RTEMS Eclipse Manual

Release 4.11.3

©Copyright 2016, RTEMS Project (built 15th February 2018)

CONTENTS

Part I

RTEMS Eclipse Manual

COPYRIGHT (c) 2016

Chris Johns <chrisj@rtems.org>

The authors have used their best efforts in preparing this material. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. No warranty of any kind, expressed or implied, with regard to the software or the material contained in this document is provided. No liability arising out of the application or use of any product described in this document is assumed. The authors reserve the right to revise this material and to make changes from time to time in the content hereof without obligation to notify anyone of such revision or changes.

The RTEMS Project is hosted at http://www.rtems.org/. Any inquiries concerning RTEMS, its related support components, or its documentation should be directed to the Community Project hosted at http://www.rtems.org/.

RTEMS Online Resources

Home	https://www.rtems.org/
Developers	https://devel.rtems.org/
Documentation	https://docs.rtems.org/
Bug Reporting	https://devel.rtems.org/query
Mailing Lists	https://lists.rtems.org/
Git Repositories	https://git.rtems.org/

Part II

Table of Contents

CHAPTER
ONE

OVERVIEW

Welcome to the *RTEMS* Eclipse Manual.

This document covers using Eclipse with RTEMS.

RTEMS, Real-Time Executive for Multiprocessor Systems, is a real-time executive (kernel) which provides a high performance environment for embedded applications.

Eclipse is an Integrated Development Environment (IDE) for a wide range of languages and platforms.

RTEMS's eco-system provides all the tools and capabilities to integrate with Eclipse. You can build and develop RTEMS with Eclipse as well as build applications with Eclipse.

Unless otherwise stated this document refers to the Eclipse Mars release.

CHAPTER
TWO

RTEMS DEVELOPMENT

RTEMS can be developed using Eclipse. The RTEMS kernel is an *autotools* or *autoconf* and *automake* based package. You can create a project in Eclipse that lets you configure and build a BSP for an architecture. We assume you have already build and installed your tools using the RTEMS Source Builder.

2.1 Kernel Source

Download or clone the RTEMS Kernel source code. We will clone the source code:

```
$ git clone git://git.rtems.org/rtems.git rtems.master
Cloning into 'rtems'...
remote: Counting objects: 483342, done.
remote: Compressing objects: 100% (88974/88974), done.
remote: Total 483342 (delta 390053), reused 475669 (delta 383809)
Receiving objects: 100% (483342/483342), 69.88 MiB | 1.37 MiB/s, done.
Resolving deltas: 100% (390053/390053), done.
Checking connectivity... done.
```

We need to *bootstrap* the kernel source code. A *botostrap* invokes the various *autotools* commands need to generate build system files. First we need to the path to our tools:

```
$ export PATH=/opt/rtems/4.12/bin:$PATH
```

Now run the *bootstrap* command:

```
$ cd rtems.master
$ ./bootstrap
```

Sit back, this can take a while. The Getting Started Guide talks about using the RSB's *sb-bootstrap* to run the bootstrap process in parallel on all available cores. The output of the bootstrap has not been copied into this documentment.

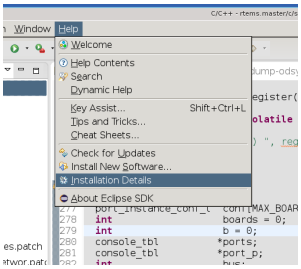
The source code is now ready.

2.2 Eclipse SDK Software

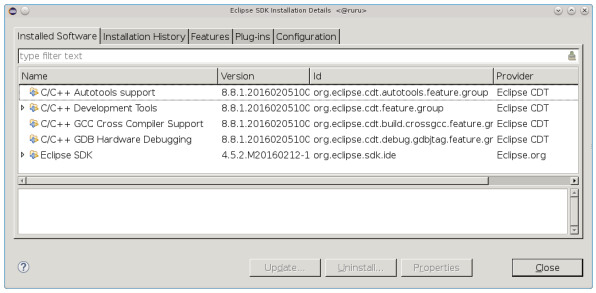
We need the following Eclipse SDK Software packages installed:

- C/C++ Autotools support
- C/C++ Development Tools
- C/C++ GCC Cross Compiler Support

