



# RTEMS User Manual

*Release 5.0.0 (master)*

©Copyright 2017, RTEMS Project (built 26th March 2018)



# CONTENTS

<b>1</b>	<b>Overview</b>	<b>3</b>
<b>2</b>	<b>Real-time Application Systems</b>	<b>5</b>
<b>3</b>	<b>Real-time Executive</b>	<b>7</b>
<b>4</b>	<b>RTEMS Ecosystem</b>	<b>9</b>
4.1	Rational . . . . .	10
4.2	Open Source . . . . .	11
4.3	Deployment . . . . .	12
<b>5</b>	<b>Quick Start</b>	<b>13</b>
<b>6</b>	<b>Host Computer</b>	<b>15</b>
6.1	Host Operating Systems . . . . .	16
6.2	POSIX Hosts . . . . .	17
6.2.1	Root Access . . . . .	17
6.3	Apple OS X . . . . .	18
6.4	Microsoft Windows . . . . .	19
6.4.1	Windows Path Length . . . . .	19
6.4.2	Parallel Builds with Make . . . . .	19
6.4.3	POSIX Support . . . . .	20
6.4.4	Python . . . . .	20
6.4.5	Installing MSYS2 . . . . .	20
<b>7</b>	<b>Installation</b>	<b>27</b>
7.1	Prefixes . . . . .	28
7.1.1	Project Sandboxing . . . . .	28
7.2	Releases . . . . .	31
7.2.1	RTEMS Tools and Kernel . . . . .	31
7.3	Developer (Unstable) . . . . .	38
7.3.1	POSIX and OS X Host Tools Chain . . . . .	38
7.3.2	Windows Host Tool Chain . . . . .	42
7.3.2.1	RTEMS Windows Tools . . . . .	42
7.3.2.2	Building the Kernel . . . . .	45
7.4	RTEMS Kernel . . . . .	50
7.4.1	Development Sources . . . . .	50
7.4.2	Tools Path Set Up . . . . .	50
7.4.3	Bootstrapping . . . . .	50

7.4.4	Building a BSP	52
7.4.5	Installing A BSP	55
7.4.6	Contributing Patches	56
<b>8</b>	<b>Hardware</b>	<b>59</b>
8.1	Targets	60
8.2	Architectures	61
8.3	Board Support Packages (BSP)	62
8.4	Tiers	63
<b>9</b>	<b>Board Support Packages (BSPs)</b>	<b>65</b>
9.1	ARM Board Support Packages	66
9.1.1	NXP i.MX	66
9.1.1.1	Build Configuration Options	66
9.1.1.2	Boot via U-Boot	66
9.1.1.3	Clock Driver	67
9.1.1.4	Console Driver	67
9.1.1.5	I2C Driver	67
9.1.1.6	SPI Driver	67
9.1.1.7	Network Interface Driver	67
9.1.1.8	Caveats	68
<b>10</b>	<b>Host Tools</b>	<b>69</b>
10.1	RTEMS Linker	70
10.2	RTEMS Symbols	71
10.2.1	Symbol Table	71
10.2.2	2-Pass Linking	71
10.2.3	Command	72
10.2.4	Examples	72
10.3	RTEMS Executable Information	74
10.3.1	System Initialisation	74
10.3.2	Command	74
10.3.3	Examples	75
10.4	RTEMS BSP Builder	78
10.4.1	Developer Workflows	78
10.4.2	Build Characteristics	78
10.4.2.1	Profiles	79
10.4.2.2	Builds	79
10.4.2.2.1	All Build	80
10.4.3	Build Configurations	80
10.4.4	Performance	81
10.4.5	Command	82
10.4.5.1	Examples	83
10.5	RTEMS Tester and Run Commands	86
10.5.1	Available BSP testers	86
10.5.2	Building RTEMS Tests	87
10.5.3	Running the Tests	88
10.5.4	Test Status	90
10.5.4.1	Pass	90
10.5.4.2	Fail	90
10.5.4.3	User-input	90
10.5.4.4	Expected-fail	90

10.5.4.5 Indeterminate . . . . .	90
10.5.4.6 Benchmark . . . . .	91
10.5.4.7 Timeout . . . . .	91
10.5.4.8 Invalid . . . . .	91
10.5.5 Reporting . . . . .	91
10.5.5.1 All . . . . .	91
10.5.5.2 Failures . . . . .	91
10.5.5.3 None . . . . .	91
10.5.6 Running Tests in Parallel . . . . .	93
10.5.7 Command Line Help . . . . .	93
<b>11 Support</b>	<b>95</b>
<b>12 RTEMS Project Support</b>	<b>97</b>
12.1 Bug Tracker . . . . .	98
12.2 Documentation . . . . .	99
12.3 Mailing Lists . . . . .	100
12.4 IRC . . . . .	101
12.5 Developers . . . . .	102
<b>13 Commercial Support Services</b>	<b>103</b>
<b>14 Glossary</b>	<b>105</b>
<b>15 Index</b>	<b>107</b>



**COPYRIGHT (c) 2016-2017.**

**RTEMS Foundation, The RTEMS Documentation Project**

**License:**

Creative Commons Attribution-ShareAlike 4.0 International Public License

<https://creativecommons.org/licenses/by-sa/4.0/legalcode>

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	<a href="https://www.rtems.org/">https://www.rtems.org/</a>
Developers	<a href="https://devel.rtems.org/">https://devel.rtems.org/</a>
Documentation	<a href="https://docs.rtems.org/">https://docs.rtems.org/</a>
Bug Reporting	<a href="https://devel.rtems.org/query">https://devel.rtems.org/query</a>
Mailing Lists	<a href="https://lists.rtems.org/">https://lists.rtems.org/</a>
Git Repositories	<a href="https://git.rtems.org/">https://git.rtems.org/</a>





# OVERVIEW

Welcome to the *RTEMS* User Manual.

This document covers the topics a user of RTEMS needs to be able to install, configure, build and create applications for the RTEMS operating system.

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

## Developers

Developers should look at the [Developer Site](#) for technical information. The design and development of RTEMS is located there.

- standards based user interfaces
- multitasking capabilities
- homogeneous and heterogeneous multiprocessor systems
- event-driven, priority-based, preemptive scheduling
- optional rate monotonic scheduling
- intertask communication and synchronization
- priority inheritance
- responsive interrupt management
- dynamic memory allocation
- high level of user configurability
- open source with a friendly user license

RTEMS provides features found in modern operating systems:

- file systems
- networking
- USB
- permanent media such as flash disks, cards and USB devices
- support for various languages

- parallel programming language support

# REAL-TIME APPLICATION SYSTEMS

Real-time application systems are a special class of computer applications. They have a complex set of characteristics that distinguish them from other software problems. Generally, they must adhere to more rigorous requirements. The correctness of the system depends not only on the results of computations, but also on the time at which the results are produced. The most important and complex characteristic of real-time application systems is that they must receive and respond to a set of external stimuli within rigid and critical time constraints referred to as deadlines. Systems can be buried by an avalanche of interdependent, asynchronous or cyclical event streams.

Deadlines can be further characterized as either hard or soft based upon the value of the results when produced after the deadline has passed. A deadline is hard if the results have no value after the deadline has passed, or a catastrophic event results from their intended use if not completed on time. In contrast, results produced after a soft deadline may still have some value.

Another distinguishing requirement of real-time application systems is the ability to coordinate or manage a large number of concurrent activities. Since software is a synchronous entity, this presents special problems. One instruction follows another in a repeating synchronous cycle. Even though mechanisms have been developed to allow for the processing of external asynchronous events, the software design efforts required to process and manage these events and tasks are growing more complicated.

The design process is complicated further by spreading this activity over a set of processors instead of a single processor. The challenges associated with designing and building real-time application systems become very complex when multiple processors are involved. New requirements such as interprocessor communication channels and global resources that must be shared between competing processors are introduced. The ramifications of multiple processors complicate each and every characteristic of a real-time system.



## REAL-TIME EXECUTIVE

Fortunately, real-time operating systems, or real-time executives, serve as a cornerstone on which to build the application system. A real-time multitasking executive allows an application to be cast into a set of logical, autonomous processes or tasks which become quite manageable. Each task is internally synchronous, but different tasks execute independently, resulting in an asynchronous processing stream. Tasks can be dynamically paused for many reasons resulting in a different task being allowed to execute for a period of time. The executive also provides an interface to other system components such as interrupt handlers and device drivers. System components may request the executive to allocate and coordinate resources, and to wait for and trigger synchronizing conditions. The executive system calls effectively extend the CPU instruction set to support efficient multitasking. By causing tasks to travel through well-defined state transitions, system calls permit an application to demand-switch between tasks in response to real-time events.

By properly grouping stimuli responses into separate tasks a system can now asynchronously switch between independent streams of execution. This allows the system to directly respond to external stimuli as they occur, as well as meet critical performance specifications that are typically measured by guaranteed response time and transaction throughput. The multiprocessor extensions of RTEMS provide the features necessary to manage the extra requirements introduced by a system distributed across several processors. It removes the physical barriers of processor boundaries from the world of the system designer, enabling more critical aspects of the system to receive the required attention. Such a system, based on an efficient real-time, multiprocessor executive, is a more realistic model of the outside world or environment for which it is designed. As a result, the system will always be more logical, efficient, and reliable.

By using the directives provided by RTEMS, the real-time applications developer is freed from the problem of controlling and synchronizing multiple tasks and processors. In addition, one need not develop, test, debug, and document routines to manage memory, pass messages, or provide mutual exclusion. The developer is then able to concentrate solely on the application. By using standard software components, the time and cost required to develop sophisticated real-time applications is significantly reduced.



## RTEMS ECOSYSTEM

The RTEMS Ecosystem is the collection of tools, packages, code, documentation and online content provided by the RTEMS Project. The ecosystem provides a way to develop, maintain, and use RTEMS. It's parts interact with the user, the host environment, and each other to make RTEMS accessible, useable and predicible.

The ecosystem is for users, developers and maintainers and it is an on going effort that needs your help and support. The RTEMS project is always improving the way it delivers the kernel to you and your feedback is important so please join the mailing lists and contribute back comments, success stories, bugs and patches.

What the RTEMS project describes here to develop, maintain and use RTEMS does not dictate what you need to use in your project. You can and should select the work-flow that best suites the demands of your project and what you are delivering.

## 4.1 Rational

RTEMS is complex and the focus of the RTEMS Ecosystem is to simplify the complexity for users by providing a stable documented way to build, configure and run RTEMS. RTEMS is more than a kernel running real-time applications on target hardware, it is part of a project's and therefore team's workflow and every project and team is different.

RTEMS's ecosystem does not mandate a way to work. It is a series of parts, components, and items that are used to create a suitable development environment to work with. The processes explained in this manual are the same things an RTEMS maintainer does to maintain the kernel or an experienced user does to build their production system. It is important to keep this in mind when working through this manual. We encourage users to explore what can be done and to discover ways to make it fit their needs. The ecosystem provided by the RTEMS Project will not install in a single click of a mouse because we want users to learn the parts they will come to depend on as their project's development matures.

The RTEMS Ecosystem provides a standard interface that is the same on all supported host systems. Standardizing how a user interacts with RTEMS is important and making that experience portable is also important. As a result the ecosystem is documented at the command line level and we leave GUI and IDE integration for users and integrators.

Standardizing the parts and how to use them lets users create processes and procedures that are stable over releases. The RTEMS Ecosystem generates data that can be used to audit the build process so their configuration can be documented.

The ecosystem is based around the source code used in the various parts, components and items of the RTEMS development environment. A user can create an archive of the complete build process including all the source code for long term storage. This is important for projects with a long life cycle.



## 4.2 Open Source

RTEMS is an open source operating system and an open source project and this extends to the ecosystem. We encourage users to integrate the processes to build tools, the kernel and any 3rd party libraries into their project's configuration management processes.

All the parts that make up the ecosystem are open source. The ecosystem uses a package's source code to create an executable on a host so when an example RTEMS executable is created and run for the first time the user will have built every tool as well as the executable from source. The RTEMS Project believes the freedom this gives a user is as important as the freedom of having access to the source code for a package.

## 4.3 Deployment

The RTEMS Project provides the ecosystem as source code that users can download to create personalised development environments. The RTEMS Project does not provide packaging and deployment for a specific host environment, target architecture or BSP. The RTEMS Project encourages users and organizations to fill this role for the community.

---

# QUICK START

The following is a quick start guide that provides a basic set of commands to build the RTEMS Tools and Kernel. The quick start guide provides links to the detailed sections if any problems are encountered.

The detailed procedure for installing RTEMS can be found in *Chapter 7 - Installation*.

The development host computer needs to be set up for this quick start procedure to complete successfully. *Chapter 6 - Host Computer* details what is needed for the supported host operating systems. If Windows is being used it is recommended following the procedure in *Chapter 6 Section 4 - Microsoft Windows* first.

There are many ways and locations a suitable environment can be set up. A common factor that defines the final location of tools and projects is the place you have suitable storage. *Chapter 7 Section 1 - Prefixes* and *Chapter 7 Section 1.1 - Project Sandboxing* provide detailed examples of possible locations and set ups .

This procedure installs a developer set up using the RTEMS Git repositories on a Unix (POSIX) or MacOS host. The output from the commands has been removed and replaced with . . .

Create a workspace, download the RTEMS Source Builder (RSB) and build a tool chain (*Chapter 7 Section 3.1 - POSIX and OS X Host Tools Chain*):

```
1 $ cd
2 $ mkdir -p development/rtems
3 $ cd development/rtems
4 $ git clone git://git.rtems.org/rtems-source-builder.git rsb
5   ...
6 $ cd rsb
7 $ ./source-builder/sb-check
8   ...
9 $ cd rtems
10 $ ../source-builder/sb-set-builder \
11     --prefix=/usr/home/chris/development/rtems/5 5/rtems-sparc
12   ...
```

Build the RTEMS Kernel (*Chapter 7 Section 3.2.2 - Building the Kernel*) by cloning the repository, running the bootstrap procedure, building and finally installing the kernel:

```
1 $ export PATH=$HOME/development/rtems/5/bin:$PATH
2 $ cd
3 $ cd development/rtems
4 $ mkdir kernel
5 $ cd kernel
```

```
6 $ git clone git://git.rtems.org/rtems.git rtems
7   ...
8 $ cd rtems
9 $ ./bootstrap -c && ./bootstrap -p && \
10     $HOME/development/rtems/rsb/source-builder/sb-bootstrap
11   ...
12 $ cd ..
13 $ mkdir erc32
14 $ cd erc32
15 $ $HOME/development/rtems/kernel/rtems/configure --prefix=$HOME/development/rtems/5 \
16     --target=sparc-rtems5 --enable-rtemsbsp=erc32 --enable-posix
17   ...
18 $ make -j 8
19   ...
20 $ make install
```

You can now build a 3rd party library or an application.

## HOST COMPUTER

RTEMS applications are developed using cross-development tools running on a development computer, more often called the host computer. These are typically your desktop machine or a special build server. All RTEMS tools and runtime libraries are built from source on your host machine. The RTEMS Project does not maintain binary builds of the tools. This differs to what you normally experience with host operating systems, and it is, however this approach works well. RTEMS is not a host operating system and it is not a distribution. Deploying binary packages for every possible host operating system is too big a task for the RTEMS Project and it is not a good use of core developer time. Their time is better spent making RTEMS better and faster.

