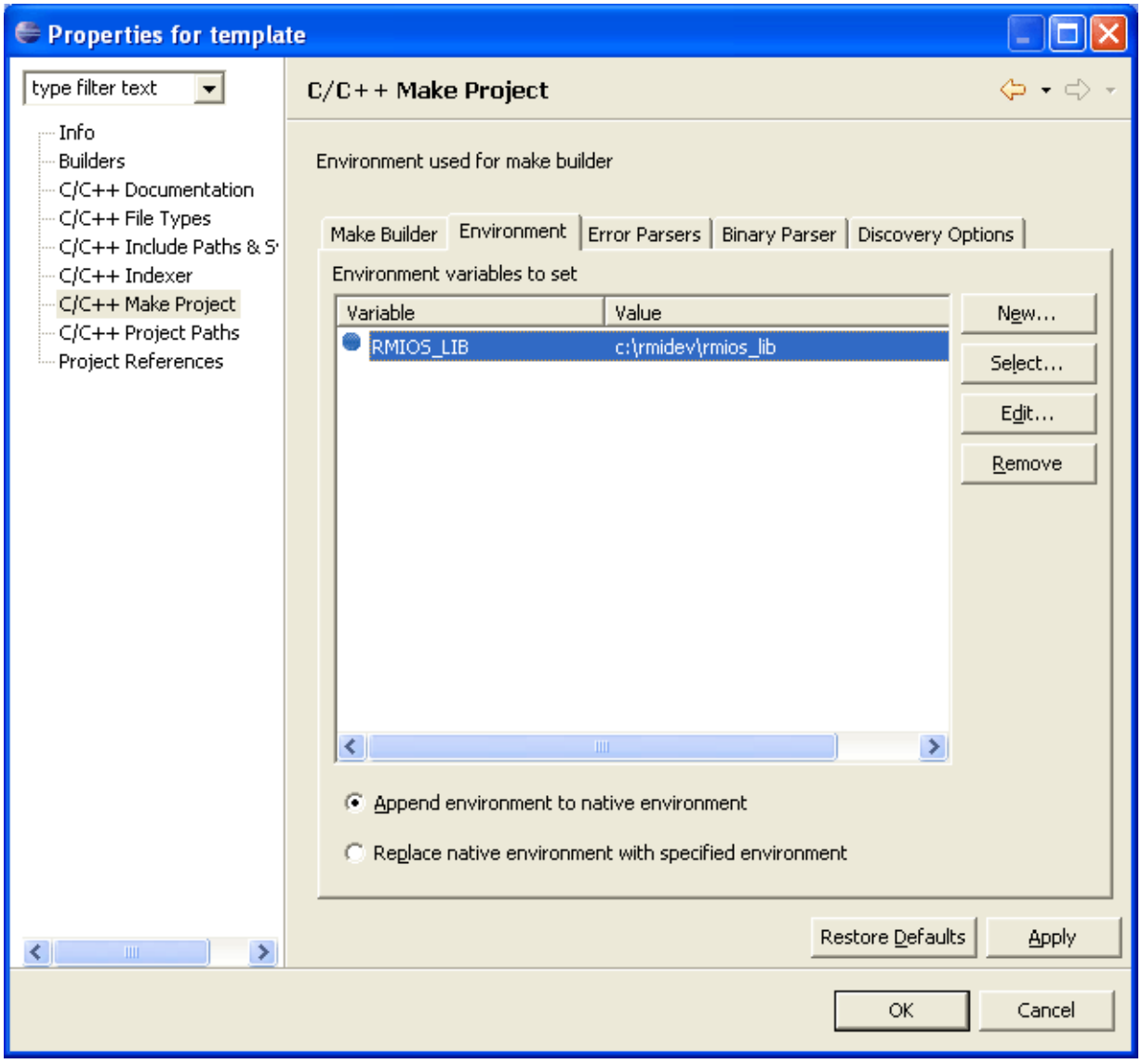
**Figure 4: Eclipse window, Properties option**




The “Properties” window is invoked. In the left pane of this window, select “C/C++ Make Project offer” option. Select the ‘Environment’ tab.

While installing the RMIOS toolkit, an environment variable error will be generated if you are installing in a normal user profile. In order to rectify this error, you need to set up “RMIOS_LIB” variable in this tab. Click the “New” button and add the new variable. If the default settings for installation are used, then this value is c:\rmidev\rmios_lib.