Start Eclipse and check to see if you have the them installed via the **Help, Installation Details** menu item:



The dialog box shows the installed software packages and you can see the **C/C++ Autotools support** and the **C/C++ Development Tools** are installed:

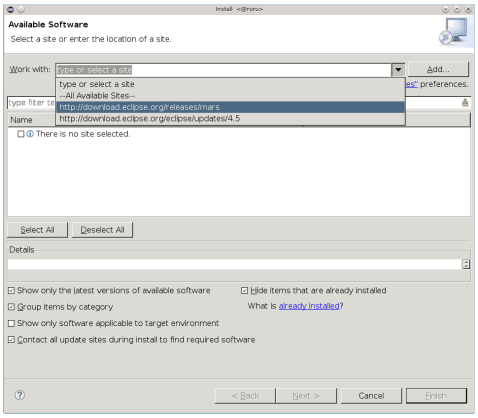


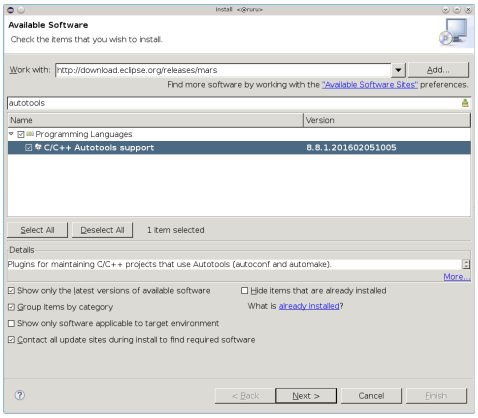
You can see some other software packages are installed in the figure. You can ignore those.

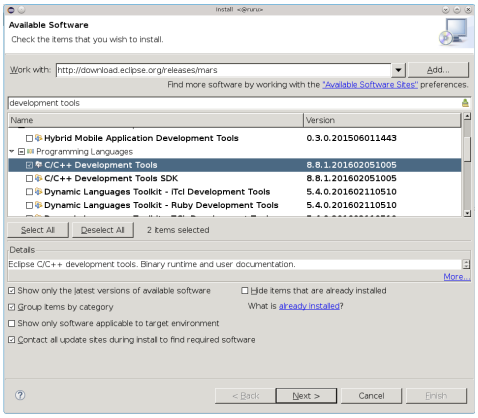
If you do not have the listed software packages install select **Help, Install New Software** and in the **Work with:** list box select **http://download.eclipse.org/releases/mars**.

Afer a small period of time a list of available packages will populate and you can select the ones we are interested in. Enter `autotools` in the search box and select the package:

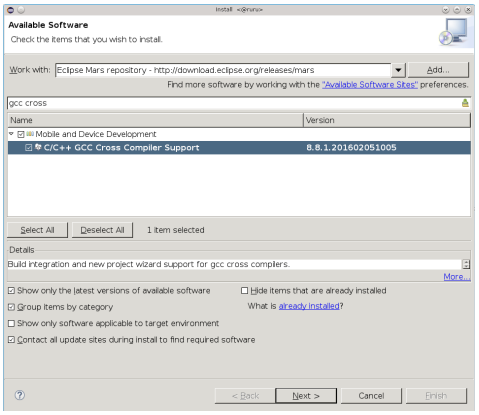
Clear the search line and enter `development tools` in the search box and then scroll down to find **C/C++ Development Tools**:







Again clear the search line and enter `gcc cross` in the search box and select the package:



Click **Next** and once the **Install Details** have determined what is needed select **Finish** to install the packages.

2.3 Kernel Build Project

We create a project in Eclipse that can configure and build RTEMS for the `pc686` BSP. This BSP is based on the `pc386` BSP and is under the `i386` architecture.

We assume you have built and installed the `i386` RTEMS Tools, obtained the RTEMS kernel code and `bootstrapped` it if a git clone, and installed the required Eclipse Software packages.

The paths used in this project are:

`/opt/work/rtems/4.11`
The RTEMS Tools prefix the tools are install under.