The RTEMS Project's aim is to give you complete freedom to decide on the languages used in your project, which version control system, and the build system for your application.

The rule for selecting a computer for a developer is *more is better* but we do understand there are limits. Projects set up different configurations, some have a development machine per developer while others set up a tightly controlled central build server. RTEMS Ecosystem is flexible and lets you engineer a development environment that suites you. The basic specs are:

- Multicore processor
- 8G bytes RAM
- 256G harddisk

RTEMS makes no demands on graphics.

If you are using a VM or your host computer is not a fast modern machine do not be concerned. The tools may take longer to build than faster hardware however building tools is something you do once. Once the tools and RTEMS is built all your time can be spent writing and developing your application. Over an hour can happen and for the ARM architecture and with all BSPs it can be many hours.

## 6.1 Host Operating Systems

### *GDB and Python*

RTEMS uses Python in GDB to aid debugging which means GDB needs to be built with Python development libraries. Please check the RSB documentation and install the packages specified for your host. Make sure a python development package is included.

A wide range of host operating systems and hardware can be used. The host operating systems supported are:

- Linux
- FreeBSD
- NetBSD
- Apple OS X
- Windows
- Solaris

The functionality on a POSIX operating such as Linux and FreeBSD is similar and most features on Windows are supported but you are best to ask on the [Users Mailing List](#) if you have a specific question.

We recommend you maintain your operating system by installing any updates.

## 6.2 POSIX Hosts

POSIX hosts are most Unix operating systems such as Linux, FreeBSD and NetBSD. RTEMS development works well on Unix and can scale from a single user and a desktop machine to a team with decentralised or centralised development infrastructure.

### 6.2.1 Root Access

You either have root access to your host development machine or you do not. Some users are given hardware that is centrally managed. If you do not have root access you can create your work environment in your home directory. You could use a prefix of `$HOME/development/rtems` or `$HOME/rtems`. Note, the `$HOME` environment variable can be substituted with `~`.

*Chapter 7 Section 1 - Prefixes* details using Prefixes to manage the installation.

RTEMS Tools and packages do not require root access to be built and we encourage you to not build the tools as root. If you need to control write access then it is best to manage this with groups assigned to users.

If you have root access you can decide to install the tools under any suitable prefix. This may depend on the hardware in your host development machine. If the machine is a centralised build server the prefix may be used to separate production versions from the test versions and the prefix paths may have restricted access rights to only those who manage and have configuration control of the machine. We call this project sandboxing and *Chapter 7 Section 1.1 - Project Sandboxing* explains this in more detail.

## 6.3 Apple OS X

Apple's OS X is fully supported. You need to download and install a recent version of the Apple developer application Xcode. Xcode is available in the App Store. Make sure you install the Command Line Tools add on available for download within Xcode and once installed open a Terminal shell and enter the command `cc` and accept the license agreement.

The normal prefix when working on OS X as a user is under your home directory. Prefixes of `$HOME/development/rtems` or `$HOME/rtems` are suitable.

*Chapter 7 Section 1 - Prefixes* details using Prefixes to manage the installation.



## 6.4 Microsoft Windows

RTEMS supports Windows as a development host and the tools for most architectures are available. The RTEMS Project relies on the GNU tools for compilers and debuggers and we use the simulators that come with GDB and QEMU. The Windows support for these tools varies and the RTEMS Project is committed to helping the open source community improve the Windows experience. If something is not working or supported please email the [Users Mailing List](#).

The RTEMS Project's Windows tools are native Windows executables giving the user the best possible experience on Windows. Native Windows programs use the standard Windows DLLs and paths. Integration with standard Windows integrated development tools such as editors is straight forward. POSIX emulation environments such as Cygwin and the MSYS2 shell have special executables that require a POSIX emulation DLL and these emulation DLLs add an extra layer of complexity as well as a performance over-head. The RTEMS Project uses these POSIX emulation shells to run configure scripts that come with various open source packages such as gcc so they form an important and valued part of the environment we describe here. The output of this procedure forms the tools you use during your application development and they do not depend on the emulation DLLs.

The performance of the compiler is as good as you can have on Windows and the performance compiling a single file will be similar to that on a host like Linux or FreeBSD given the same hardware. Building the tools from source is much slower on Windows because POSIX shells and related tools are used and the POSIX emulation overhead it much much slower than a native POSIX operating system like Linux and FreeBSD. This overhead is only during the building of the tools and the RTEMS kernel and if you use a suitable build system that is native to Windows your application development should be similar to other operating systems.

Building is known to work on *Windows 7 64bit Professional* and *Windows 10*.

### 6.4.1 Windows Path Length

Windows path length is limited and can cause problems when building the tools. The standard Windows API has a MAX\_PATH length of 260 characters. This can effect some of the tools used by RTEMS. It is recommended you keep the top level directories as short as possible when building the RTEMS tools and you should also keep an eye on the path length when developing your application. The RTEMS built tools can handle much longer path lengths however some of the GNU tools such as those in the `binutils` package cannot.

The release packages of the RSB when unpacked have top level file names that are too big to build RTEMS. You need to change or rename that path to something smaller to build. This is indicated in *Chapter 7 Section 2 - Releases*.

### 6.4.2 Parallel Builds with Make

The MSYS2 GNU make has problems when using the `jobs` option. The RSB defaults to automatically using as many cores as the host machine has. To get a successful build on Windows it is recommended you add the `--jobs=none` option to all RSB build set commands.

### 6.4.3 POSIX Support

Building the RTEMS compilers, debugger, the RTEMS kernel and a number of other 3rd party packages requires a POSIX environment. On Windows you can use Cygwin or MSYS2. This document focuses on MSYS2. It is smaller than Cygwin and comes with the Arch Linux package manager pacman.

MSYS2 provides MinGW64 support as well as a POSIX shell called MSYS2. The MinGW64 compiler and related tools produce 64bit native Windows executables. The shell is a standard Bourne shell and the MSYS2 environment is a stripped Cygwin shell with enough support to run the various configure scripts needed to build the RTEMS tools and the RTEMS kernel.

MSYS2 is built around the pacman packaging tool. This makes MSYS2 a distribution and that is a welcome feature on Windows. You get a powerful tool to manage your development environment on Windows.

### 6.4.4 Python

We need Python to build the tools as the RSB is written in Python and we need suitable Python libraries to link to GDB as RTEMS makes use of GDB's Python support. This places specific demands on the Python we need installed and available and MSYS2 provides suitable Python versions we can use. You need to make sure you have the correct type and version of Python installed.

We cannot use the Python executables created by the Python project (python.org) as they are built by Microsoft's C (MSC) compiler. Linking the MSC Python libraries with the MinGW64 executables is not easy and MSYS provides us with a simple solution so we do not support linking MSC libraries.

MSYS2 provides two types and two versions of Python executables, MinGW and MSYS and Python version 2 and 3. For Windows we need the MinGW executable so we have suitable libraries and we have to have Python version 2 because on Windows GDB only builds with Python2.

You also need to install the MSYS version of Python along with the MinGW64 Python2 package. The MSYS Python is version 3 and the RSB can support version 2 and 3 of Python and it helps handle some of the long paths building GCC can generate.

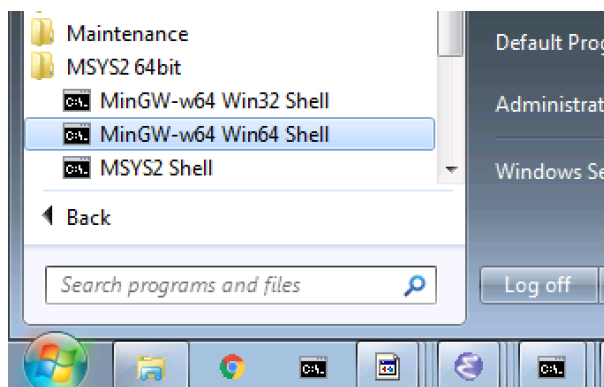
### 6.4.5 Installing MSYS2

MSYS2 is installed on a new machine using the MSYS2 installer found on <https://msys2.github.io/>. Please select the x86\_64 variant for 64bit support. Run the installer following the 7 steps listed on the page.

MSYS2 uses the pacman package manager. The Arch Linux project has detailed documentation on how to use pacman. What is shown here is a just few examples of what you can do.

#### **Pin MSYS2 Shell to Taskbar**

Pin the MSYS2 64bit Shell to the Taskbar so you always use it rather than the 32bit Shell.



Open a 64bit MSYS shell from the Start Menu:

The packages we require are:

- python
- mingw-w64-x86\_64-python2
- mingw-w64-x86\_64-gcc
- git
- bison
- cvs
- diffutils
- make
- patch
- tar
- texinfo
- unzip

---

**Note:** The actual output provided may vary due to changes in the dependent packages or newer package versions.

---

Install the packages using pacman:

```

1 ~
2 $ pacman -S python mingw-w64-x86_64-python2 mingw-w64-x86_64-gcc \
3 bison cvs diffutils git make patch tar texinfo unzip
4 resolving dependencies...
5 looking for conflicting packages...
6
7 Packages (74) db-5.3.28-2 expat-2.1.0-2 gdbm-1.11-3 heimdal-1.5.3-8
8                libgdbm-1.11-3 mingw-w64-x86_64-binutils-2.26-21
9                mingw-w64-x86_64-bzip2-1.0.6-5
10               mingw-w64-x86_64-ca-certificates-20150426-2
11               mingw-w64-x86_64-crt-git-5.0.0.4627.03684c4-1
12               mingw-w64-x86_64-expat-2.1.0-6 mingw-w64-x86_64-gcc-libs-5.3.0-2
13               mingw-w64-x86_64-gdbm-1.11-3 mingw-w64-x86_64-gettext-0.19.6-2
  
```

```

14      mingw-w64-x86_64-gmp-6.1.0-1
15      mingw-w64-x86_64-headers-git-5.0.0.4627.53be55d-1
16      mingw-w64-x86_64-isl-0.15-1 mingw-w64-x86_64-libffi-3.2.1-3
17      mingw-w64-x86_64-libiconv-1.14-5
18      mingw-w64-x86_64-libsystre-1.0.1-2
19      mingw-w64-x86_64-libtasn1-4.7-1
20      mingw-w64-x86_64-libtre-git-r122.c2f5d13-4
21      mingw-w64-x86_64-libwinpthread-git-5.0.0.4573.628fdbf-1
22      mingw-w64-x86_64-mpc-1.0.3-2 mingw-w64-x86_64-mpfr-3.1.3.p0-2
23      mingw-w64-x86_64-ncurses-6.0.20160220-2
24      mingw-w64-x86_64-openssl-1.0.2.g-1
25      mingw-w64-x86_64-p11-kit-0.23.1-3
26      mingw-w64-x86_64-readline-6.3.008-1 mingw-w64-x86_64-tcl-8.6.5-1
27      mingw-w64-x86_64-termcap-1.3.1-2 mingw-w64-x86_64-tk-8.6.5-1
28      mingw-w64-x86_64-windows-default-manifest-6.4-2
29      mingw-w64-x86_64-winpthreads-git-5.0.0.4573.628fdbf-1
30      mingw-w64-x86_64-zlib-1.2.8-9 openssl-7.1p2-1 perl-5.22.0-2
31      perl-Authen-SASL-2.16-2 perl-Convert-BinHex-1.123-2
32      perl-Encode-Locale-1.04-1 perl-Error-0.17024-1
33      perl-File-Listing-6.04-2 perl-HTML-Parser-3.71-3
34      perl-HTML-Tagset-3.20-2 perl-HTTP-Cookies-6.01-2
35      perl-HTTP-Daemon-6.01-2 perl-HTTP-Date-6.02-2
36      perl-HTTP-Message-6.06-2 perl-HTTP-Negotiate-6.01-2
37      perl-IO-Socket-SSL-2.016-1 perl-IO-stringy-2.111-1
38      perl-LWP-MediaTypes-6.02-2 perl-MIME-tools-5.506-1
39      perl-MailTools-2.14-1 perl-Net-HTTP-6.09-1
40      perl-Net-SMTP-SSL-1.02-1 perl-Net-SSLeay-1.70-1
41      perl-TermReadKey-2.33-1 perl-TimeDate-2.30-2 perl-URI-1.68-1
42      perl-WWW-RobotRules-6.02-2 perl-libwww-6.13-1 vim-7.4.1468-1
43      bison-3.0.4-1 cvs-1.11.23-2 diffutils-3.3-3 git-2.7.2-1
44      make-4.1-4 mingw-w64-x86_64-gcc-5.3.0-2
45      mingw-w64-x86_64-python2-2.7.11-4 patch-2.7.5-1 python-3.4.3-3
46      tar-1.28-3 texinfo-6.0-1 unzip-6.0-2
47
48 Total Download Size: 114.10 MiB
49 Total Installed Size: 689.61 MiB
50
51 :: Proceed with installation? [Y/n] y
52 :: Retrieving packages...
53 mingw-w64-x86_64-gm... 477.1 KiB 681K/s 00:01 [#####] 100%
54 mingw-w64-x86_64-li... 24.2 KiB 755K/s 00:00 [#####] 100%
55 mingw-w64-x86_64-gc... 541.9 KiB 705K/s 00:01 [#####] 100%
56 mingw-w64-x86_64-ex... 106.7 KiB 702K/s 00:00 [#####] 100%
57 mingw-w64-x86_64-bz... 77.9 KiB 666K/s 00:00 [#####] 100%
58 mingw-w64-x86_64-li... 600.2 KiB 703K/s 00:01 [#####] 100%
59 mingw-w64-x86_64-ge... 3.0 MiB 700K/s 00:04 [#####] 100%
60 mingw-w64-x86_64-gd... 151.8 KiB 483K/s 00:00 [#####] 100%
61 mingw-w64-x86_64-li... 34.5 KiB 705K/s 00:00 [#####] 100%
62 mingw-w64-x86_64-li... 69.2 KiB 713K/s 00:00 [#####] 100%
63 mingw-w64-x86_64-li... 9.3 KiB 778K/s 00:00 [#####] 100%
64 mingw-w64-x86_64-nc... 1800.5 KiB 701K/s 00:03 [#####] 100%
65 mingw-w64-x86_64-li... 171.4 KiB 708K/s 00:00 [#####] 100%
66 mingw-w64-x86_64-p1... 193.5 KiB 709K/s 00:00 [#####] 100%
67 mingw-w64-x86_64-ca... 382.1 KiB 705K/s 00:01 [#####] 100%
68 mingw-w64-x86_64-zl... 148.6 KiB 704K/s 00:00 [#####] 100%
69 mingw-w64-x86_64-op... 3.3 MiB 624K/s 00:05 [#####] 100%
70 mingw-w64-x86_64-te... 12.6 KiB 76.7K/s 00:00 [#####] 100%