**Figure 5: Make Project Settings, Environment Tab**

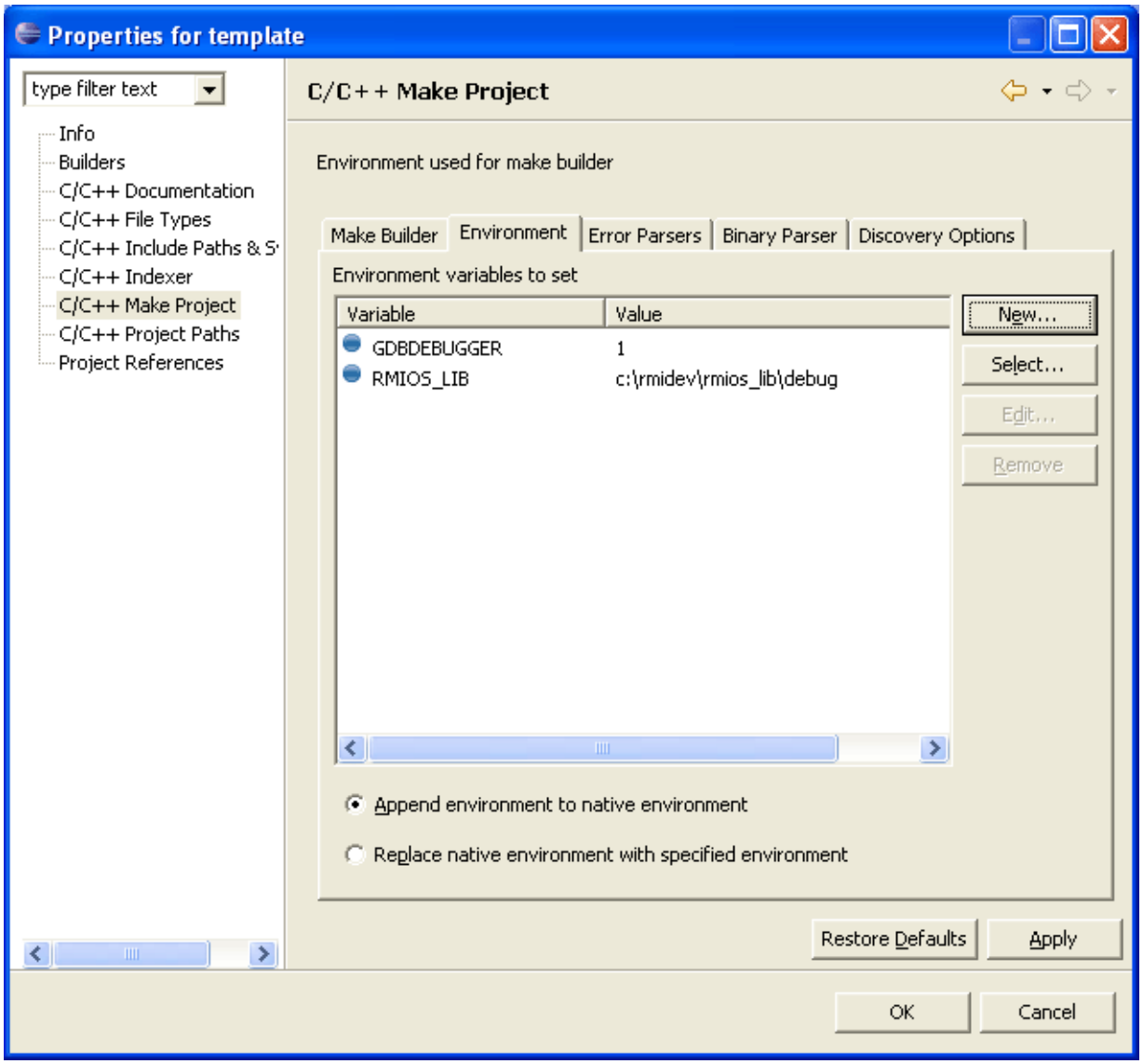


**Settings for debugging with GDB/Insight in "Environment" Tab.**

If the RMIOS applications are to be built with debugging support, then set the following values:

- GDBDEBUGGER variable as "1".
- RMIOS_LIB variable to c:\rmidev\rmios_lib\debug directory.