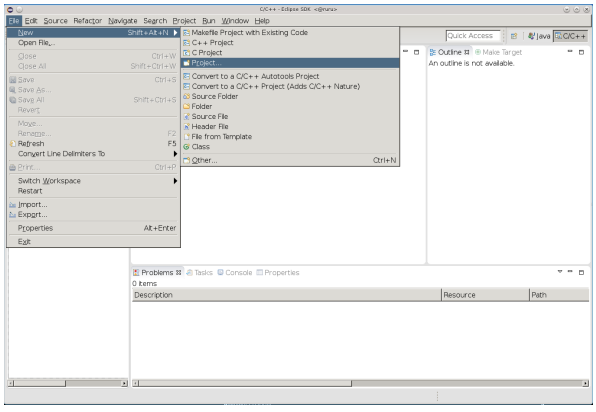
`/opt/work/chris/rtems/kernel/rtems.master`
The RTEMS Kernel source code.

`/opt/work/chris/rtems/kernel/4.12`
The RTEMS Kernel prefix.

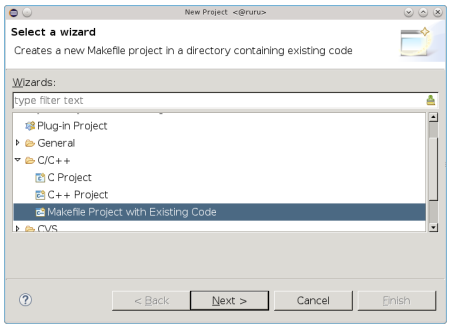
`/opt/work/chris/rtems/kernel/bsp/pc`
The RTEMS Kernel BSP build directory.

The menus shown here may vary from those you have as Eclipse changes them based on what you do.

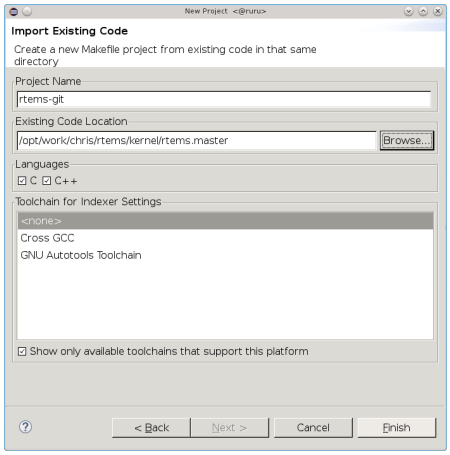
Select **File, New, Project** :



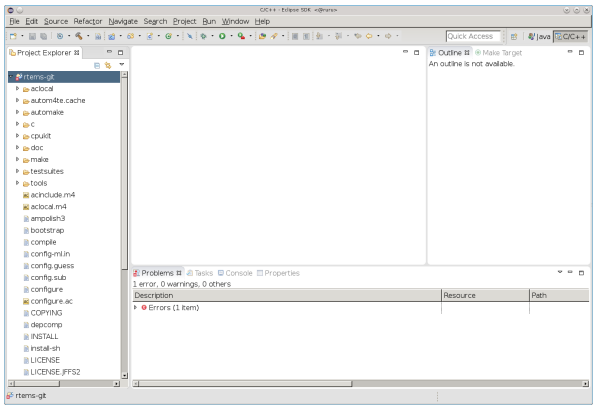
Click on **C/C++** and select **Makefile Project with Existing Code** then select **Next** :



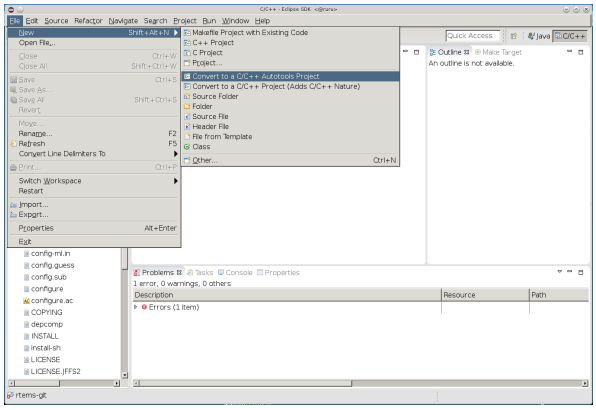
Enter the project name `rtems-git` into the **Project Name** field and select the **Browse...** button and the path to the RTEMS Kernel source code then click **Finish** :