```

71	mingw-w64-x86_64-re...	327.4 KiB	277K/s	00:01	[#####]	100%
72	mingw-w64-x86_64-tc...	2.9 MiB	699K/s	00:04	[#####]	100%
73	mingw-w64-x86_64-tk...	1869.2 KiB	703K/s	00:03	[#####]	100%
74	mingw-w64-x86_64-py...	10.9 MiB	699K/s	00:16	[#####]	100%
75	mingw-w64-x86_64-bi...	12.7 MiB	688K/s	00:19	[#####]	100%
76	mingw-w64-x86_64-he...	5.0 MiB	645K/s	00:08	[#####]	100%
77	mingw-w64-x86_64-cr...	2.6 MiB	701K/s	00:04	[#####]	100%
78	mingw-w64-x86_64-is...	524.3 KiB	684K/s	00:01	[#####]	100%
79	mingw-w64-x86_64-mp...	265.2 KiB	705K/s	00:00	[#####]	100%
80	mingw-w64-x86_64-mp...	62.3 KiB	82.9K/s	00:01	[#####]	100%
81	mingw-w64-x86_64-wi...	1484.0 B	0.00B/s	00:00	[#####]	100%
82	mingw-w64-x86_64-wi...	33.2 KiB	346K/s	00:00	[#####]	100%
83	mingw-w64-x86_64-gc...	25.1 MiB	701K/s	00:37	[#####]	100%
84	python-3.4.3-3-x86_64	12.1 MiB	700K/s	00:18	[#####]	100%
85	bison-3.0.4-1-x86_64	1045.1 KiB	703K/s	00:01	[#####]	100%
86	heimdal-1.5.3-8-x86_64	543.7 KiB	703K/s	00:01	[#####]	100%
87	cvs-1.11.23-2-x86_64	508.2 KiB	388K/s	00:01	[#####]	100%
88	diffutils-3.3-3-x86_64	265.7 KiB	478K/s	00:01	[#####]	100%
89	expat-2.1.0-2-x86_64	13.1 KiB	817K/s	00:00	[#####]	100%
90	vim-7.4.1468-1-x86_64	6.1 MiB	700K/s	00:09	[#####]	100%
91	openssh-7.1p2-1-x86_64	653.4 KiB	703K/s	00:01	[#####]	100%
92	db-5.3.28-2-x86_64	41.7 KiB	719K/s	00:00	[#####]	100%
93	libgdbm-1.11-3-x86_64	20.4 KiB	754K/s	00:00	[#####]	100%
94	gdbm-1.11-3-x86_64	108.5 KiB	704K/s	00:00	[#####]	100%
95	perl-5.22.0-2-x86_64	12.4 MiB	702K/s	00:18	[#####]	100%
96	perl-Error-0.17024-...	17.1 KiB	742K/s	00:00	[#####]	100%
97	perl-Authen-SASL-2....	42.4 KiB	731K/s	00:00	[#####]	100%
98	perl-Encode-Locale-...	9.7 KiB	745K/s	00:00	[#####]	100%
99	perl-HTTP-Date-6.02...	8.6 KiB	784K/s	00:00	[#####]	100%
100	perl-File-Listing-6...	7.7 KiB	769K/s	00:00	[#####]	100%
101	perl-HTML-Tagset-3....	10.3 KiB	732K/s	00:00	[#####]	100%
102	perl-HTML-Parser-3....	76.9 KiB	516K/s	00:00	[#####]	100%
103	perl-LWP-MediaTypes...	18.0 KiB	752K/s	00:00	[#####]	100%
104	perl-URI-1.68-1-any	75.6 KiB	609K/s	00:00	[#####]	100%
105	perl-HTTP-Message-6...	71.3 KiB	625K/s	00:00	[#####]	100%
106	perl-HTTP-Cookies-6...	20.4 KiB	499K/s	00:00	[#####]	100%
107	perl-HTTP-Daemon-6....	14.2 KiB	749K/s	00:00	[#####]	100%
108	perl-HTTP-Negotiate...	11.4 KiB	817K/s	00:00	[#####]	100%
109	perl-Net-HTTP-6.09-...	19.8 KiB	732K/s	00:00	[#####]	100%
110	perl-WWW-RobotRules...	12.2 KiB	766K/s	00:00	[#####]	100%
111	perl-libwww-6.13-1-any	122.2 KiB	661K/s	00:00	[#####]	100%
112	perl-TimeDate-2.30-...	35.9 KiB	718K/s	00:00	[#####]	100%
113	perl-MailTools-2.14...	58.4 KiB	712K/s	00:00	[#####]	100%
114	perl-IO-stringy-2.1...	52.6 KiB	721K/s	00:00	[#####]	100%
115	perl-Convert-BinHex...	30.1 KiB	733K/s	00:00	[#####]	100%
116	perl-MIME-tools-5.5...	180.4 KiB	705K/s	00:00	[#####]	100%
117	perl-Net-SSLeay-1.7...	191.2 KiB	708K/s	00:00	[#####]	100%
118	perl-IO-Socket-SSL-...	112.5 KiB	703K/s	00:00	[#####]	100%
119	perl-Net-SMTP-SSL-1...	3.5 KiB	881K/s	00:00	[#####]	100%
120	perl-TermReadKey-2....	20.9 KiB	745K/s	00:00	[#####]	100%
121	git-2.7.2-1-x86_64	3.6 MiB	702K/s	00:05	[#####]	100%
122	make-4.1-4-x86_64	387.0 KiB	671K/s	00:01	[#####]	100%
123	patch-2.7.5-1-x86_64	75.9 KiB	684K/s	00:00	[#####]	100%
124	tar-1.28-3-x86_64	671.9 KiB	379K/s	00:02	[#####]	100%
125	texinfo-6.0-1-x86_64	992.7 KiB	625K/s	00:02	[#####]	100%
126	unzip-6.0-2-x86_64	93.1 KiB	705K/s	00:00	[#####]	100%
127	(74/74) checking keys in keyring				[#####]	100%

```

128 (74/74) checking package integrity [#####] 100%
129 (74/74) loading package files [#####] 100%
130 (74/74) checking for file conflicts [#####] 100%
131 (74/74) checking available disk space [#####] 100%
132 :: Processing package changes...
133 ( 1/74) installing python [#####] 100%
134 ( 2/74) installing mingw-w64-x86_64-gmp [#####] 100%
135 ( 3/74) installing mingw-w64-x86_64-libwinpthr... [#####] 100%
136 ( 4/74) installing mingw-w64-x86_64-gcc-libs [#####] 100%
137 ( 5/74) installing mingw-w64-x86_64-expat [#####] 100%
138 ( 6/74) installing mingw-w64-x86_64-bzip2 [#####] 100%
139 ( 7/74) installing mingw-w64-x86_64-libiconv [#####] 100%
140 ( 8/74) installing mingw-w64-x86_64-gettext [#####] 100%
141 ( 9/74) installing mingw-w64-x86_64-gdbm [#####] 100%
142 (10/74) installing mingw-w64-x86_64-libffi [#####] 100%
143 (11/74) installing mingw-w64-x86_64-libtre-git [#####] 100%
144 (12/74) installing mingw-w64-x86_64-libsyste [#####] 100%
145 (13/74) installing mingw-w64-x86_64-ncurses [#####] 100%
146 (14/74) installing mingw-w64-x86_64-libtasn1 [#####] 100%
147 (15/74) installing mingw-w64-x86_64-p11-kit [#####] 100%
148 (16/74) installing mingw-w64-x86_64-ca-certifi... [#####] 100%
149 (17/74) installing mingw-w64-x86_64-zlib [#####] 100%
150 (18/74) installing mingw-w64-x86_64-openssl [#####] 100%
151 (19/74) installing mingw-w64-x86_64-termcap [#####] 100%
152 (20/74) installing mingw-w64-x86_64-readline [#####] 100%
153 (21/74) installing mingw-w64-x86_64-tcl [#####] 100%
154 (22/74) installing mingw-w64-x86_64-tk [#####] 100%
155 (23/74) installing mingw-w64-x86_64-python2 [#####] 100%
156 (24/74) installing mingw-w64-x86_64-binutils [#####] 100%
157 (25/74) installing mingw-w64-x86_64-headers-git [#####] 100%
158 (26/74) installing mingw-w64-x86_64-crt-git [#####] 100%
159 (27/74) installing mingw-w64-x86_64-isl [#####] 100%
160 (28/74) installing mingw-w64-x86_64-mpfr [#####] 100%
161 (29/74) installing mingw-w64-x86_64-mpc [#####] 100%
162 (30/74) installing mingw-w64-x86_64-windows-de... [#####] 100%
163 (31/74) installing mingw-w64-x86_64-winpthread... [#####] 100%
164 (32/74) installing mingw-w64-x86_64-gcc [#####] 100%
165 (33/74) installing bison [#####] 100%
166 (34/74) installing heimdal [#####] 100%
167 (35/74) installing cvs [#####] 100%
168 (36/74) installing diffutils [#####] 100%
169 (37/74) installing expat [#####] 100%
170 (38/74) installing vim [#####] 100%
171 (39/74) installing openssh [#####] 100%
172 (40/74) installing db [#####] 100%
173 (41/74) installing libgdbm [#####] 100%
174 (42/74) installing gdbm [#####] 100%
175 (43/74) installing perl [#####] 100%
176 (44/74) installing perl-Error [#####] 100%
177 (45/74) installing perl-Authen-SASL [#####] 100%
178 (46/74) installing perl-Encode-Locale [#####] 100%
179 (47/74) installing perl-HTTP-Date [#####] 100%
180 (48/74) installing perl-File-Listing [#####] 100%
181 (49/74) installing perl-HTML-Tagset [#####] 100%
182 (50/74) installing perl-HTML-Parser [#####] 100%
183 (51/74) installing perl-LWP-MediaTypes [#####] 100%
184 (52/74) installing perl-URI [#####] 100%

```



```

185 (53/74) installing perl-HTTP-Message [#####] 100%
186 (54/74) installing perl-HTTP-Cookies [#####] 100%
187 (55/74) installing perl-HTTP-Daemon [#####] 100%
188 (56/74) installing perl-HTTP-Negotiate [#####] 100%
189 (57/74) installing perl-Net-HTTP [#####] 100%
190 (58/74) installing perl-WWW-RobotRules [#####] 100%
191 (59/74) installing perl-libwww [#####] 100%
192 Optional dependencies for perl-libwww
193 perl-LWP-Protocol-HTTPS: for https:// url schemes
194 (60/74) installing perl-TimeDate [#####] 100%
195 (61/74) installing perl-MailTools [#####] 100%
196 (62/74) installing perl-IO-stringy [#####] 100%
197 (63/74) installing perl-Convert-BinHex [#####] 100%
198 module test... pass.
199 (64/74) installing perl-MIME-tools [#####] 100%
200 (65/74) installing perl-Net-SSLeay [#####] 100%
201 (66/74) installing perl-IO-Socket-SSL [#####] 100%
202 (67/74) installing perl-Net-SMTP-SSL [#####] 100%
203 (68/74) installing perl-TermReadKey [#####] 100%
204 (69/74) installing git [#####] 100%
205 Optional dependencies for git
206 python2: various helper scripts
207 subversion: git svn
208 (70/74) installing make [#####] 100%
209 (71/74) installing patch [#####] 100%
210 Optional dependencies for patch
211 ed: for patch -e functionality
212 (72/74) installing tar [#####] 100%
213 (73/74) installing texinfo [#####] 100%
214 (74/74) installing unzip [#####] 100%

```





## INSTALLATION

This section details how to set up and install the RTEMS Ecosystem. You will create a set of tools and an RTEMS kernel for your selected Board Support Package (BSP).

You will be asked to follow a few simple steps and when you have finished you will have a development environment set up you can use to build applications for RTEMS. You will have also created a development environment you and a team can adapt for a project of any size and complexity.

RTEMS applications are developed using cross-development tools running on a development computer, more commonly referred to as the host computer. These are typically your desktop machine or a special build server. All RTEMS tools and runtime libraries are built from source on your host machine. The RTEMS Project does not maintain binary builds of the tools. This may appear to be the opposite to what you normally experience with host operating systems, and it is, however this approach works well. RTEMS is not a host operating system and it is not a distribution. Providing binary packages for every possible host operating system is too big a task for the RTEMS Project and it is not a good use of core developer time. Their time is better spent making RTEMS better and faster.

The RTEMS Project base installation set ups the tools and the RTEMS kernel for the selected BSPs. The tools run on your host computer are used to compile, link, and format executables so they can run on your target hardware.

The RTEMS Project supports two set ups, release and developer environments. Release installations create the tools and kernel in a single pass ready for you to use. The tools and kernel are stable and only bug fixes are added creating new dot point releases. The developer set up tracks the Git repositories for the tools and kernel.

## 7.1 Prefixes

You will see the term *prefix* referred to throughout this documentation and in a wide number of software packages you can download from the internet. A **prefix** is the path on your computer a software package is built and installed under. Packages that have a **prefix** will place all parts under the **prefix** path. On a host computer like Linux the packages you install from your distribution typically use a platform specific standard **prefix**. For example on Linux it is `/usr` and on FreeBSD it is `/usr/local`.

We recommend you *DO NOT* use the standard **prefix** when installing the RTEMS Tools. The standard **prefix** is the default **prefix** each package built by the RSB contains. If you are building the tools when logged in as a *Standard User* and not as the *Super User* (`root`) or *Administrator* the RTEMS Source Builder (RSB) *will* fail and report an error if the default **prefix** is not writable. We recommend you leave the standard **prefix** for the packages your operating system installs or software you manually install such as applications.

A further reason not to use the standard **prefix** is to allow more than one version of RTEMS to exist on your host machine at a time. The `autoconf` and `automake` tools required by RTEMS are not versioned and vary between the various versions of RTEMS. If you use a single **prefix** such as the standard **prefix** there is a chance parts from a package of different versions may interact. This should not happen but it can.

For POSIX or Unix hosts, the RTEMS Project uses `/opt/rtems` as its standard **prefix**. We view this **prefix** as a production level path, and we prefer to place development versions under a different **prefix** away from the production versions. Under this top level **prefix** we place the various versions we need for development. For example the version 4.11.0 **prefix** would be `/opt/rtems/4.11.0`. If an update called 4.11.1 is released the **prefix** would be `/opt/rtems/4.11.1`. These are recommendations and the choice of what you use is entirely yours. You may decide to have a single path for all RTEMS 4.11 releases of `/opt/rtems/4.11`.

For Windows a typical **prefix** is `C:\opt\rtems` and as an MSYS2 path this is `/c/opt/rtems`.

### 7.1.1 Project Sandboxing

Project specific sandboxes let you have a number of projects running in parallel with each project in its own sandbox. You simply have a *prefix* per project and under that prefix you create a simple yet repeatable structure.

As an example lets say I have a large disk mounted under `/bd` for *Big Disk*. As root create a directory called `projects` and give the directory suitable permissions to be writable by you as a user.

Lets create a project sandbox for my *Box Sorter* project. First create a project directory called `/bd/projects/box-sorter`. Under this create `rtems` and under that create `rtems-4.11.0`. Under this path you can follow the *Chapter 7 Section 2 - Releases* procedure to build a tool set using the prefix of `/bd/projects/box-sorter/rtems/4.11.0`. You are free to create your project specific directories under `/bd/projects/box-sorter`. The top level directories would be:

#### **/bd/projects**

Project specific development trees.

#### **/bd/projects/box-sorter**

Box Sorter project sandbox.

**/bd/projects/box-sorter/rtems/4.11.0**

Project prefix for RTEMS 4.11.0 compiler, debuggers, tools and installed Board Support Package (BSP).