**Figure 6: Make Project Settings Window, Environment Tab**

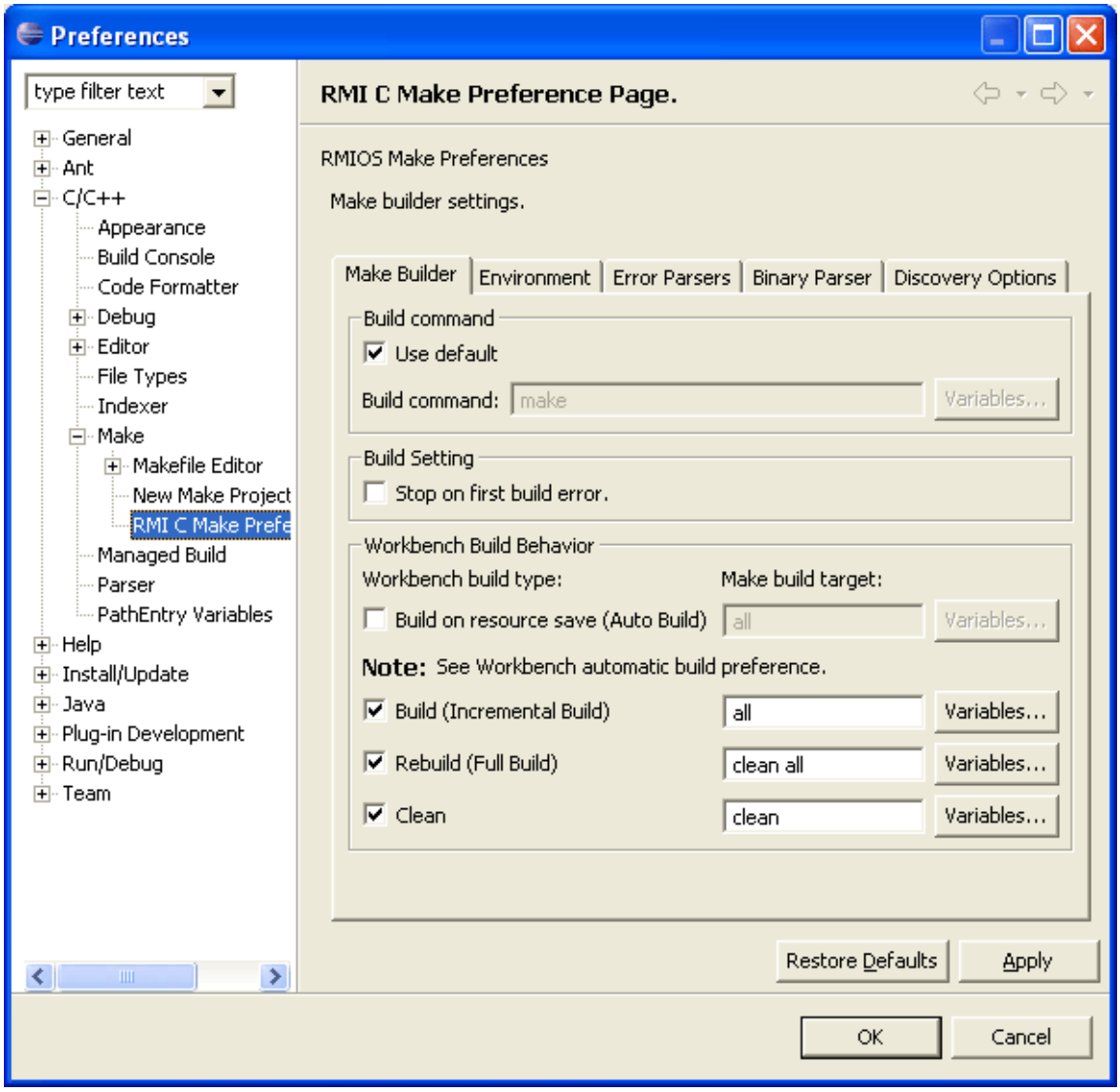



## 4.2 How to change RMIOS Project's Default Make Settings?

To change the default make settings for new RMIOS projects, select the option **Window ⏩ Preferences** in the Eclipse main window. The "Preferences" window is invoked.

In the Preferences dialog, select **C/C++ ⏩ Make ⏩ RMI C Make preferences**, to change the default settings. These settings will be used as the default for all future RMIOS C projects.

**Figure 7: Preferences Window**

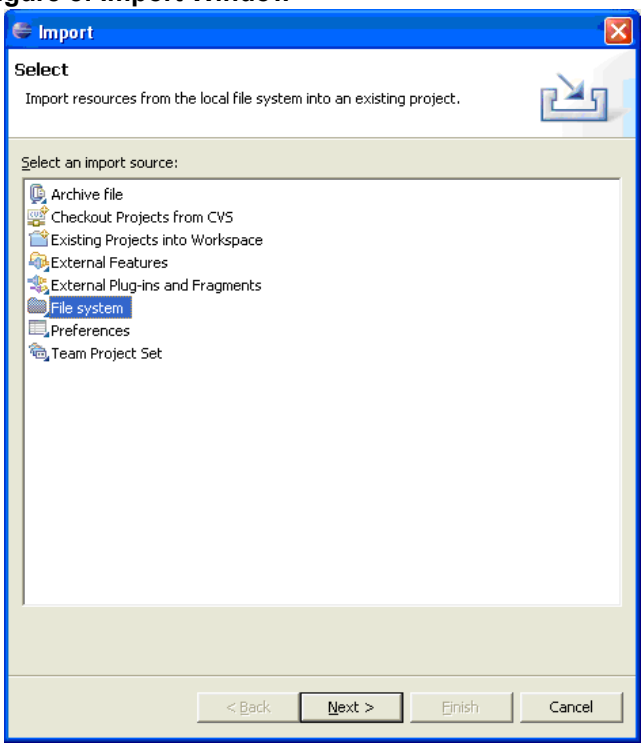



## 4.3 To import an example project from the RMIOS sources

In the main window of the Eclipse application, right click on the “project” option in the left pane, and select “Import”. The Import wizard is invoked.

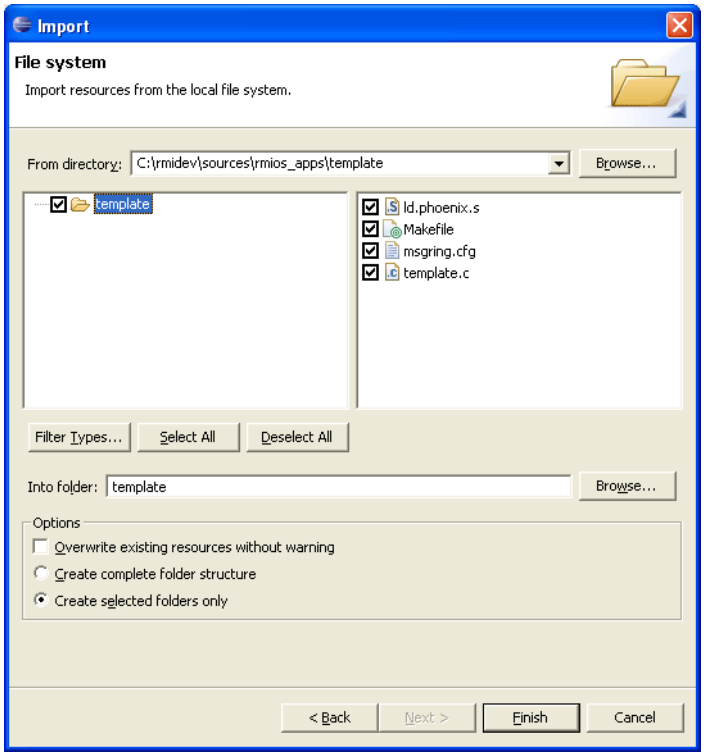
In the import wizard, choose the “File System” option as shown below.