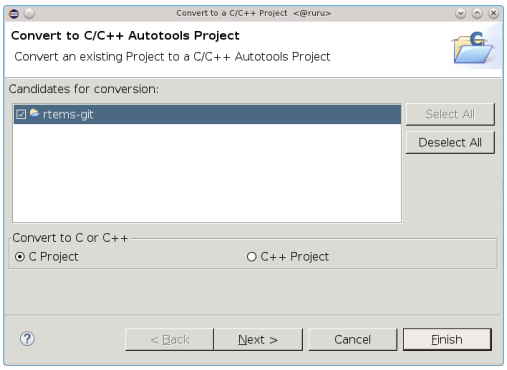
Eclipse will show the RTEMS Kernel source code in the **Project Explorer** panel:



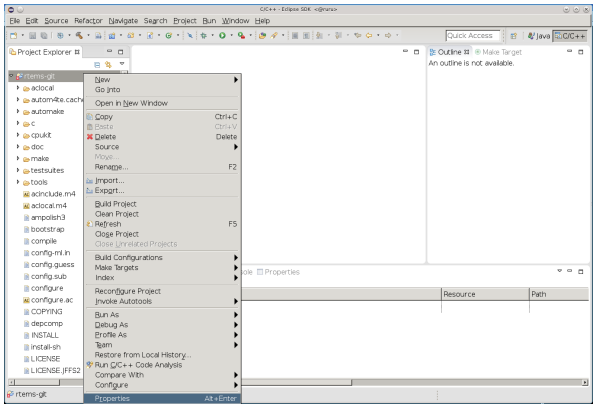
We now convert the project to an Autotools project. Select **File, New, Convert to a C/C++ Autotools Project** :



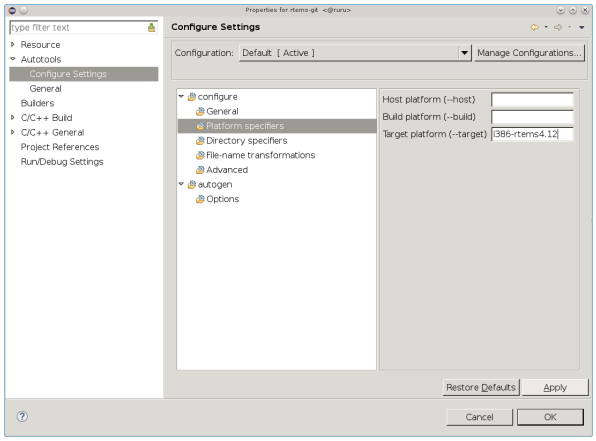
Select **C Project** then **Finish** :



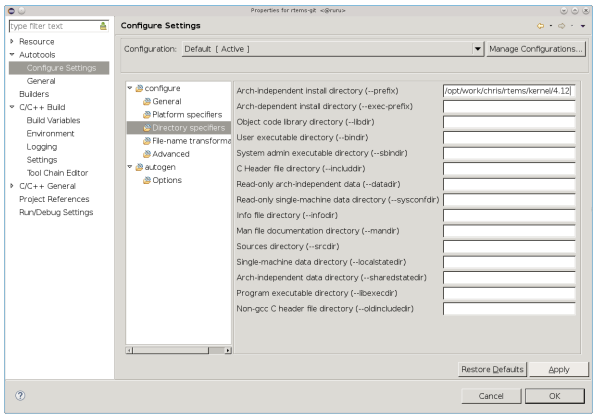
We now configure the project's properties by right clicking on the `rtems-git` project title and then **Properties** :



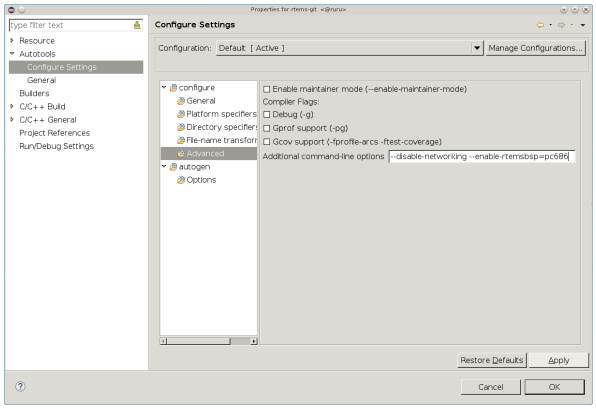
Click on the **Autotools** item then **Configure Settings** and **Platform specifiers** and set the **Target platform** field with `i386-rtems4.12`:



Select **Platform directories** and enter the **Arch-independent install directory (–prefix)** to the RTEMS Kernel prefix of `/opt/work/chris/rtems/kernel/4.12`:

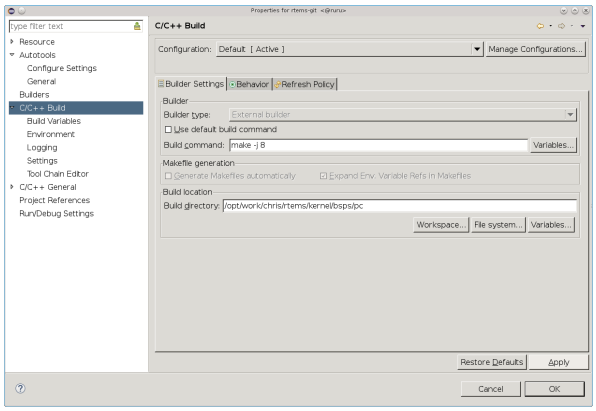


We disable networking to use the external LibBSD package and set the BSP to `pc686`. Select the **Advanced** and in the **Additional command-line options** enter `--disable-networking` and `--enable-rtemsbsps=pc686`. You can add extra options you may need:

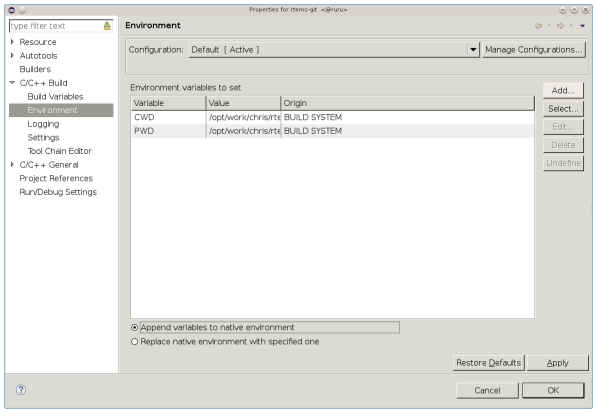


Select **C/C++ Build** and **Environment**. Uncheck or clear the **Use default build com-**

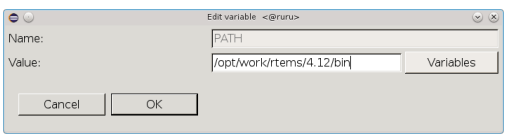
mand and add `-j N` where `N` is the number of cores you have in your machine. The figure has told *make* to run 8 jobs, one per core for an 8 core machine. Click on the **File system...** button and navigate to the BSP build directory. This is the location Eclipse builds the BSP. RTEMS requires you build outside the source tree and in this example we are forcing the build directory to something specific. Finish by pressing **Apply** :



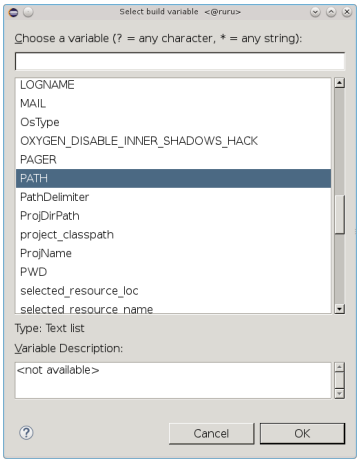
Select **Environment** under **C/C++ Build** as we need to set the path to the RTEMS Tools. In this example we set the path in the Eclipse project so each project can have a specific set of tools. Press the **Add...** button:



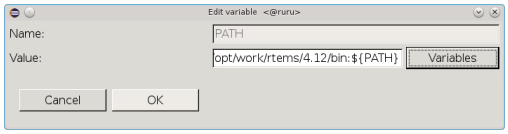
Enter the path to the tools, in our case it is `/opt/work/rtems/4.12/bin`, then press **Variables** :