A variation is to use the `--without-rtems` option with the RSB to not build the BSPs when building the tools and to build RTEMS specifically for each project. This lets you have a production tools installed at a top level on your disk and each project can have a specific and possibly customised version of RTEMS. The top level directories would be:

**/bd/rtems**

The top path to production tools.

**/bd/rtems/4.11.0**

Production prefix for RTEMS 4.11.0 compiler, debuggers and tools.

**/bd/projects**

Project specific development trees.

**/bd/projects/box-sorter**

Box Sorter project sandbox.

**/bd/projects/box-sorter/rtems**

Box Sorter project's custom RTEMS kernel source and installed BSP.

A further variation if there is an RTEMS kernel you want to share between projects is it to move this to a top level and share. In this case you will end up with:

**/bd/rtems**

The top path to production tools and kernels.

**/bd/rtems/4.11.0**

Production prefix for RTEMS 4.11.0.

**/bd/rtems/4.11.0/tools**

Production prefix for RTEMS 4.11.0 compiler, debuggers and tools.

**/bd/rtems/4.11.0/bsps**

Production prefix for RTEMS 4.11.0 Board Support Packages (BSPs).

**/bd/projects**

Project specific development trees.

**/bd/projects/box-sorter**

Box Sorter project sandbox.

Finally you can have a single set of *production* tools and RTEMS BSPs on the disk under `/bd/rtems` you can share between your projects. The top level directories would be:

**/bd/rtems**

The top path to production tools and kernels.

**/bd/rtems/4.11.0**

Production prefix for RTEMS 4.11.0 compiler, debuggers, tools and Board Support Packages (BSPs).

**/bd/projects**

Project specific development trees.

**/bd/projects/box-sorter**

Box Sorter project sandbox.

The project sandoxing approach allows you move a specific production part into the project's sandbox to allow you to customise it. This is useful if you are testing new releases. The typical dependency is the order listed above. You can test new RTEMS kernels with production tools but new tools will require you build the kernel with them. Release notes with each release will let know what you need to update.

If the machine is a central project development machine simply replace projects with users and give each user a personal directory.

## 7.2 Releases

RTEMS releases provide a stable version of the kernel for the supported architectures. RTEMS maintains the current and previous releases. Support for older releases is provided using the RTEMS support channels.

Please read *Chapter 6 - Host Computer* before continuing. The following procedure assumes you have installed and configured your host operating. It also assumes you have installed any dependent packages needed when building the tools and the kernel.

You need to select a location to build and install the RTEMS Tool chain and RTEMS. Make sure there is plenty of disk space and a fast disk is recommended. Our procedure will document building and installing the tools in a base directory called `/opt/rtems`. This path will require root access. If you are working on a machine you do not have root access to you can use a home directory, If building on Windows use `/c/opt/rtems` to keep the top level paths as short as possible. *Chapter 6 Section 4.1 - Windows Path Length* provides more detail about path lengths on Windows.

The location used to install the tools and kernel is called the *prefix*. *Chapter 7 Section 1 - Prefixes* explains prefixes and how to use them. It is best to have a *prefix* for each different version of RTEMS you are using. If you are using RTEMS 4.11 in production it is **not** a good idea to install a development version of 5 over the top by using the same *prefix* as the 4.11 build. A separate *prefix* for each version avoids this.

Released versions of the RTEMS Source Builder (RSB) downloads all source code for all packages from the [FTP File Server](#) rather than from the package's home site. Hosting all the source on the [FTP File Server](#) ensures the source is present for the life of the release on the [FTP File Server](#). If there is a problem accessing the RTEMS FTP the RSB will fall back to the packages home site.

The [FTP File Server](#) is hosted at the Oregon State University's The Open Source Lab (<http://osuosl.org/>). This is a nonprofit organization working for the advancement of open source technologies and RTEMS is very fortunate to be shosted here. It has excellent internet access and performance.

---

### Note: Controlling the RTEMS Kernel Build

Building releases by default builds the RTEMS kernel. To not build the RTEMS kernel add the `--without-rtems` option to the RSB command line.

By default all the BSPs for an architecture are built. If you only wish to have a specific BSP built you can specify the BSP list by providing to the RSB the option `--with-rtemsbsp`. For example to build two BSPs for the SPARC architecture you can supply `--with-rtemsbsp="erc32 leon3"`. This can speed the build time up for some architectures that have a lot of BSPs.

---

Once you have built the tools and kernel you can move to the Packages section of the manual.

### 7.2.1 RTEMS Tools and Kernel

This procedure will build a SPARC tool chain. Set up a suitable workspace to build the release in. On Unix:

```

1 $ cd
2 $ mkdir -p development/rtems/releases
3 $ cd development/rtems/releases

```

If building on Windows:

```

1 $ cd /c
2 $ mkdir -p opt/rtems
3 $ cd opt/rtems

```

**Note** the paths on Windows will be different to those shown.

Download the RTEMS Source Builder (RSB) from the RTEMS FTP server:

```

1 $ wget https://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/rtems-source-builder-4.11.0.
  ↪tar.xz
2 --2016-03-21 10:50:04-- https://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/rtems-source-
  ↪builder-4.11.0.tar.xz
3 Resolving ftp.rtems.org (ftp.rtems.org)... 140.211.10.151
4 Connecting to ftp.rtems.org (ftp.rtems.org)|140.211.10.151|:443... connected.
5 HTTP request sent, awaiting response... 200 OK
6 Length: 967056 (944K) [application/x-xz]
7 Saving to: 'rtems-source-builder-4.11.0.tar.xz'
8
9 rtems-source-builder-4.1 100%[=====>] 944.39K 206KB/s  in_
  ↪5.5s
10
11 2016-03-21 10:50:11 (173 KB/s) - 'rtems-source-builder-4.11.0.tar.xz' saved [967056/
  ↪967056]

```

On Unix unpack the RSB release tar file using:

```

1 $ tar Jxf rtems-source-builder-4.11.0.tar.xz
2 $ cd rtems-source-builder-4.11.0/rtems/

```

On Windows you need to shorten the path (See *Chapter 6 Section 4.1 - Windows Path Length*) after you have unpacked the tar file:

```

1 $ tar Jxf rtems-source-builder-4.11.0.tar.xz
2 $ mv rtems-source-builder-4.11.0 4.110
3 $ cd 4.11.0

```

Build a tool chain for the SPARC architecture. We are using the SPARC architecture in our example because GDB has a good simulator that lets us run and test the samples RTEMS builds by default

If building on Windows add `--jobs=none` to avoid GNU make issues on Windows discussed in *Chapter 6 Section 4.2 - Parallel Builds with Make*.

```

1 $ ../source-builder/sb-set-builder \
2   --prefix=/opt/rtems/4.11 4.11/rtems-sparc
3 Build Set: 4.11/rtems-sparc
4 Build Set: 4.11/rtems-autotools.bset
5 Build Set: 4.11/rtems-autotools-internal.bset
6 config: tools/rtems-autoconf-2.69-1.cfg
7 package: autoconf-2.69-x86_64-freebsd10.1-1
8 Creating source directory: sources

```

```

9 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/autoconf-2.69.tar.gz
↳-> sources/autoconf-2.69.tar.gz
10 downloading: sources/autoconf-2.69.tar.gz - 1.8MB of 1.8MB (100%)
11 building: autoconf-2.69-x86_64-freebsd10.1-1
12 config: tools/rtems-automake-1.12.6-1.cfg
13 package: automake-1.12.6-x86_64-freebsd10.1-1
14 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/automake-1.12.6.tar.
↳gz -> sources/automake-1.12.6.tar.gz
15 downloading: sources/automake-1.12.6.tar.gz - 2.0MB of 2.0MB (100%)
16 Creating source directory: patches
17 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/automake-1.12.6-
↳bugzilla.redhat.com-1239379.diff -> patches/automake-1.12.6-bugzilla.redhat.com-1239379.
↳diff
18 downloading: patches/automake-1.12.6-bugzilla.redhat.com-1239379.diff - 408.0 bytes of
↳408.0 bytes (100%)
19 building: automake-1.12.6-x86_64-freebsd10.1-1
20 cleaning: autoconf-2.69-x86_64-freebsd10.1-1
21 cleaning: automake-1.12.6-x86_64-freebsd10.1-1
22 Build Set: Time 0:00:32.749337
23 Build Set: 4.11/rtems-autotools-base.bset
24 config: tools/rtems-autoconf-2.69-1.cfg
25 package: autoconf-2.69-x86_64-freebsd10.1-1
26 building: autoconf-2.69-x86_64-freebsd10.1-1
27 reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-freebsd10.1-1.txt
28 reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-freebsd10.1-1.xml
29 config: tools/rtems-automake-1.12.6-1.cfg
30 package: automake-1.12.6-x86_64-freebsd10.1-1
31 building: automake-1.12.6-x86_64-freebsd10.1-1
32 reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-freebsd10.1-1.txt
33 reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-freebsd10.1-1.xml
34 installing: autoconf-2.69-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
35 installing: automake-1.12.6-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
36 cleaning: autoconf-2.69-x86_64-freebsd10.1-1
37 cleaning: automake-1.12.6-x86_64-freebsd10.1-1
38 Build Set: Time 0:00:15.619219
39 Build Set: Time 0:00:48.371085
40 config: devel/expat-2.1.0-1.cfg
41 package: expat-2.1.0-x86_64-freebsd10.1-1
42 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/expat-2.1.0.tar.gz ->
↳sources/expat-2.1.0.tar.gz
43 downloading: sources/expat-2.1.0.tar.gz - 549.4kB of 549.4kB (100%)
44 building: expat-2.1.0-x86_64-freebsd10.1-1
45 reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-freebsd10.1-1.txt
46 reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-freebsd10.1-1.xml
47 config: tools/rtems-binutils-2.26-1.cfg
48 package: sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1
49 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/binutils-2.26.tar.
↳bz2 -> sources/binutils-2.26.tar.bz2
50 downloading: sources/binutils-2.26.tar.bz2 - 24.4MB of 24.4MB (100%)
51 building: sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1
52 reporting: tools/rtems-binutils-2.26-1.cfg ->
53 sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1.txt
54 reporting: tools/rtems-binutils-2.26-1.cfg ->
55 sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1.xml
56 config: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg
57 package: sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1
58 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/gcc-4.9.3.tar.bz2 ->
↳sources/gcc-4.9.3.tar.bz2

```

```

59 downloading: sources/gcc-4.9.3.tar.bz2 - 85.8MB of 85.8MB (100%)
60 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/newlib-2.2.0.
↳20150423.tar.gz -> sources/newlib-2.2.0.20150423.tar.gz
61 downloading: sources/newlib-2.2.0.20150423.tar.gz - 16.7MB of 16.7MB (100%)
62 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/mpfr-3.0.1.tar.bz2 ->
↳ sources/mpfr-3.0.1.tar.bz2
63 downloading: sources/mpfr-3.0.1.tar.bz2 - 1.1MB of 1.1MB (100%)
64 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/mpc-0.8.2.tar.gz ->
↳sources/mpc-0.8.2.tar.gz
65 downloading: sources/mpc-0.8.2.tar.gz - 535.5kB of 535.5kB (100%)
66 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/gmp-5.0.5.tar.bz2 ->
↳sources/gmp-5.0.5.tar.bz2
67 downloading: sources/gmp-5.0.5.tar.bz2 - 2.0MB of 2.0MB (100%)
68 building: sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1
69 reporting: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg ->
70 sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1.txt
71 reporting: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg ->
72 sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1.xml
73 config: tools/rtems-gdb-7.9-1.cfg
74 package: sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1
75 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/gdb-7.9.tar.xz ->
↳sources/gdb-7.9.tar.xz
76 downloading: sources/gdb-7.9.tar.xz - 17.0MB of 17.0MB (100%)
77 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0001-sim-erc32-
↳Disassembly-in-stand-alone-mode-did-not-wo.patch -> patches/0001-sim-erc32-Disassembly-
↳in-stand-alone-mode-did-not-wo.patch
78 downloading: patches/0001-sim-erc32-Disassembly-in-stand-alone-mode-did-not-wo.patch - 1.
↳9kB of 1.9kB (100%)
79 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0002-sim-erc32-
↳Corrected-wrong-CPU-implementation-and-ver.patch -> patches/0002-sim-erc32-Corrected-
↳wrong-CPU-implementation-and-ver.patch
80 downloading: patches/0002-sim-erc32-Corrected-wrong-CPU-implementation-and-ver.patch -
↳827.0 bytes of 827.0 bytes (100%)
81 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0003-sim-erc32-
↳Perform-pseudo-init-if-binary-linked-to-no.patch -> patches/0003-sim-erc32-Perform-
↳pseudo-init-if-binary-linked-to-no.patch
82 downloading: patches/0003-sim-erc32-Perform-pseudo-init-if-binary-linked-to-no.patch - 2.
↳6kB of 2.6kB (100%)
83 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0004-sim-erc32-Use-
↳fenv.h-for-host-FPU-access.patch -> patches/0004-sim-erc32-Use-fenv.h-for-host-FPU-
↳access.patch
84 downloading: patches/0004-sim-erc32-Use-fenv.h-for-host-FPU-access.patch - 4.9kB of 4.9kB
↳(100%)
85 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0005-sim-erc32-
↳Remove-unused-defines-in-Makefile-and-swit.patch -> patches/0005-sim-erc32-Remove-
↳unused-defines-in-Makefile-and-swit.patch
86 downloading: patches/0005-sim-erc32-Remove-unused-defines-in-Makefile-and-swit.patch -
↳871.0 bytes of 871.0 bytes (100%)
87 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0006-sim-erc32-Fix-
↳incorrect-simulator-performance-report.patch -> patches/0006-sim-erc32-Fix-incorrect-
↳simulator-performance-report.patch
88 downloading: patches/0006-sim-erc32-Fix-incorrect-simulator-performance-report.patch - 5.
↳6kB of 5.6kB (100%)
89 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0007-sim-erc32-File-
↳loading-via-command-line-did-not-work.patch -> patches/0007-sim-erc32-File-loading-via-
↳command-line-did-not-work.patch
90 downloading: patches/0007-sim-erc32-File-loading-via-command-line-did-not-work.patch - 1.
↳0kB of 1.0kB (100%)