**Figure 8: Import Window**




Click the “Next” button to invoke the “File System” window. In the “File system” window, select the project/application (in this case, it is ‘template’) from the RMIOS sources directory, and click the ‘Finish’ button.

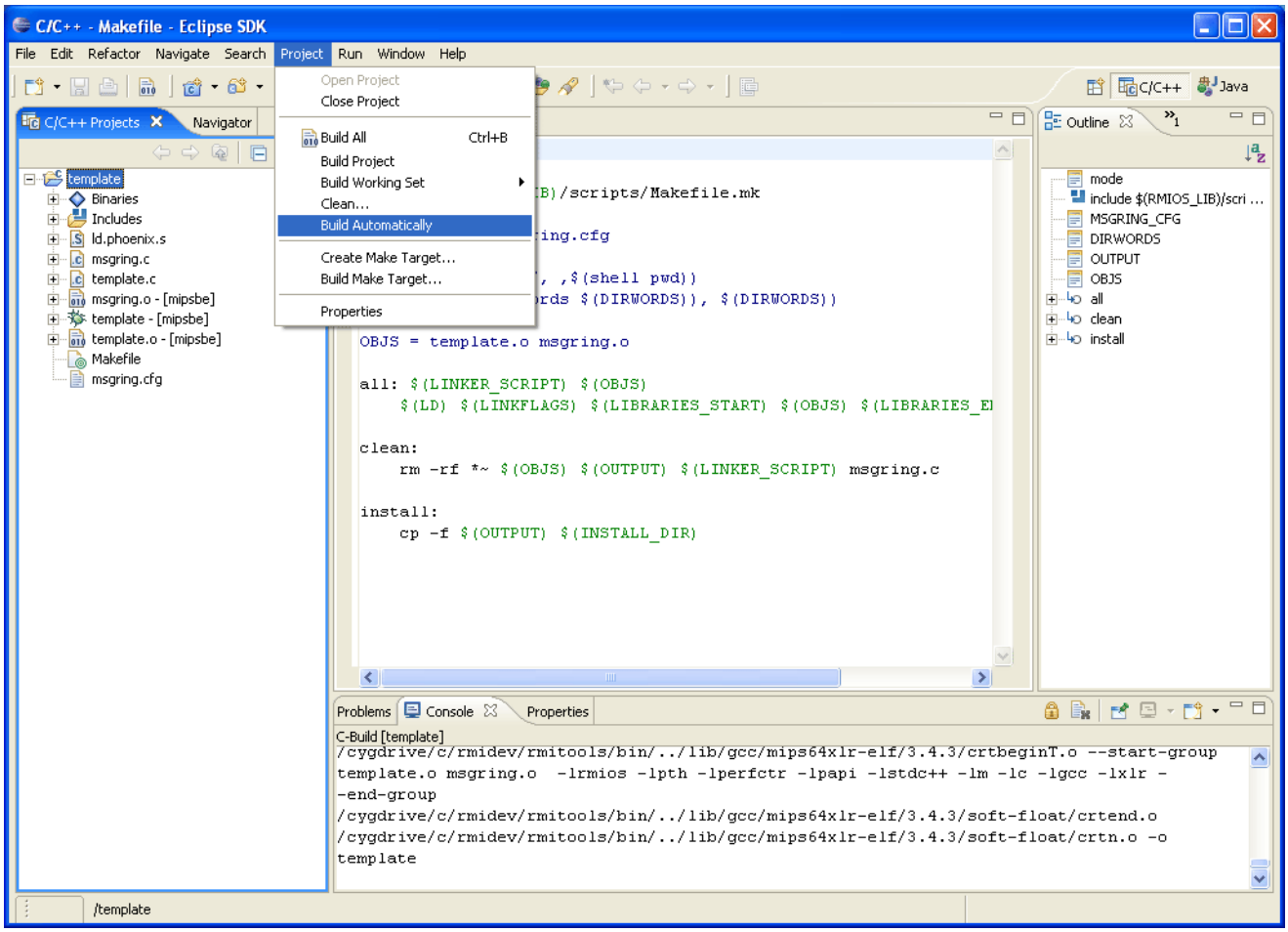
**Figure 9: File System Window**



## 4.4 Building the example project

You can use the menu option **Project ⏩ Build Project** to build RMIOS applications. Once the application is built this way, you can see the executable and object files in the workspace as shown below.

**Figure 10: Work Space Window**




**Note: Ensure to deselect the option Project ⏩ Build Automatically to obtain a successful build of the application.**

# 5 Debugging RMIOS Applications

Insight GDB is the GNU debugger, which is used for debugging RMIOS applications.

**Note: This software is installed as a part of RMIOS development tool kit.**

## 5.1 What does GDB do?

The GNU Debugger, usually referred to as GDB, is the standard debugger for the GNU software system.

GDB allows you to see what is going on `inside' another program while it executes (or) what another program was doing at the moment it crashed.

GDB can perform four main kinds of operations to help you detect bugs in your applications:

- Start your program, specifying anything that might affect its behavior.
- Make your program stop on specified conditions.
- Examine what happened, when your program stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

**Note: For more information on GDB, refer to the GDB web site at: http://www.gnu.org/software/gdb/gdb.html**

## 5.2 Remote debugging in Network Mode

The connection between the host (gdb) and target (xlr) is established through a TCP/IP socket based network link in this mode of debugging.