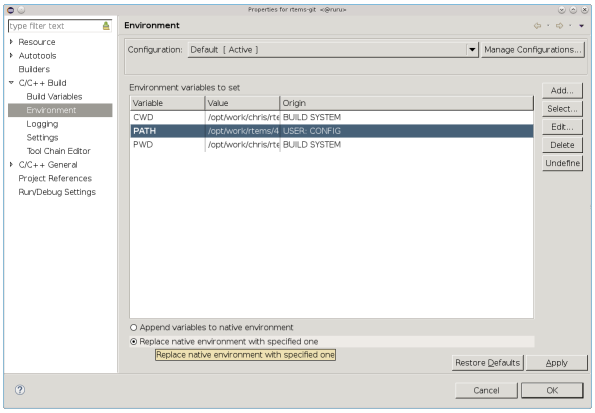
Scroll down and select **PATH** and then press **OK** :



You will now see the path in the **Value:** field. Make sure you have a path separator between the end of the tools path and the path variable we have just added. In this case is a Unix host and the separator is *:*. Windows use *;*. Press **OK** when you have a valid path:



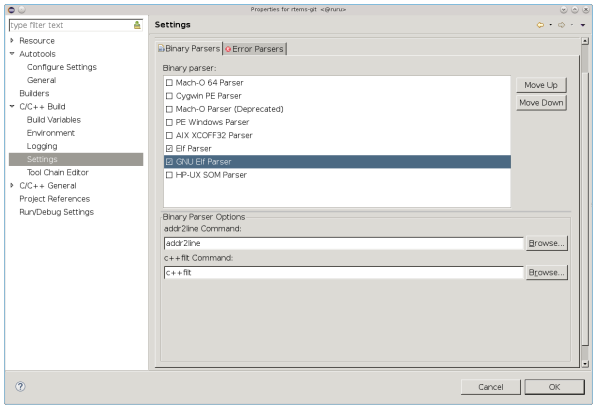
The **Environment** panel will now show the added *PATH* variable. Click **Replace native environment with specified one** as shown and then press **Apply** :

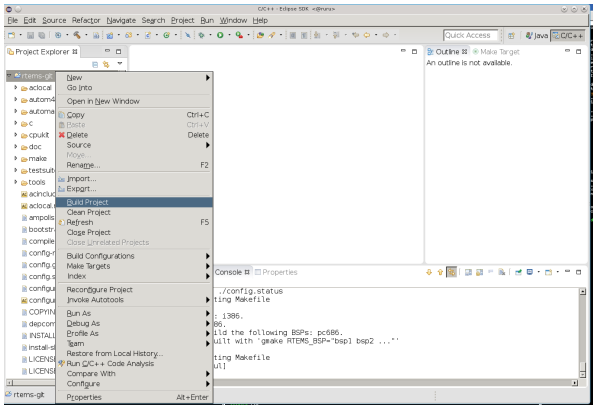


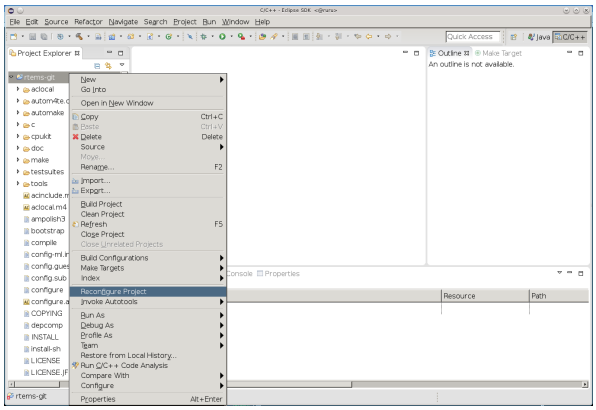
Select **Settings** under **C/C++ Build** and check **Elf Parser** and **GNU Elf Parser** and then press **OK** :

We are now ready to run configure using Eclipse. Right click on the project name `rtems-git` and then **Reconfigure Project** :

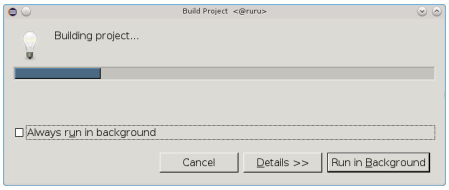
Select the **Console** tab in the output panel to view the configure process output. You will no-