```



```

91 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0008-sim-erc32-Added-
↳v-command-line-switch-for-verbose-ou.patch -> patches/0008-sim-erc32-Added-v-command-
↳line-switch-for-verbose-ou.patch
92 downloading: patches/0008-sim-erc32-Added-v-command-line-switch-for-verbose-ou.patch - 3.
↳6kB of 3.6kB (100%)
93 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0009-sim-erc32-
↳Removed-type-mismatch-compiler-warnings.patch -> patches/0009-sim-erc32-Removed-type-
↳mismatch-compiler-warnings.patch
94 downloading: patches/0009-sim-erc32-Removed-type-mismatch-compiler-warnings.patch - 1.9kB
↳of 1.9kB (100%)
95 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0010-sim-erc32-
↳Switched-emulated-memory-to-host-endian-or.patch -> patches/0010-sim-erc32-Switched-
↳emulated-memory-to-host-endian-or.patch
96 downloading: patches/0010-sim-erc32-Switched-emulated-memory-to-host-endian-or.patch - 16.
↳0kB of 16.0kB (100%)
97 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0011-sim-erc32-
↳use-SIM_AC_OPTION_HOSTENDIAN-to-probe-for-.patch -> patches/0011-sim-erc32-use-SIM_
↳AC_OPTION_HOSTENDIAN-to-probe-for-.patch
98 downloading: patches/0011-sim-erc32-use-SIM_AC_OPTION_HOSTENDIAN-to-probe-for-.patch - 14.
↳8kB of 14.8kB (100%)
99 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0012-sim-erc32-Use-
↳memory_iread-function-for-instruction-.patch -> patches/0012-sim-erc32-Use-memory_iread-
↳function-for-instruction-.patch
100 downloading: patches/0012-sim-erc32-Use-memory_iread-function-for-instruction-.patch - 3.
↳8kB of 3.8kB (100%)
101 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0013-sim-erc32-Fix-a-
↳few-compiler-warnings.patch-> patches/0013-sim-erc32-Fix-a-few-compiler-warnings.patch
102 downloading: patches/0013-sim-erc32-Fix-a-few-compiler-warnings.patch - 2.2kB of 2.2kB
↳(100%)
103 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0014-sim-erc32-Use-
↳gdb-callback-for-UART-I-0-when-linked-.patch -> patches/0014-sim-erc32-Use-gdb-callback-
↳for-UART-I-0-when-linked-.patch
104 downloading: patches/0014-sim-erc32-Use-gdb-callback-for-UART-I-0-when-linked-.patch - 9.
↳2kB of 9.2kB (100%)
105 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0015-sim-erc32-
↳Access-memory-subsystem-through-struct-mem.patch -> patches/0015-sim-erc32-Access-
↳memory-subsystem-through-struct-mem.patch
106 downloading: patches/0015-sim-erc32-Access-memory-subsystem-through-struct-mem.patch - 22.
↳9kB of 22.9kB (100%)
107 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0016-sim-erc32-Use-
↳readline.h-for-readline-types-and-func.patch -> patches/0016-sim-erc32-Use-readline.h-
↳for-readline-types-and-func.patch
108 downloading: patches/0016-sim-erc32-Use-readline.h-for-readline-types-and-func.patch - 1.
↳5kB of 1.5kB (100%)
109 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0017-sim-erc32-Move-
↳local-extern-declarations-into-sis.h.patch -> patches/0017-sim-erc32-Move-local-extern-
↳declarations-into-sis.h.patch
110 downloading: patches/0017-sim-erc32-Move-local-extern-declarations-into-sis.h.patch - 5.
↳8kB of 5.8kB (100%)
111 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0018-sim-erc32-Add-
↳support-for-LEON3-processor-emulation.patch -> patches/0018-sim-erc32-Add-support-for-
↳LEON3-processor-emulation.patch
112 downloading: patches/0018-sim-erc32-Add-support-for-LEON3-processor-emulation.patch - 66.
↳7kB of 66.7kB (100%)
113 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0019-sim-erc32-Add-
↳support-for-LEON2-processor-emulation.patch -> patches/0019-sim-erc32-Add-support-for-
↳LEON2-processor-emulation.patch

```

```

114 downloading: patches/0019-sim-erc32-Add-support-for-LEON2-processor-emulation.patch - 26.
    ↳ 1kB of 26.1kB (100%)
115 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0020-sim-erc32-
    ↳ Updated-documentation.patch -> patches/0020-sim-erc32-Updated-documentation.patch
116 downloading: patches/0020-sim-erc32-Updated-documentation.patch - 16.1kB of 16.1kB (100%)
117 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0021-sim-erc32-Add-
    ↳ data-watchpoint-support.patch -> patches/0021-sim-erc32-Add-data-watchpoint-support.
    ↳ patch
118 downloading: patches/0021-sim-erc32-Add-data-watchpoint-support.patch - 10.1kB of 10.1kB_
    ↳ (100%)
119 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0022-Add-watchpoint-
    ↳ support-to-gdb-simulator-interface.patch -> patches/0022-Add-watchpoint-support-to-gdb-
    ↳ simulator-interface.patch
120 downloading: patches/0022-Add-watchpoint-support-to-gdb-simulator-interface.patch - 25.
    ↳ 5kB of 25.5kB (100%)
121 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0023-sim-erc32-ELF-
    ↳ loading-could-fail-on-unaligned-sectio.patch -> patches/0023-sim-erc32-ELF-loading-
    ↳ could-fail-on-unaligned-sectio.patch
122 downloading: patches/0023-sim-erc32-ELF-loading-could-fail-on-unaligned-sectio.patch - 1.
    ↳ 3kB of 1.3kB (100%)
123 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/gdb-sim-arange-
    ↳ inline.diff -> patches/gdb-sim-arange-inline.diff
124 downloading: patches/gdb-sim-arange-inline.diff - 761.0 bytes of 761.0 bytes (100%)
125 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/gdb-sim-cgen-inline.
    ↳ diff -> patches/gdb-sim-cgen-inline.diff
126 downloading: patches/gdb-sim-cgen-inline.diff - 706.0 bytes of 706.0 bytes (100%)
127 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/patch-gdb-python-
    ↳ python-config.py -> patches/patch-gdb-python-python-config.py
128 downloading: patches/patch-gdb-python-python-config.py - 449.0 bytes of 449.0 bytes (100%)
129 building: sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1
130 reporting: tools/rtems-gdb-7.9-1.cfg ->
131 sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1.txt
132 reporting: tools/rtems-gdb-7.9-1.cfg ->
133 sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1.xml
134 config: tools/rtems-tools-4.11-1.cfg
135 package: rtems-tools-4.11.0-1
136 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/rtems-tools-4.11.0.tar.xz ->_
    ↳ sources/rtems-tools-4.11.0.tar.xz
137 downloading: sources/rtems-tools-4.11.0.tar.xz - 1.6MB of 1.6MB (100%)
138 building: rtems-tools-4.11.0-1
139 reporting: tools/rtems-tools-4.11-1.cfg -> rtems-tools-4.11.0-1.txt
140 reporting: tools/rtems-tools-4.11-1.cfg -> rtems-tools-4.11.0-1.xml
141 config: tools/rtems-kernel-4.11.cfg
142 package: sparc-rtems4.11-kernel-4.11.0-1
143 download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/rtems-4.11.0.tar.xz ->_
    ↳ sources/rtems-4.11.0.tar.xz
144 downloading: sources/rtems-4.11.0.tar.xz - 9.3MB of 9.3MB (100%)
145 building: sparc-rtems4.11-kernel-4.11.0-1
146 reporting: tools/rtems-kernel-4.11.cfg -> sparc-rtems4.11-kernel-4.11.0-1.txt
147 reporting: tools/rtems-kernel-4.11.cfg -> sparc-rtems4.11-kernel-4.11.0-1.xml
148 installing: expat-2.1.0-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
149 installing: sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
150 installing: sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1 -> /opt/
    ↳ work/rtems/4.11.0
151 installing: sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
152 installing: rtems-tools-4.11.0-1 -> /opt/work/rtems/4.11.0
153 installing: sparc-rtems4.11-kernel-4.11.0-1 -> /opt/work/rtems/4.11.0

```

```
154 cleaning: expat-2.1.0-x86_64-freebsd10.1-1
155 cleaning: sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1
156 cleaning: sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1
157 cleaning: sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1
158 cleaning: rtems-tools-4.11.0-1
159 cleaning: sparc-rtems4.11-kernel-4.11.0-1
160 Build Set: Time 0:19:15.713662
```

You can now build a 3rd party library or an application as defaulted in TBD.

## 7.3 Developer (Unstable)

RTEMS provides open access to its development processes. We call this the developer set up. The project encourages all users to inspect, review, comment and contribute to the code base. The processes described here are the same processes the core development team use when developing and maintaining RTEMS.

**Warning:** The development version is not for use in production and it can break from time to time.

Please read *Chapter 6 - Host Computer* before continuing. The following procedure assumes you have installed and configured your host operating system. It also assumes you have installed any dependent packages needed when building the tools and the kernel.

You need to select a location to build and install the RTEMS Tool chain and RTEMS. Make sure there is plenty of disk space and a fast disk is recommended. Our procedure will document building and installing the tools in a home directory called `development/rtems`. Using a home directory means you can do this without needing to be root. You can also use `/opt/rtems/build` if you have access to that path.

The location used to install the tools and kernel is called the *prefix*. It is best to have a *prefix* for each different version of RTEMS you are using. If you are using RTEMS 4.11 in production it is not a good idea to install a development version of 5 over the top. A separate *prefix* for each version avoids this.

The RTEMS tool chain changes less often than the RTEMS kernel. One method of working with development releases is to have a separate *prefix* for the RTEMS tools and a different one for the RTEMS kernel. You can then update each without interacting with the other. You can also have a number of RTEMS versions available to test with.

### *Downloading the source*

You need an internet connection to download the source. The downloaded source is cached locally and the RSB checksums it. If you run a build again the download output will be missing. Using the RSB from git will download the source from the upstream project's home site and this could be *http*, *ftp*, or *git*.

### 7.3.1 POSIX and OS X Host Tools Chain

This procedure will build a SPARC tool chain.

Clone the RTEMS Source Builder (RSB) repository:

```
1 $ cd
2 $ mkdir -p development/rtems
3 $ cd development/rtems
4 $ git clone git://git.rtems.org/rtems-source-builder.git rsb
5 Cloning into 'rsb'...
6 remote: Counting objects: 5837, done.
7 remote: Compressing objects: 100% (2304/2304), done.
8 remote: Total 5837 (delta 4014), reused 5056 (delta 3494)
```

```

9 Receiving objects: 100% (5837/5837), 2.48 MiB | 292.00 KiB/s, done.
10 Resolving deltas: 100% (4014/4014), done.
11 Checking connectivity... done.

```

Check all the host packages you need are present. Current libraries are not checked and this includes checking for the python development libraries GDB requires:

```

1 $ cd rsb
2 $ ./source-builder/sb-check
3 RTEMS Source Builder - Check, 5 (089327b5dcf9)
4 Environment is ok

```

Build a tool chain for the SPARC architecture. We are using the SPARC architecture because GDB has a good simulator that lets us run and test the samples RTEMS builds by default. The current development version is 5 and is on master:

```

1 $ cd rtems
2 $ ../source-builder/sb-set-builder --prefix=/usr/home/chris/development/rtems/5 5/rtems-
  ↳ sparc
3 RTEMS Source Builder - Set Builder, 5 (089327b5dcf9)
4 Build Set: 5/rtems-sparc
5 Build Set: 5/rtems-autotools.bset
6 Build Set: 5/rtems-autotools-internal.bset
7 config: tools/rtems-autoconf-2.69-1.cfg
8 package: autoconf-2.69-x86_64-linux-gnu-1
9 Creating source directory: sources
10 download: ftp://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz -> sources/autoconf-2.69.
  ↳ tar.gz
11 downloading: sources/autoconf-2.69.tar.gz - 1.8MB of 1.8MB (100%)
12 building: autoconf-2.69-x86_64-linux-gnu-1
13 config: tools/rtems-automake-1.12.6-1.cfg
14 package: automake-1.12.6-x86_64-linux-gnu-1
15 download: ftp://ftp.gnu.org/gnu/automake/automake-1.12.6.tar.gz -> sources/automake-1.12.
  ↳ 6.tar.gz
16 downloading: sources/automake-1.12.6.tar.gz - 2.0MB of 2.0MB (100%)
17 Creating source directory: patches
18 download: https://git.rtems.org/rtems-tools/plain/tools/5/automake/automake-1.12.6-
  ↳ bugzilla.redhat.com-1239379.diff -> patches/automake-1.12.6-bugzilla.redhat.com-1239379.
  ↳ diff
19 downloading: patches/automake-1.12.6-bugzilla.redhat.com-1239379.diff - 408.0 bytes of
  ↳ 408.0 bytes (100%)
20 building: automake-1.12.6-x86_64-linux-gnu-1
21 cleaning: autoconf-2.69-x86_64-linux-gnu-1
22 cleaning: automake-1.12.6-x86_64-linux-gnu-1
23 Build Set: Time 0:00:12.713221
24 Build Set: 5/rtems-autotools-base.bset
25 config: tools/rtems-autoconf-2.69-1.cfg
26 package: autoconf-2.69-x86_64-linux-gnu-1
27 building: autoconf-2.69-x86_64-linux-gnu-1
28 reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-linux-gnu-1.txt
29 reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-linux-gnu-1.xml
30 config: tools/rtems-automake-1.12.6-1.cfg
31 package: automake-1.12.6-x86_64-linux-gnu-1
32 building: automake-1.12.6-x86_64-linux-gnu-1
33 reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-linux-gnu-1.txt
34 reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-linux-gnu-1.xml
35 installing: autoconf-2.69-x86_64-linux-gnu-1 -> /usr/home/chris/development/rtems/5