**Note: Network mode is supported in RMIOS + VxWorks and RMIOS + Linux scenarios of RMIOS debugging options. For more information on the other methods for RMIOS debugging, refer to the "XLR - RMIOS Debugger" User manual.**

### 5.2.1 RMIOS Debugging through VxWorks

RMIOS applications are debugged using GDB through VxWorks as follows:

VxWorks is loaded in virtual CPU 0 (core 0, thread 0) and provides the socket interface through which packets from GDB are tunneled to RMIOS applications running in other virtual CPUs (1-31).

### 5.2.2 RMIOS Debugging through Linux.

RMIOS applications are debugged using GDB through Linux as follows:

Linux is loaded in virtual CPU 0 (core 0, thread 0) and provides the socket interface through which packets from GDB are tunneled to RMIOS applications running in other virtual CPUs (1-31).

## 5.3 Steps to bringup insight debugger

### 5.3.1 How to get the images?

You can either use the release binaries from SDK 1.1 or checkout the sources and compile to obtain the images.

### To get RMIOS image:

From SDK 1.1 Sources :

Copy /opt/rmi/1.1/rmios/binaries/debug/apps/debug_test to the tftp directory.

(or)

Build the sources to get the image as shown below:

1. Change the directory to RMIOS Lib as follows:

   ```
   cd  /opt/rmi/1.1/rmios/src/rmios_lib
   ```

2. Compile “rmios_lib” as "make GDBDEBUGGER=1 clean all install." This creates “librmios.a” under the lib directory.
3. Change the directory using:

   ```
   cd rmios_apps/debug_test
   ```

4. Export “RMIOS_LIB=path_of_rmios_lib.”
5. Compile “rmios_apps” as "make GDBDEBUGGER=1 clean all install." This creates "debug_test" image in the current directory.
6. Copy the image “debug_test” to the tftp directory.

### VxWorks image:

1. Get the latest VxWorks sources from the location “/proj/aps/apslabuser/vxWorks/latest” in any of the Unix machines in RMI U.S.
2. Set "XLR_RMIOS_DEBUG_SUPPORT = 1" in the Makefile.
3. Make “xlrPrimary” (in Tornado). This creates the image “primary_vxworks” in the current directory.
4. Copy the image “primary_vxworks” to the tftp directory.

### Linux image:

From SDK 1.1 :

1. Copy the Linux source from /opt/rmi/1.1/linux/src/ to the current directory.
2. Change directory using the following command:

   ```
   cd src
   ```

3. Make “menuconfig.”
4. Select **Device Drivers** ⏩ **Character devices** ⏩ **RMIOS Debugger support**, save and exit.

5. Make the image “vmlinux.”
6. Copy the image “vmlinux” to the tftp directory.

### Insight image:

You can obtain the “insight” image from the release and install the same.

#### 5.3.2 On the xlr side

### With vxworks tunneler:

```
ifconfig -i gmac0
tftpc -s 10.7.0.124 -f path_of_debug_test_image
userapp_os -m 0xf0
tftpc -s 10.7.0.124 -f path_of_primary_vxworks_image
elfload
userapp
```

From the vxworks shell, it can be checked if the vxworks tunneler and "rmi_debugger" task is up, using the following command:

```
<vxworks_shell> i
```

**Note: The tunneler which is part of primary_vxworks in vCPU 0 facilitates communication between insight/gdb and any of the secondary vCPUs(1-31) containing rmios applications (here, it is debug_test).**

### With linux tunneler:

```
ifconfig -i gmac0
tftpc -s 10.7.0.124 -f path_of_debug_test_image
userapp_os -m 0xf0
tftpc -s 10.7.0.124 -f path_of_vmlinux_image
userapp_mask_cpus 1
elfload
userapp mem=176m@ 16m
```

When Linux comes up, it creates a pseudo device “/dev/phnxdeb” with a major number. Note down this major number. Once Linux is up, it  prompts for login. Login as root to enter the Linux shell, which is shown below.

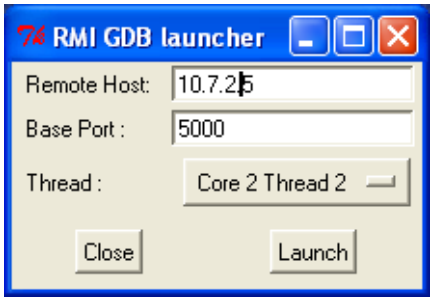
```
<linux_shell> mkdir /devtree
<linux_shell> ifconfig eth0 ipaddr_of_xlr
<linux_shell> mount -t nfs -o nolock 10.7.0.124:/phoenix_nfsroot/rdroot/devtree /
devtree /* this is the ip addr and NFS dir in the tested environment; Use the
appropriate ip addr and NFS dir in your environment */
```

```
<linux_shell> chroot /devtree/root

<linux_shell> mount -t proc none /proc

/* Check if /dev/phnxdeb is present with major no same as above as follows: */

<linux_shell> ls -l /dev/phnxdeb


If not, create the node as:

<linux_shell> mknod c majorno 0


/* copy linux tunneler to current dir */

<linux_shell> cp -R /opt/rmi/1.1/rmios/src/rmios_apps/debug_test/network/linux/
app .

<linux_shell> cd app

<linux_shell> gcc xlrRmiosDebug.c


/* This will create a server app a.out. Run a.out */


<linux_shell> ./a.out &


/* This will print a message "starting rmios debug server" */
```

## 5.4 Launching Insight gdb for RMIOS IDE

In Windows systems, go to **Start ⏩ Programs ⏩ RMIOS Development IDE ⏩ RMI Insight Launcher** to invoke the 'RMI GDB Launcher' window.