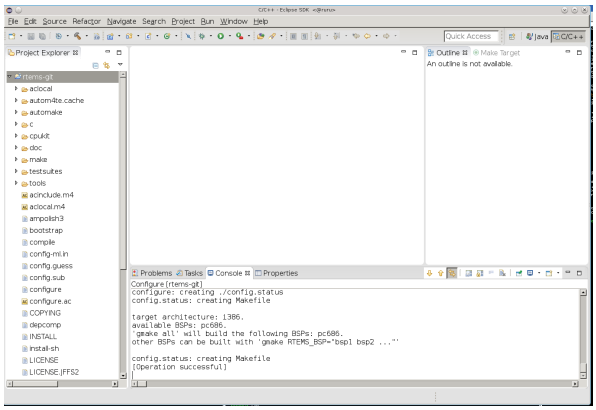






tice the end of the configure process shows the names of the BSPs we have asked to build. In our case this is the `pc686` BSP:

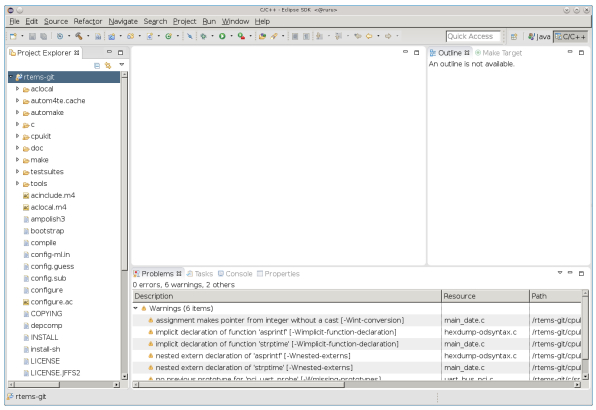




We can now build RTEMS using Eclipse. Right click on the project name `rtems-git` and then select **Build Project** :

A **Build Project** message box will appear showing the progress:

When finished click on the **Problems** output tab to view any errors or warnings:



If you get errors during the configure phase or building you will need to determine rea-

son why. The main source of errors will be the path to the tools. Check the top of the `config.log` file `configure` generates. This file can be found in the top directory of you BSP build tree. The file will list the path components near the top and you should see the path to your tools listed first. While looking make sure the configure command matches what you expect and matches the documentation for configuring RTEMS.

If the contents of `config.log` look fine check the build log. The project's **Properties** dialog under **C/C++ Build**, **Logging** has a path to a build log. Open the build log and search for the error. If you cannot figure out the source of the error please ask on the Users Mailing List for help.

..comment SPDX-License-Identifier: CC-BY-SA-4.0

CHAPTER
THREE

GLOSSARY

Binutils
GNU Binary Utilities such as the assembler `as`, linker `ld` and a range of other tools used in the development of software.

DLL
Dynamically Linker Library used on Windows.

GCC
GNU Compiler Tool chain. It is the GNU C/C++ compiler, binutils and GDB.

GDB
GNU Debugger

MinGW
Minimal GNU system for Windows that lets GCC built programs use the standard Windows operating system DLLs. It lets you build native Windows programs with the GNU GCC compiler.

MinGW64
Minimal GNU system for 64bit Windows. MinGW64 is not the MinGW project.

MSYS2
Minimal System 2 is a fork of the MinGW project's MSYS tool and the MinGW MSYS tool is a fork of Cygwin project. The Cygwin project provides a POSIX emulation layer for Windows so POSIX software can run on Windows. MSYS is a minimal version that is just enough to let `configure` scripts run. MSYS has a simplied path structure to make it easier to building native Windows programs.

POSIX
Portable Operating System Interface is a standard that lets software be portable between compliant operating systems.

prefix
A path used when building a package so all parts of the package reside under that path.

RSB
RTEMS Source Builder is part of the RTEMS Tools Project. It builds packages such as the tools for the RTEMS operating system.

RTEMS
The Real-Time Executive for Multiprocessor Systems or RTEMS is a open source fully featured Real Time Operating System or RTOS that supports a variety of open standard application programming interfaces (API) and interface standards such as POSIX and BSD sockets.

Test Suite
See Testsuite

Testsuite
RTEMS test suite located in the `testsuites/` directory.

Waf
Waf build system. For more information see http://www.waf.io/

- genindex
- search

INDEX