```

```

36 installing: automake-1.12.6-x86_64-linux-gnu-1 -> /usr/home/chris/development/rtems/5
37 cleaning: autoconf-2.69-x86_64-linux-gnu-1
38 cleaning: automake-1.12.6-x86_64-linux-gnu-1
39 Build Set: Time 0:00:09.105363
40 Build Set: Time 0:00:21.822083
41 config: devel/expat-2.1.0-1.cfg
42 package: expat-2.1.0-x86_64-linux-gnu-1
43 download: http://downloads.sourceforge.net/project/expat/expat/2.1.0/expat-2.1.0.tar.gz ->
  ↪ sources/expat-2.1.0.tar.gz
44 redirect: https://vorboss.dl.sourceforge.net/project/expat/expat/2.1.0/expat-2.1.0.tar.gz
45 downloading: sources/expat-2.1.0.tar.gz - 549.4kB of 549.4kB (100%)
46 building: expat-2.1.0-x86_64-linux-gnu-1
47 reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-linux-gnu-1.txt
48 reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-linux-gnu-1.xml
49 config: tools/rtems-binutils-2.29-1.cfg
50 package: sparc-rtems5-binutils-2.29-x86_64-linux-gnu-1
51 download: ftp://ftp.gnu.org/gnu/binutils/binutils-2.29.tar.bz2 -> sources/binutils-2.29.
  ↪tar.bz2
52 downloading: sources/binutils-2.29.tar.bz2 - 27.7MB of 27.7MB (100%)
53 download: https://devel.rtems.org/raw-attachment/ticket/3091/0001-Fix-Binutils-2.29-
  ↪PR21884.patch -> patches/0001-Fix-Binutils-2.29-PR21884.patch
54 downloading: patches/0001-Fix-Binutils-2.29-PR21884.patch - 8.8kB of 8.8kB (100%)
55 building: sparc-rtems5-binutils-2.29-x86_64-linux-gnu-1
56 reporting: tools/rtems-binutils-2.29-1.cfg -> sparc-rtems5-binutils-2.29-x86_64-linux-gnu-
  ↪1.txt
57 reporting: tools/rtems-binutils-2.29-1.cfg -> sparc-rtems5-binutils-2.29-x86_64-linux-gnu-
  ↪1.xml
58 config: tools/rtems-gcc-7.2.0-newlib-2.5.0.20170922-1.cfg
59 package: sparc-rtems5-gcc-7.2.0-newlib-2.5.0.20170922-x86_64-linux-gnu-1
60 download: https://ftp.gnu.org/gnu/gcc/gcc-7.2.0/gcc-7.2.0.tar.xz -> sources/gcc-7.2.0.tar.
  ↪xz
61 downloading: sources/gcc-7.2.0.tar.xz - 59.4MB of 59.4MB (100%)
62 download: https://gcc.gnu.org/git/?p=gcc.git;a=commitdiff_plain;h=62ffbc7502f0ff88ff7566cd6d7c59c0483ecc0
  ↪-> patches/gcc-62ffbc7502f0ff88ff7566cd6d7c59c0483ecc0.patch
63 downloading: patches/gcc-62ffbc7502f0ff88ff7566cd6d7c59c0483ecc0.patch - 1.8kB
64 download: https://gcc.gnu.org/git/?p=gcc.git;a=blobdiff_plain;f=gcc/config/gcc;
  ↪h=593631849bb5e0df5cc4ff42c1a1cc34b7eec2f8;hp=a9196cd26d9ec24c2e3f6026f63348cae3734861;
  ↪hb=e840389000b8339a63bee56d8b3...<see log> -> patches/gcc-593631849bb5e0df5cc4ff42c1a1cc34b7eec2f8.
  ↪patch
65 downloading: patches/gcc-593631849bb5e0df5cc4ff42c1a1cc34b7eec2f8.patch - 806.0 bytes
66 download: https://gcc.gnu.org/git/?p=gcc.git;a=blobdiff_plain;f=gcc/config/rs6000/rtems.h;
  ↪h=7ea9ebdb77b6a9b7060ad2362318e0e12b9058ae;hp=8a62fdcbaf321d616021c4c396619b7f56cf5ed2;
  ↪hb=e840389000b8339a...<see log> -> patches/gcc-7ea9ebdb77b6a9b7060ad2362318e0e12b9058ae.
  ↪patch
67 downloading: patches/gcc-7ea9ebdb77b6a9b7060ad2362318e0e12b9058ae.patch - 3.2kB
68 download: ftp://sourceware.org/pub/newlib/newlib-2.5.0.20170922.tar.gz -> sources/newlib-
  ↪2.5.0.20170922.tar.gz
69 downloading: sources/newlib-2.5.0.20170922.tar.gz - 17.3MB of 17.3MB (100%)
70 download: https://devel.rtems.org/raw-attachment/ticket/2514/0001-RTEMS-Self-contained-
  ↪POSIX-objects.patch -> patches/0001-RTEMS-Self-contained-POSIX-objects.patch
71 downloading: patches/0001-RTEMS-Self-contained-POSIX-objects.patch - 5.7kB of 5.7kB (100%)
72 download: https://sourceware.org/git/gitweb.cgi?p=newlib-cygwin.git;a=patch;
  ↪h=c165a27c0147471977377acd8918ab3b446f947a -> patches/newlib-cygwin-git-
  ↪c165a27c0147471977377acd8918ab3b446f947a.patch
73 downloading: patches/newlib-cygwin-git-c165a27c0147471977377acd8918ab3b446f947a.patch -
  ↪986.0 bytes
74 download: https://sourceware.org/git/gitweb.cgi?p=newlib-cygwin.git;a=patch;
  ↪h=ce189d8afef720b0977b5cae7f9eabf5d49b530c -> patches/newlib-cygwin-git-
  ↪ce189d8afef720b0977b5cae7f9eabf5d49b530c.patch

```



```

75 downloading: patches/newlib-cygwin-git-ce189d8afef720b0977b5cae7f9eabf5d49b530c.patch - 3.
   ↪ 4kB
76 download: https://ftp.gnu.org/gnu/mpfr/mpfr-3.1.4.tar.bz2 -> sources/mpfr-3.1.4.tar.bz2
77 downloading: sources/mpfr-3.1.4.tar.bz2 - 1.2MB of 1.2MB (100%)
78 download: https://ftp.gnu.org/gnu/mpc/mpc-1.0.3.tar.gz -> sources/mpc-1.0.3.tar.gz
79 downloading: sources/mpc-1.0.3.tar.gz - 654.2kB of 654.2kB (100%)
80 download: https://ftp.gnu.org/gnu/gmp/gmp-6.1.0.tar.bz2 -> sources/gmp-6.1.0.tar.bz2
81 downloading: sources/gmp-6.1.0.tar.bz2 - 2.3MB of 2.3MB (100%)
82 building: sparc-rtems5-gcc-7.2.0-newlib-2.5.0.20170922-x86_64-linux-gnu-1
83 reporting: tools/rtems-gcc-7.2.0-newlib-2.5.0.20170922-1.cfg -> sparc-rtems5-gcc-7.2.0-
   ↪ newlib-2.5.0.20170922-x86_64-linux-gnu-1.txt
84 reporting: tools/rtems-gcc-7.2.0-newlib-2.5.0.20170922-1.cfg -> sparc-rtems5-gcc-7.2.0-
   ↪ newlib-2.5.0.20170922-x86_64-linux-gnu-1.xml
85 config: tools/rtems-gdb-8.0.1-1.cfg
86 package: sparc-rtems5-gdb-8.0.1-x86_64-linux-gnu-1
87 download: http://ftp.gnu.org/gnu/gdb/gdb-8.0.1.tar.xz -> sources/gdb-8.0.1.tar.xz
88 downloading: sources/gdb-8.0.1.tar.xz - 18.7MB of 18.7MB (100%)
89 download: https://gaisler.org/gdb/gdb-8.0.1-sis-leon2-leon3.diff -> patches/gdb-8.0.1-sis-
   ↪ leon2-leon3.diff
90 downloading: patches/gdb-8.0.1-sis-leon2-leon3.diff - 224.5kB of 224.5kB (100%)
91 building: sparc-rtems5-gdb-8.0.1-x86_64-linux-gnu-1
92 reporting: tools/rtems-gdb-8.0.1-1.cfg -> sparc-rtems5-gdb-8.0.1-x86_64-linux-gnu-1.txt
93 reporting: tools/rtems-gdb-8.0.1-1.cfg -> sparc-rtems5-gdb-8.0.1-x86_64-linux-gnu-1.xml
94 config: tools/rtems-tools-5-1.cfg
95 package: rtems-tools-HEAD-1
96 Creating source directory: sources/git
97 git: clone: git://git.rtems.org/rtems-tools.git -> sources/git/rtems-tools.git
98 git: reset: git://git.rtems.org/rtems-tools.git
99 git: fetch: git://git.rtems.org/rtems-tools.git -> sources/git/rtems-tools.git
100 git: checkout: git://git.rtems.org/rtems-tools.git => HEAD
101 git: pull: git://git.rtems.org/rtems-tools.git
102 building: rtems-tools-HEAD-1
103 reporting: tools/rtems-tools-5-1.cfg -> rtems-tools-HEAD-1.txt
104 reporting: tools/rtems-tools-5-1.cfg -> rtems-tools-HEAD-1.xml
105 config: tools/rtems-kernel-5.cfg
106 package: sparc-rtems5-kernel-5-1
107 building: sparc-rtems5-kernel-5-1
108 reporting: tools/rtems-kernel-5.cfg -> sparc-rtems5-kernel-5-1.txt
109 reporting: tools/rtems-kernel-5.cfg -> sparc-rtems5-kernel-5-1.xml
110 installing: expat-2.1.0-x86_64-linux-gnu-1 -> /usr/home/chris/development/rtems/5
111 installing: sparc-rtems5-binutils-2.29-x86_64-linux-gnu-1 -> /usr/home/chris/development/
   ↪ rtems/5
112 installing: sparc-rtems5-gcc-7.2.0-newlib-2.5.0.20170922-x86_64-linux-gnu-1 -> /usr/home/
   ↪ chris/development/rtems/5
113 installing: sparc-rtems5-gdb-8.0.1-x86_64-linux-gnu-1 -> /usr/home/chris/development/
   ↪ rtems/5
114 installing: rtems-tools-HEAD-1 -> /usr/home/chris/development/rtems/5
115 installing: sparc-rtems5-kernel-5-1 -> /usr/home/chris/development/rtems/5
116 cleaning: expat-2.1.0-x86_64-linux-gnu-1
117 cleaning: sparc-rtems5-binutils-2.29-x86_64-linux-gnu-1
118 cleaning: sparc-rtems5-gcc-7.2.0-newlib-2.5.0.20170922-x86_64-linux-gnu-1
119 cleaning: sparc-rtems5-gdb-8.0.1-x86_64-linux-gnu-1
120 cleaning: rtems-tools-HEAD-1
121 cleaning: sparc-rtems5-kernel-5-1
122 Build Set: Time 0:39:33.988995

```

### 7.3.2 Windows Host Tool Chain

This section details how you create an RTEMS development environment on Windows. The installation documented here is on *Windows 7 64bit Professional*. Building on *Windows 10* has been reported as working.

Please see *Chapter 6 Section 4 - Microsoft Windows* before continuing.

---

**Note:** If the RSB reports error: no hosts defaults found; please add you have probably opened an MSYS2 32bit Shell. Close all 32bit Shell windows and open the MSYS2 64bit Shell.

---

#### 7.3.2.1 RTEMS Windows Tools

Create a workspace for RTEMS using the following shell command:

##### *Creating Tool Archives*

Add `--bset-tar-file` to the `sb-set-builder` command line to create tar files of the built package set.

```
1 ~
2 $ mkdir -p /c/opt/rtems
```

The `/c` path is an internal MSYS2 mount point of the C: drive. The command creates the RTEMS work space on the C: drive. If you wish to use another drive please substitute `/c` with your drive letter.

We build and install all RTEMS packages under the *prefix* we just created. Change to that directory and get a copy of the RSB:

```
1 ~
2 $ cd /c/opt/rtems
3 /c/opt/rtems
4 $ git clone git://git.rtems.org/rtems-source-builder.git rsb
5 Cloning into 'rsb'...
6 remote: Counting objects: 5716, done.
7 remote: Compressing objects: 100% (2183/2183), done.
8 remote: Total 5716 (delta 3919), reused 5071 (delta 3494)
9 Receiving objects: 100% (5716/5716), 2.46 MiB | 656.00 KiB/s, done.
10 Resolving deltas: 100% (3919/3919), done.
11 Checking connectivity... done.
12 Checking out files: 100% (630/630), done.
13 /c/opt/rtems
14 $ cd rsb
```

We are building RTEMS 4.11 tools so select the *4.11* branch:

```
1 /c/opt/rtems/rsb
2 $ git checkout 4.11
3 Branch 4.11 set up to track remote branch 4.11 from origin.
4 Switched to a new branch '4.11'
```



```
5 /c/opt/rtems/rsb
6 $
```

Check the RSB has a valid environment:

```
1 /c/opt/rtems/rsb
2 $ cd rtems
3 /c/opt/rtems/rsb/rtems
4 $ ../source-builder/sb-check
5 RTEMS Source Builder - Check, 4.11 (01ac76f2f90f)
6 Environment is ok
7 /c/opt/rtems/rsb/rtems
8 $
```

To build a set of RTEMS tools for the Intel i386 architecture. The build runs a single job rather than a job per CPU in your machine and will take a long time so please be patient. The RSB creates a log file containing all the build output and it will be changing size. The RSB command to build i386 tools is:

```
1 /c/opt/rtems/rsb/rtems
2 $ ../source-builder/sb-set-builder --prefix=/c/opt/rtems/4.11 \
3                                     --jobs=none 4.11/rtems-i386
4 RTEMS Source Builder - Set Builder, 4.11 (01ac76f2f90f)
5 Build Set: 4.11/rtems-i386
6 Build Set: 4.11/rtems-autotools.bset
7 Build Set: 4.11/rtems-autotools-internal.bset
8 config: tools/rtems-autoconf-2.69-1.cfg
9 package: autoconf-2.69-x86_64-w64-mingw32-1
10 Creating source directory: sources
11 download: ftp://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz -> sources/autoconf-2.69.
12 ↪tar.gz
13 downloading: sources/autoconf-2.69.tar.gz - 1.8MB of 1.8MB (100%)
14 building: autoconf-2.69-x86_64-w64-mingw32-1
15 config: tools/rtems-automake-1.12.6-1.cfg
16 package: automake-1.12.6-x86_64-w64-mingw32-1
17 download: ftp://ftp.gnu.org/gnu/automake/automake-1.12.6.tar.gz -> sources/automake-1.12.
18 ↪6.tar.gz
19 downloading: sources/automake-1.12.6.tar.gz - 2.0MB of 2.0MB (100%)
20 building: automake-1.12.6-x86_64-w64-mingw32-1
21 cleaning: autoconf-2.69-x86_64-w64-mingw32-1
22 cleaning: automake-1.12.6-x86_64-w64-mingw32-1
23 Build Set: Time 0:00:42.515625
24 Build Set: 4.11/rtems-autotools-base.bset
25 config: tools/rtems-autoconf-2.69-1.cfg
26 package: autoconf-2.69-x86_64-w64-mingw32-1
27 building: autoconf-2.69-x86_64-w64-mingw32-1
28 reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-w64-mingw32-1.txt
29 reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-w64-mingw32-1.xml
30 config: tools/rtems-automake-1.12.6-1.cfg
31 package: automake-1.12.6-x86_64-w64-mingw32-1
32 building: automake-1.12.6-x86_64-w64-mingw32-1
33 reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-w64-mingw32-1.txt
34 reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-w64-mingw32-1.xml
35 tarball: tar/rtems-4.11-autotools-x86_64-w64-mingw32-1.tar.bz2
36 installing: autoconf-2.69-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
37 installing: automake-1.12.6-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
38 cleaning: autoconf-2.69-x86_64-w64-mingw32-1
```

```

37 cleaning: automake-1.12.6-x86_64-w64-mingw32-1
38 Build Set: Time 0:00:37.718750
39 Build Set: Time 0:01:20.234375
40 config: devel/expat-2.1.0-1.cfg
41 package: expat-2.1.0-x86_64-w64-mingw32-1
42 download: http://downloads.sourceforge.net/project/expat/expat/2.1.0/expat-2.1.0.tar.gz ->
↳ sources/expat-2.1.0.tar.gz
43 redirect: http://iweb.dl.sourceforge.net/project/expat/expat/2.1.0/expat-2.1.0.tar.gz
44 downloading: sources/expat-2.1.0.tar.gz - 549.4kB of 549.4kB (100%)
45 building: expat-2.1.0-x86_64-w64-mingw32-1
46 reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-w64-mingw32-1.txt
47 reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-w64-mingw32-1.xml
48 config: tools/rtems-binutils-2.24-1.cfg
49 package: i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1
50 download: ftp://ftp.gnu.org/gnu/binutils/binutils-2.24.tar.bz2 -> sources/binutils-2.24.
↳ tar.bz2
51 downloading: sources/binutils-2.24.tar.bz2 - 21.7MB of 21.7MB (100%)
52 building: i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1
53 reporting: tools/rtems-binutils-2.24-1.cfg -> i386-rtems4.11-binutils-2.24-x86_64-w64-
↳ mingw32-1.txt
54 reporting: tools/rtems-binutils-2.24-1.cfg -> i386-rtems4.11-binutils-2.24-x86_64-w64-
↳ mingw32-1.xml
55 config: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg
56 package: i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1
57 download: ftp://ftp.gnu.org/gnu/gcc/gcc-4.9.3/gcc-4.9.3.tar.bz2 -> sources/gcc-4.9.3.tar.
↳ bz2
58 downloading: sources/gcc-4.9.3.tar.bz2 - 85.8MB of 85.8MB (100%)
59 download: ftp://sourceware.org/pub/newlib/newlib-2.2.0.20150423.tar.gz -> sources/newlib-
↳ 2.2.0.20150423.tar.gz
60 downloading: sources/newlib-2.2.0.20150423.tar.gz - 16.7MB of 16.7MB (100%)
61 download: http://www.mpfr.org/mpfr-3.0.1/mpfr-3.0.1.tar.bz2 -> sources/mpfr-3.0.1.tar.bz2
62 downloading: sources/mpfr-3.0.1.tar.bz2 - 1.1MB of 1.1MB (100%)
63 download: http://www.multiprecision.org/mpc/download/mpc-0.8.2.tar.gz -> sources/mpc-0.8.
↳ 2.tar.gz
64 downloading: sources/mpc-0.8.2.tar.gz - 535.5kB of 535.5kB (100%)
65 download: ftp://ftp.gnu.org/gnu/gmp/gmp-5.0.5.tar.bz2 -> sources/gmp-5.0.5.tar.bz2
66 downloading: sources/gmp-5.0.5.tar.bz2 - 2.0MB of 2.0MB (100%)
67 building: i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1
68 reporting: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg ->
i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1.txt
69 reporting: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg ->
i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1.xml
70 reporting: tools/rtems-gdb-7.9-1.cfg ->
71 i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1
72 config: tools/rtems-gdb-7.9-1.cfg
73 package: i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1
74 download: http://ftp.gnu.org/gnu/gdb/gdb-7.9.tar.xz -> sources/gdb-7.9.tar.xz
75 downloading: sources/gdb-7.9.tar.xz - 17.0MB of 17.0MB (100%)
76 download: https://git.rtems.org/rtems-tools/plain/tools/4.11/gdb/gdb-sim-arange-inline.
↳ diff -> patches/gdb-sim-arange-inline.diff
77 downloading: patches/gdb-sim-arange-inline.diff - 761.0 bytes of 761.0 bytes (100%)
78 download: https://git.rtems.org/rtems-tools/plain/tools/4.11/gdb/gdb-sim-cgen-inline.diff_
↳ -> patches/gdb-sim-cgen-inline.diff
79 downloading: patches/gdb-sim-cgen-inline.diff - 706.0 bytes of 706.0 bytes (100%)
80 building: i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1
81 reporting: tools/rtems-gdb-7.9-1.cfg ->
82 i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1.txt
83 reporting: tools/rtems-gdb-7.9-1.cfg ->
84 i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1.xml