**Figure 1: RMI GDB Launcher**




Enter the following details in this window:

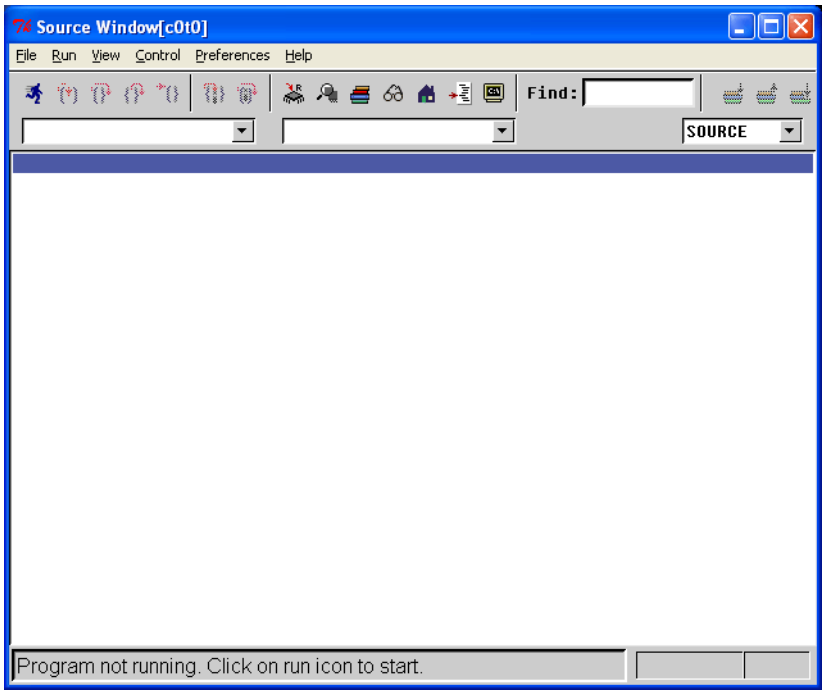
- Remote Host: Enter the target 'IP address' (in this case it is the IP address of XLR).

- Base Port: The starting ‘Port number’ of the GDB tunnel running on the primary OS is displayed here. The default value is “5000”.

- Thread:  This field contains a drop-down list of 32 vCPUs. Select the vCPU, (i.e.. core-thread pair) that you want to connect. Ensure this vCPU is already up with “debug_test” image in xlr.

After entering these details, click the ‘Launch’ button to invoke the Insight gdb GUI, which is shown in the figure below:

**Figure 2: Source Window**




- In this window, select **File ⏩Open ⏩**and provide the path of the image “debug_test,” which should be downloaded to the windows machine.

- It is preferable if the file you open is in the same directory as the source since the listing is better while debugging, as opposed to opening in the tftpboot directory, which may not provide the complete listing.

- To do the Source level debugging of “rmios_lib,” set the source directory on Insight as shown below:

    - Open the GDB console window (click on console icon in the tool bar).

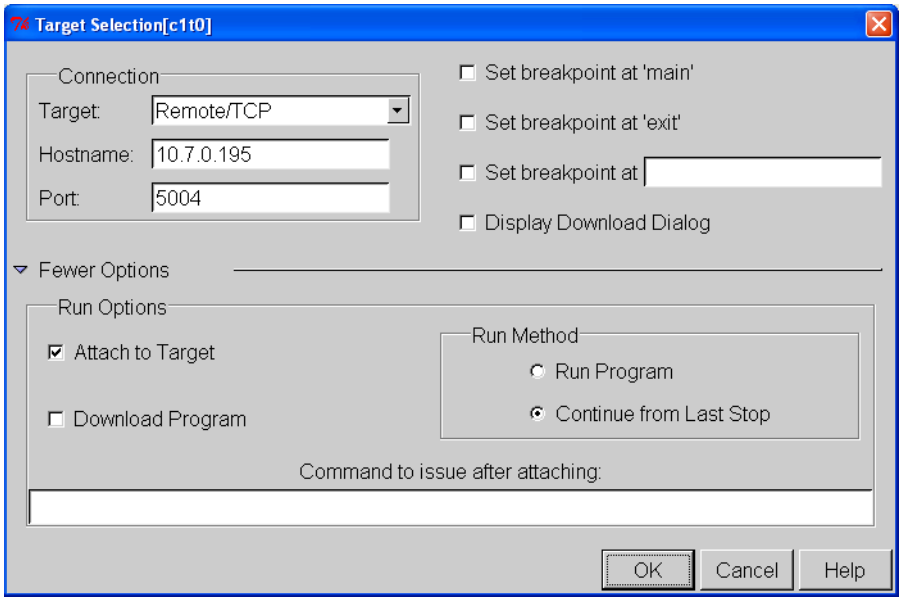
    - Set the source directory on the console with the command:

```
dir /cygdrive/c/rmidev/sources/rmios_lib/srcs/kernel:/cygdrive/c/rmidev/sources/
rmios_lib/srcs/kernel/include
```

- Ensure to check the option "**File ⏩Target-Settings**."

- Select **Run ⏩ Connect to target**, to invoke **“Target Selection”** window as shown below.

**Figure 3: Target Selection Window**




- In the Target Selection window, perform the following steps:

1. Select Remote/TCP option in the "Target" field.
2. Fill the "Hostname" field with the target IP address.
3. Ensure that the "Port" field corresponds to the vCPU that you want to connect. Port numbers 5001-5031 correspond to vCPU 1-31 respectively.

   **Note: Port number cannot be 5000 as it corresponds to the vCPU 0, which is not up with "debug_test image."**

4. Click on "More options," deselect 'Download to target' and all other options related to breakpoints.
5. Click the "OK" button. A "GDB" message window pops up with a message "Successfully connected."
6. Now in the "Source window," the highlighted instruction is where the execution has stopped.
7. Click the "Continue" button in the menu bar .

   **Note: Do not press the "Run" button.**

8. Click the "STOP" button in the menu bar to stop the execution (through a ctrl-c). A "Warning window" appears stating the message "Program received signal SIGTERM, Terminated."
9. Clicking "OK" in this "Warning window" shows the "Source window" containing the highlighted instruction where the execution has stopped.
10. Breakpoints can be set on an instruction (by right-clicking the mouse) in the "Source window" and a "Continue" will make the execution stop at the breakpoint.

    **Note: Breakpoints can be set on functions, with the mouse on the function names, or on lines, with the mouse over the line numbers.**

11. Other commands can be used by selecting appropriate menu/buttons.

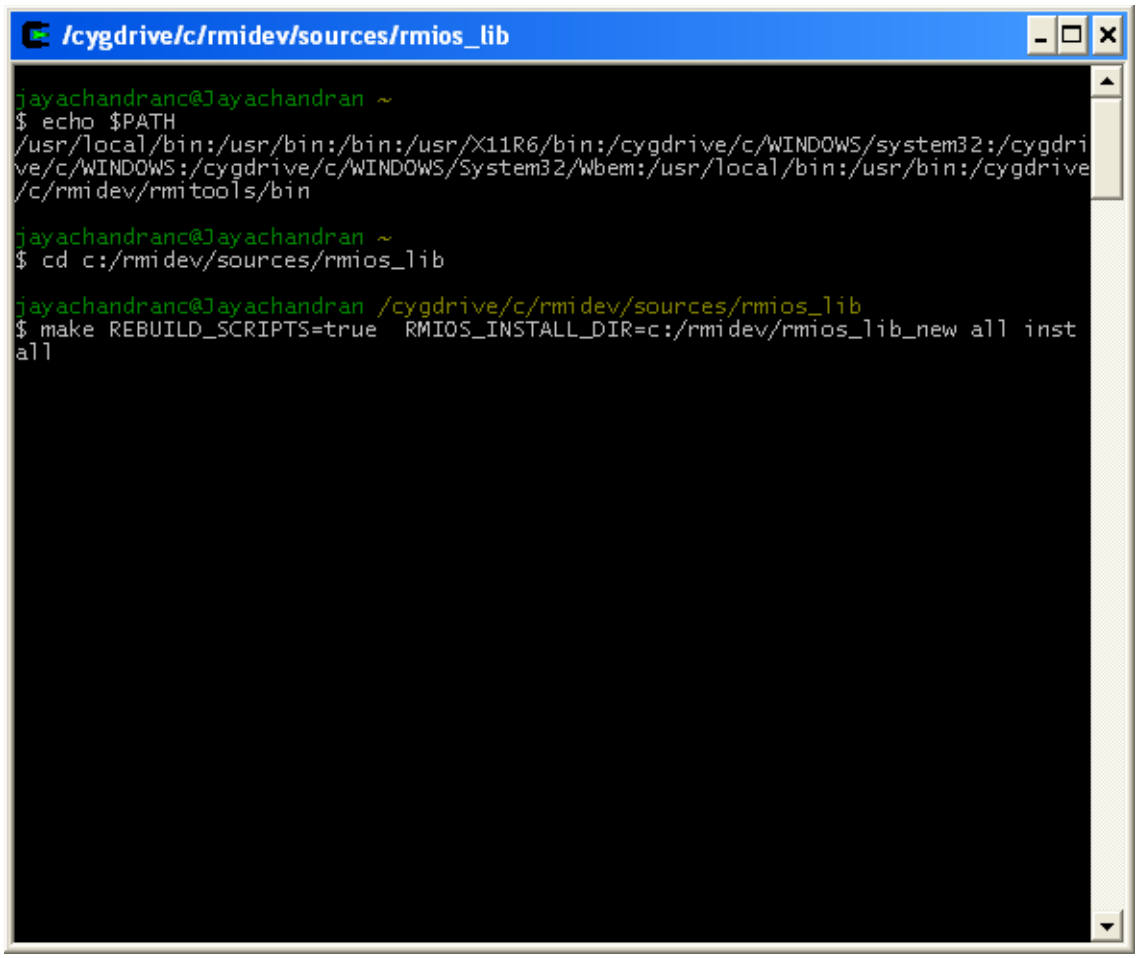
12. Click the ‘OK’ button to connect to the target (XLR) and start the debugging operations.