```

```

85 config: tools/rtems-tools-4.11-1.cfg
86 package: rtems-tools-4.11-1
87 Creating source directory: sources/git
88 git: clone: git://git.rtems.org/rtems-tools.git -> sources/git/rtems-tools.git
89 git: reset: git://git.rtems.org/rtems-tools.git
90 git: fetch: git://git.rtems.org/rtems-tools.git -> sources/git/rtems-tools.git
91 git: checkout: git://git.rtems.org/rtems-tools.git => 4.11
92 git: pull: git://git.rtems.org/rtems-tools.git
93 building: rtems-tools-4.11-1
94 reporting: tools/rtems-tools-4.11-1.cfg -> rtems-tools-4.11-1.txt
95 reporting: tools/rtems-tools-4.11-1.cfg -> rtems-tools-4.11-1.xml
96 config: tools/rtems-kernel-4.11.cfg
97 installing: expat-2.1.0-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
98 installing: i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
99 installing: i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1 ->
  ↪C:\opt\rtems\4.11
100 installing: i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
101 installing: rtems-tools-4.11-1 -> C:\opt\rtems\4.11
102 cleaning: expat-2.1.0-x86_64-w64-mingw32-1
103 cleaning: i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1
104 cleaning: i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1
105 cleaning: i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1
106 cleaning: rtems-tools-4.11-1
107 Build Set: Time 1:32:58.972919
108 /c/opt/rtems/rsb/rtems
109 $

```

### 7.3.2.2 Building the Kernel

We can now build the RTEMS kernel using the RTEMS tools we have just built. First we need to set the path to the tools:

```

1 /c
2 $ cd /c/opt/rtems
3 /c/opt/rtems
4 $ export PATH=/c/opt/rtems/4.11/bin:$PATH
5 /c/opt/rtems
6 $

```

We currently build RTEMS from the git release branch for 4.11:

```

1 /c/opt/rtems
2 $ mkdir kernel
3 /c/opt/rtems
4 $ cd kernel
5 /c/opt/rtems/kernel
6 $ git clone git://git.rtems.org/rtems.git rtems
7 Cloning into 'rtems'...
8 remote: Counting objects: 482766, done.
9 remote: Compressing objects: 100% (88781/88781), done.
10 remote: Total 482766 (delta 389610), reused 475155 (delta 383437)
11 Receiving objects: 100% (482766/482766), 69.77 MiB | 697.00 KiB/s, done.
12 Resolving deltas: 100% (389610/389610), done.
13 Checking connectivity... done.
14 Checking out files: 100% (10626/10626), done.

```

```

15 /c/opt/rtems/kernel
16 $ cd rtems
17 /c/opt/rtems/kernel/rtems
18 $ git checkout 4.11
19 Checking out files: 100% (2553/2553), done.
20 Branch 4.11 set up to track remote branch 4.11 from origin.
21 Switched to a new branch '4.11'
22 /c/opt/rtems/kernel
23 $

```

The kernel code cloned from git needs to be *bootstrapped*. Bootstrapping creates autoconf and automake generated files. To bootstrap we first clean away any files, then generate the pre-install header file lists and finally we generate the autoconf and automake files using the RSB's bootstrap tool. First we clean any generated files that exist:

```

1 /c/opt/rtems/kernel/rtems
2 $ ./bootstrap -c
3 removing automake generated Makefile.in files
4 removing configure files
5 removing aclocal.m4 files

```

Then we generate the pre-install header file automake make files:

```

1 /c/opt/rtems/kernel/rtems
2 $ ./bootstrap -p
3 Generating ./src/ada/preinstall.am
4 Generating ./src/lib/libbsp/arm/altera-cyclone-v/preinstall.am
5 Generating ./src/lib/libbsp/arm/atsam/preinstall.am
6 Generating ./src/lib/libbsp/arm/beagle/preinstall.am
7 Generating ./src/lib/libbsp/arm/csb336/preinstall.am
8 Generating ./src/lib/libbsp/arm/csb337/preinstall.am
9 Generating ./src/lib/libbsp/arm/edb7312/preinstall.am
10 Generating ./src/lib/libbsp/arm/gdbarmsim/preinstall.am
11 .....
12 Generating ./cpukit/score/cpu/mips/preinstall.am
13 Generating ./cpukit/score/cpu/moxie/preinstall.am
14 Generating ./cpukit/score/cpu/nios2/preinstall.am
15 Generating ./cpukit/score/cpu/no_cpu/preinstall.am
16 Generating ./cpukit/score/cpu/or1k/preinstall.am
17 Generating ./cpukit/score/cpu/powerpc/preinstall.am
18 Generating ./cpukit/score/cpu/sh/preinstall.am
19 Generating ./cpukit/score/cpu/sparc/preinstall.am
20 Generating ./cpukit/score/cpu/sparc64/preinstall.am
21 Generating ./cpukit/score/cpu/v850/preinstall.am
22 Generating ./cpukit/score/preinstall.am
23 Generating ./cpukit/telnetd/preinstall.am
24 Generating ./cpukit/wrapup/preinstall.am
25 Generating ./cpukit/zlib/preinstall.am
26 /c/opt/rtems/kernel/rtems

```

Finally we run the RSB's parallel bootstrap command:

```

1 $ /c/opt/rtems/rsb/source-builder/sb-bootstrap
2 RTEMS Source Builder - RTEMS Bootstrap, 4.11 (76188ee494dd)
3 1/139: autoreconf: configure.ac
4 2/139: autoreconf: c/configure.ac
5 3/139: autoreconf: c/src/configure.ac

```

```

6 4/139: autoreconf: c/src/ada-tests/configure.ac
7 5/139: autoreconf: c/src/lib/libbsp/arm/configure.ac
8 6/139: autoreconf: c/src/lib/libbsp/arm/altera-cyclone-v/configure.ac
9 7/139: autoreconf: c/src/lib/libbsp/arm/atsam/configure.ac
10 8/139: autoreconf: c/src/lib/libbsp/arm/beagle/configure.ac
11 9/139: autoreconf: c/src/lib/libbsp/arm/csb336/configure.ac
12 10/139: autoreconf: c/src/lib/libbsp/arm/csb337/configure.ac
13 11/139: autoreconf: c/src/lib/libbsp/arm/edb7312/configure.ac
14 .....
15 129/139: autoreconf: testsuites/samples/configure.ac
16 130/139: autoreconf: testsuites/smptests/configure.ac
17 131/139: autoreconf: testsuites/sptests/configure.ac
18 132/139: autoreconf: testsuites/tmtests/configure.ac
19 133/139: autoreconf: testsuites/tools/configure.ac
20 134/139: autoreconf: testsuites/tools/generic/configure.ac
21 135/139: autoreconf: tools/build/configure.ac
22 136/139: autoreconf: tools/cpu/configure.ac
23 137/139: autoreconf: tools/cpu/generic/configure.ac
24 138/139: autoreconf: tools/cpu/nios2/configure.ac
25 139/139: autoreconf: tools/cpu/sh/configure.ac
26 Bootstrap time: 0:20:38.759766
27 /c/opt/rtems/kernel/rtems
28 $

```

We will build the RTEMS kernel for the i386 target and the pc686 BSP. You can check the available BSPs by running the `rtems-bsps` command found in the top directory of the RTEMS kernel source. We build the Board Support Package (BSP) outside the kernel source tree:

```

1 /c/opt/rtems/kernel/rtems
2 $ cd ..
3 /c/opt/rtems/kernel
4 $ mkdir pc686
5 /c/opt/rtems/kernel
6 $ cd pc686
7 /c/opt/rtems/kernel/pc686
8 $

```

Configure the RTEMS kernel to build pc686 BSP for the i386 target with networking disabled, We will build the external libBSD stack later:

```

1 /c/opt/rtems/kernel/pc686
2 $ /c/opt/rtems/kernel/rtems/configure --prefix=/c/opt/rtems/4.11 \
3     --target=i386-rtems4.11 --disable-networking --enable-rtemsbsp=pc686
4 checking for gmake... no
5 checking for make... make
6 checking for RTEMS Version... 4.11.99.0
7 checking build system type... x86_64-pc-mingw64
8 checking host system type... x86_64-pc-mingw64
9 checking target system type... i386-pc-rtems4.11
10 checking for a BSD-compatible install... /usr/bin/install -c
11 checking whether build environment is sane... yes
12 checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
13 checking for gawk... gawk
14 checking whether make sets $(MAKE)... yes
15 checking whether to enable maintainer-specific portions of Makefiles... no
16 checking that generated files are newer than configure... done
17 configure: creating ./config.status

```

```

18 configure: configuring in ./tools/build
19 .....
20 checking whether make sets $(MAKE)... yes
21 checking whether to enable maintainer-specific portions of Makefiles... no
22 checking that generated files are newer than configure... done
23 configure: creating ./config.status
24 config.status: creating Makefile
25
26 target architecture: i386.
27 available BSPs: pc686.
28 'make all' will build the following BSPs: pc686.
29 other BSPs can be built with 'make RTEMS_BSP="bsp1 bsp2 ..."'
30
31 config.status: creating Makefile
32 /c/opt/rtems/kernel/pc686
33 $

```

### Build the kernel:

```

1 /c/opt/rtems/kernel/pc686
2 $ make
3 Making all in tools/build
4 make[1]: Entering directory '/c/opt/rtems/kernel/pc686/tools/build'
5 make all-am
6 make[2]: Entering directory '/c/opt/rtems/kernel/pc686/tools/build'
7 gcc -DHAVE_CONFIG_H -I. -I/c/opt/rtems/kernel/rtems/tools/build -g -O2 -MT
8 cklength.o -MD -MP -MF .deps/cklength.Tpo -c -o cklength.o
9 /c/opt/rtems/kernel/rtems/tools/build/cklength.c
10 gcc -DHAVE_CONFIG_H -I. -I/c/opt/rtems/kernel/rtems/tools/build -g -O2 -MT
11 eolstrip.o -MD -MP -MF .deps/eolstrip.Tpo -c -o eolstrip.o
12 /c/opt/rtems/kernel/rtems/tools/build/eolstrip.c
13 .....
14 i386-rtems4.11-objcopy -O binary nsecs.nxe nsecs.bin
15 ../../../../pc686/build-tools/bin2boot -v nsecs.ralf 0x00097E00
16 ../../../../pc686/lib/start16.bin 0x00097C00 0 nsecs.bin 0x00100000 0
17 header address 0x00097e00, its memory size 0xzx
18 first image address 0x00097c00, its memory size 0x0000200
19 second image address 0x00100000, its memory size 0x0003d800
20 rm -f nsecs.nxe
21 make[6]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites/
↳samples/nsecs'
22 make[5]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites/
↳samples'
23 make[4]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites/
↳samples'
24 make[4]: Entering directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites'
25 make[4]: Nothing to be done for 'all-am'.
26 make[4]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites'
27 make[3]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites'
28 make[2]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686'
29 make[1]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c'
30 make[1]: Entering directory '/c/opt/rtems/kernel/pc686'
31 make[1]: Nothing to be done for 'all-am'.
32 make[1]: Leaving directory '/c/opt/rtems/kernel/pc686'
33 /c/opt/rtems/kernel/pc686
34 $

```

Install the kernel to our prefix:

```

1 /c/opt/rtems/kernel/pc686
2 $ make install
3 Making install in tools/build
4 make[1]: Entering directory '/c/opt/rtems/kernel/pc686/tools/build'
5 make[2]: Entering directory '/c/opt/rtems/kernel/pc686/tools/build'
6 /usr/bin/mkdir -p '/c/opt/rtems/4.11/bin'
7 /usr/bin/install -c cklength.exe eolstrip.exe packhex.exe unhex.exe
8 rtems-bin2c.exe '/c/opt/rtems/4.11/bin'
9 /usr/bin/mkdir -p '/c/opt/rtems/4.11/bin'
10 /usr/bin/install -c install-if-change '/c/opt/rtems/4.11/bin'
11 make[2]: Nothing to be done for 'install-data-am'.
12 make[2]: Leaving directory '/c/opt/rtems/kernel/pc686/tools/build'
13 make[1]: Leaving directory '/c/opt/rtems/kernel/pc686/tools/build'
14 Making install in tools/cpu
15 make[1]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu'
16 Making install in generic
17 make[2]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu/generic'
18 make[3]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu/generic'
19 make[3]: Nothing to be done for 'install-exec-am'.
20 make[3]: Nothing to be done for 'install-data-am'.
21 make[3]: Leaving directory '/c/opt/rtems/kernel/pc686/tools/cpu/generic'
22 make[2]: Leaving directory '/c/opt/rtems/kernel/pc686/tools/cpu/generic'
23 make[2]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu'
24 make[3]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu'
25 make[3]: Nothing to be done for 'install-exec-am'.
26 make[3]: Nothing to be done for 'install-data-am'.
27 .....
28 make[2]: Entering directory '/c/opt/rtems/kernel/pc686'
29 make[2]: Nothing to be done for 'install-exec-am'.
30 /usr/bin/mkdir -p '/c/opt/rtems/4.11/make'
31 /usr/bin/install -c -m 644 /c/opt/rtems/kernel/rtems/make/main.cfg
32 /c/opt/rtems/kernel/rtems/make/leaf.cfg '/c/opt/rtems/4.11/make'
33 /usr/bin/mkdir -p '/c/opt/rtems/4.11/share/rtems4.11/make/Templates'
34 /usr/bin/install -c -m 644
35 /c/opt/rtems/kernel/rtems/make/Templates/Makefile.dir
36 /c/opt/rtems/kernel/rtems/make/Templates/Makefile.leaf
37 /c/opt/rtems/kernel/rtems/make/Templates/Makefile.lib
38 '/c/opt/rtems/4.11/share/rtems4.11/make/Templates'
39 /usr/bin/mkdir -p '/c/opt/rtems/4.11/make/custom'
40 /usr/bin/install -c -m 644 /c/opt/rtems/kernel/rtems/make/custom/default.cfg
41 '/c/opt/rtems/4.11/make/custom'
42 make[2]: Leaving directory '/c/opt/rtems/kernel/pc686'
43 make[1]: Leaving directory '/c/opt/rtems/kernel/pc686'
44 /c/opt/rtems/kernel/pc686
45 $

```



## 7.4 RTEMS Kernel

RTEMS is an open source real-time operating system. As a user you have access to all the source code. The RTEMS Kernel section will show you how you build the RTEMS kernel on your host.