    **Note: For more information on debugging RMIOS applications using Insight GUI, refer to the help provided with the Insight/gdb application.**

# 6 Building a Custom RMIOS library

1. Install Cygwin on your workstation, from http://cygwin.com/.
2. Open a Cygwin window, and check if the environment contains the Cygwin directories and the RMIOS development environment directory.
3. Change directory to c:/rmidev/sources/rmios_lib.

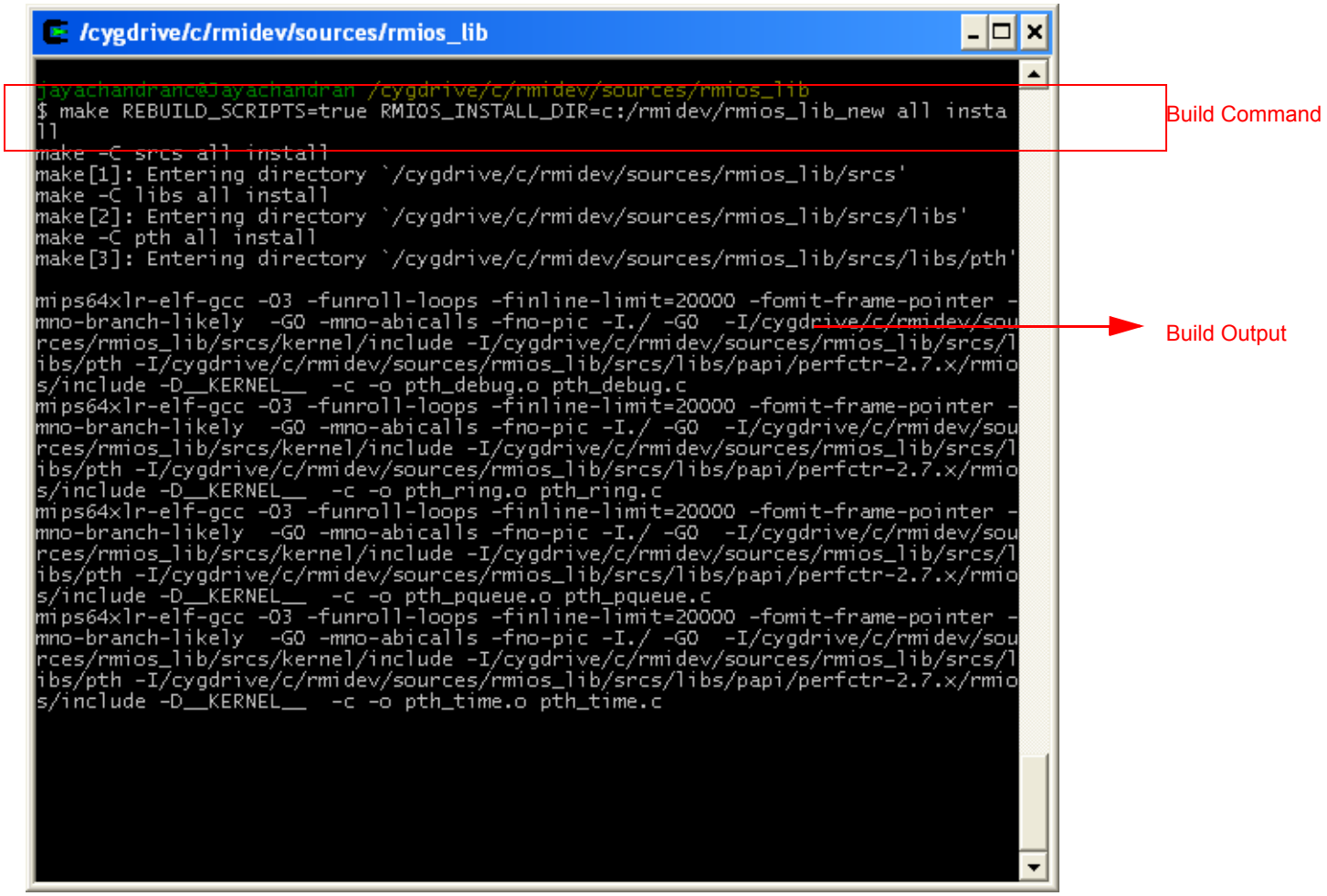
**Figure 1: Cygwin window, showing “Build” commands**




4. Use the following command to build the release version of RMIOS library. By default this will get installed in C:/rmidev/rmios_lib_new.

```
make REBUILD_SCRIPTS=true  RMIOS_INSTALL_DIR=c:/rmidev/rmios_lib_new all install
```

**Figure 2: Cygwin window showing the output of the build**




5. To build a debug version of the library, first clean the current build using the following command:

   ```
   make clean
   ```

6. Use the following command to build a debug version.

   ```
   make REBUILD_SCRIPTS=true GDBDEBUGGER=1 RMIOS_INSTALL_DIR=c:/rmidev/
   rmios_lib_new/debug all install
   ```

7. You can start using the new libraries by setting the RMIOS_LIB environment variable, while building RMIOS applications.

# 7 Limitations / Known issues with RMI Windows SDK package

This section summarizes the limitations while using the RMIOS SDK package for Windows:

- The RMIOS SDK cannot be installed in directories containing space (' '). The installer will give a warning message if you attempt to save it in such a directory. So ensure that this directory is created without any spaces.
- The Eclipse IDE launch/debug mechanism cannot be used for running or debugging RMIOS applications. Applications have to be launched using a serial console, using a TFTP server and GDB/Insight debugger is included in the package, which can be used as a debugger to debug RMIOS applications.