### 7.4.1 Development Sources

Create a new location to build the RTEMS kernel:

```
1 $ cd
2 $ cd development/rtems
3 $ mkdir kernel
4 $ cd kernel
```

Clone the RTEMS repository:

```
1 $ git clone git://git.rtems.org/rtems.git rtems
2 Cloning into 'rtems'...
3 remote: Counting objects: 483342, done.
4 remote: Compressing objects: 100% (88974/88974), done.
5 remote: Total 483342 (delta 390053), reused 475669 (delta 383809)
6 Receiving objects: 100% (483342/483342), 69.88 MiB | 1.37 MiB/s, done.
7 Resolving deltas: 100% (390053/390053), done.
8 Checking connectivity... done.
```

### 7.4.2 Tools Path Set Up

We need to set our path to include the RTEMS tools we built in the previous section. The RTEMS tools needs to be first in your path because RTEMS provides specific versions of the autoconf and automake tools. We want to use the RTEMS version and not your host's versions:

```
1 $ export PATH=$HOME/development/rtems/5/bin:$PATH
```

### 7.4.3 Bootstrapping

The developers version of the code from git requires we bootstrap the source code. This is an autoconf and automake bootstrap to create the various files generated by autoconf and automake. RTEMS does not keep these generated files under version control. The bootstrap process is slow so to speed it up the RSB provides a command that can perform the bootstrap in parallel using your available cores. We need to enter the cloned source directory then run the bootstrap commands:

```
1 $ cd rtems
2 $ ./bootstrap -c && ./bootstrap -p && \
3     $HOME/development/rtems/rsb/source-builder/sb-bootstrap
4 removing automake generated Makefile.in files
5 removing configure files
6 removing aclocal.m4 files
7 Generating ./cpukit/dtc/libfdt/preinstall.am
8 Generating ./cpukit/zlib/preinstall.am
9 Generating ./cpukit/libdl/preinstall.am
```



```

10 Generating ./cpukit/posix/preinstall.am
11 Generating ./cpukit/pppd/preinstall.am
12 Generating ./cpukit/librpc/preinstall.am
13 Generating ./cpukit/preinstall.am
14 Generating ./cpukit/sapi/preinstall.am
15 Generating ./cpukit/score/preinstall.am
16 Generating ./cpukit/score/cpu/mips/preinstall.am
17 Generating ./cpukit/score/cpu/sh/preinstall.am
18 Generating ./cpukit/score/cpu/sparc/preinstall.am
19 Generating ./cpukit/score/cpu/no_cpu/preinstall.am
20 Generating ./cpukit/score/cpu/arm/preinstall.am
21 Generating ./cpukit/score/cpu/m32c/preinstall.am
22 Generating ./cpukit/score/cpu/moxie/preinstall.am
23 Generating ./cpukit/score/cpu/v850/preinstall.am
24 Generating ./cpukit/score/cpu/sparc64/preinstall.am
25 Generating ./cpukit/score/cpu/or1k/preinstall.am
26 Generating ./cpukit/score/cpu/i386/preinstall.am
27 Generating ./cpukit/score/cpu/nios2/preinstall.am
28 Generating ./cpukit/score/cpu/epiphany/preinstall.am
29 Generating ./cpukit/score/cpu/m68k/preinstall.am
30 Generating ./cpukit/score/cpu/lm32/preinstall.am
31 Generating ./cpukit/score/cpu/powerpc/preinstall.am
32 Generating ./cpukit/score/cpu/bfin/preinstall.am
33 Generating ./cpukit/libpci/preinstall.am
34 Generating ./cpukit/libcrypt/preinstall.am
35 Generating ./cpukit/rtems/preinstall.am
36 Generating ./cpukit/telnetd/preinstall.am
37 Generating ./cpukit/libnetworking/preinstall.a
38 .....
39 Generating ./c/src/lib/libbsp/powerpc/gen5200/preinstall.am
40 Generating ./c/src/lib/libbsp/powerpc/mpc55xxevb/preinstall.am
41 Generating ./c/src/lib/libbsp/bfin/TLL6527M/preinstall.am
42 Generating ./c/src/lib/libbsp/bfin/bf537Stamp/preinstall.am
43 Generating ./c/src/lib/libbsp/bfin/eZKit533/preinstall.am
44 Generating ./c/src/librtems++/preinstall.am
45 Generating ./c/src/libchip/preinstall.am
46 Generating ./c/src/wrapup/preinstall.am
47 Generating ./c/src/ada/preinstall.am
48 RTEMS Source Builder - RTEMS Bootstrap, 5 (089327b5dcf9)
49 1/139: autoreconf: configure.ac
50 2/139: autoreconf: cpukit/configure.ac
51 3/139: autoreconf: tools/cpu/configure.ac
52 4/139: autoreconf: tools/cpu/generic/configure.ac
53 5/139: autoreconf: tools/cpu/sh/configure.ac
54 6/139: autoreconf: tools/cpu/nios2/configure.ac
55 7/139: autoreconf: tools/build/configure.ac
56 8/139: autoreconf: doc/configure.ac
57 .....
58 124/139: autoreconf: c/src/make/configure.ac
59 125/139: autoreconf: c/src/librtems++/configure.ac
60 126/139: autoreconf: c/src/ada-tests/configure.ac
61 127/139: autoreconf: testsuites/configure.ac
62 128/139: autoreconf: testsuites/libtests/configure.ac
63 129/139: autoreconf: testsuites/mptests/configure.ac
64 130/139: autoreconf: testsuites/fstests/configure.ac
65 131/139: autoreconf: testsuites/sptests/configure.ac
66 132/139: autoreconf: testsuites/tmtests/configure.ac

```

```

67 133/139: autoreconf: testsuites/smptests/configure.ac
68 134/139: autoreconf: testsuites/tools/configure.ac
69 135/139: autoreconf: testsuites/tools/generic/configure.ac
70 136/139: autoreconf: testsuites/psxtests/configure.ac
71 137/139: autoreconf: testsuites/psxtmtests/configure.ac
72 138/139: autoreconf: testsuites/rhealstone/configure.ac
73 139/139: autoreconf: testsuites/samples/configure.ac
74 Bootstrap time: 0:02:47.398824

```

#### 7.4.4 Building a BSP

We build RTEMS in a directory outside of the source tree we have just cloned and bootstrapped. You cannot build RTEMS while in the source tree. Lets create a suitable directory using the name of the BSP we are going to build:

```

1 $ cd ..
2 $ mkdir erc32
3 $ cd erc32

```

Configure RTEMS using the configure command. We use a full path to configure so the object files built contain the absolute path of the source files. If you are source level debugging you will be able to access the source code to RTEMS from the debugger. We will build for the erc32 BSP with POSIX enabled and the networking stack disabled:

```

1 $ $HOME/development/rtems/kernel/rtems/configure --prefix=$HOME/development/rtems/5 \
2           --target=sparc-rtems5 --enable-rtemsbsp=erc32 --enable-posix \
3           --disable-networking
4 checking for gmake... no
5 checking for make... make
6 checking for RTEMS Version... 4.11.99.0
7 checking build system type... x86_64-pc-linux-gnu
8 checking host system type... x86_64-pc-linux-gnu
9 checking target system type... sparc-unknown-rtems5
10 checking for a BSD-compatible install... /usr/bin/install -c
11 checking whether build environment is sane... yes
12 checking for a thread-safe mkdir -p... /bin/mkdir -p
13 checking for gawk... no
14 checking for mawk... mawk
15 checking whether make sets $(MAKE)... yes
16 checking whether to enable maintainer-specific portions of Makefiles... no
17 checking that generated files are newer than configure... done
18 .....
19 checking target system type... sparc-unknown-rtems5
20 checking rtems target cpu... sparc
21 checking for a BSD-compatible install... /usr/bin/install -c
22 checking whether build environment is sane... yes
23 checking for sparc-rtems5-strip... sparc-rtems5-strip
24 checking for a thread-safe mkdir -p... /bin/mkdir -p
25 checking for gawk... no
26 checking for mawk... mawk
27 checking whether make sets $(MAKE)... yes
28 checking whether to enable maintainer-specific portions of Makefiles... no
29 checking that generated files are newer than configure... done
30 configure: creating ./config.status
31 config.status: creating Makefile

```

```

32
33 target architecture: sparc.
34 available BSPs: erc32.
35 'make all' will build the following BSPs: erc32.
36 other BSPs can be built with 'make RTEMS_BSP="bsp1 bsp2 ..."'
37
38 config.status: creating Makefile

```

### Build RTEMS using two cores:

```

1 $ make -j 2
2 Making all in tools/build
3 make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/build'
4 make all-am
5 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/build'
6 gcc -DHAVE_CONFIG_H -I. -I/home/chris/development/rtems/kernel/rtems/tools/build -
↳ -g -O2 -MT cklength.o -MD -MP -MF .deps/cklength.Tpo -c -o cklength.o /home/chris/
↳ development/rtems/kernel/rtems/tools/build/cklength.c
7 gcc -DHAVE_CONFIG_H -I. -I/home/chris/development/rtems/kernel/rtems/tools/build -
↳ -g -O2 -MT eolstrip.o -MD -MP -MF .deps/eolstrip.Tpo -c -o eolstrip.o /home/chris/
↳ development/rtems/kernel/rtems/tools/build/eolstrip.c
8 mv -f .deps/cklength.Tpo .deps/cklength.Po
9 mv -f .deps/eolstrip.Tpo .deps/eolstrip.Po
10 gcc -DHAVE_CONFIG_H -I. -I/home/chris/development/rtems/kernel/rtems/tools/build -g -
↳ -O2 -MT compat.o -MD -MP -MF .deps/compat.Tpo -c -o compat.o /home/chris/development/
↳ rtems/kernel/rtems/tools/build/compat.c
11 gcc -DHAVE_CONFIG_H -I. -I/home/chris/development/rtems/kernel/rtems/tools/build -g -
↳ -O2 -MT packhex.o -MD -MP -MF .deps/packhex.Tpo -c -o packhex.o /home/chris/development/
↳ rtems/kernel/rtems/tools/build/packhex.c
12 mv -f .deps/compat.Tpo .deps/compat.Po
13 gcc -DHAVE_CONFIG_H -I. -I/home/chris/development/rtems/kernel/rtems/tools/build -g -
↳ -O2 -MT unhex.o -MD -MP -MF .deps/unhex.Tpo -c -o unhex.o /home/chris/development/rtems/
↳ kernel/rtems/tools/build/unhex.c
14 mv -f .deps/packhex.Tpo .deps/packhex.Po
15 gcc -DHAVE_CONFIG_H -I. -I/home/chris/development/rtems/kernel/rtems/tools/build -g -
↳ -O2 -MT rtems-bin2c.o -MD -MP -MF .deps/rtems-bin2c.Tpo -c -o rtems-bin2c.o /home/chris/
↳ development/rtems/kernel/rtems/tools/build/rtems-bin2c.c
16 mv -f .deps/unhex.Tpo .deps/unhex.Po
17 gcc -DHAVE_CONFIG_H -I. -I/home/chris/development/rtems/kernel/rtems/tools/build -
↳ -g -O2 -MT binpatch.o -MD -MP -MF .deps/binpatch.Tpo -c -o binpatch.o /home/chris/
↳ development/rtems/kernel/rtems/tools/build/binpatch.c
18 mv -f .deps/rtems-bin2c.Tpo .deps/rtems-bin2c.Po
19 gcc -g -O2 -o cklength cklength.o
20 mv -f .deps/binpatch.Tpo .deps/binpatch.Po
21 gcc -g -O2 -o eolstrip eolstrip.o compat.o
22 gcc -g -O2 -o packhex packhex.o
23 gcc -g -O2 -o rtems-bin2c rtems-bin2c.o compat.o
24 gcc -g -O2 -o unhex unhex.o compat.o
25 gcc -g -O2 -o binpatch binpatch.o
26 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/build'
27 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/build'
28 Making all in tools/cpu
29 make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
30 Making all in generic
31 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
32 make[2]: Nothing to be done for 'all'.
33 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'

```

```

34 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
35 make[2]: Nothing to be done for 'all-am'.
36 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
37 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
38 Making all in testsuites/tools
39 make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools'
40 Making all in generic
41 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools/
↳generic'
42 make[2]: Nothing to be done for 'all'.
43 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools/
↳generic'
44 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools'
45 make[2]: Nothing to be done for 'all-am'.
46 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools'
47 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools'
48 Making all in sparc-rtems5/c
49 make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/c'
50 Making all in .
51 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/c'
52 Configuring RTEMS_BSP=erc32
53 checking for gmake... no
54 checking for make... make
55 checking build system type... x86_64-pc-linux-gnu
56 checking host system type... sparc-unknown-rtems5
57 .....
58 sparc-rtems5-gcc -B../../../../../../erc32/lib/ -specs bsp_specs -qrtems -DHAVE_CONFIG_H -I.
↳-I/home/chris/development/rtems/kernel/rtems/c/src/../../../../testsuites/samples/nsecs -I.
↳-I/home/chris/development/rtems/kernel/rtems/c/src/../../../../testsuites/samples/./support/
↳include -mcpu=cypress -O2 -g -ffunction-sections -fdata-sections -Wall -Wmissing-
↳prototypes -Wimplicit-function-declaration -Wstrict-prototypes -Wnested-externs -MT.
↳init.o -MD -MP -MF .deps/init.Tpo -c -o init.o /home/chris/development/rtems/kernel/
↳rtems/c/src/../../../../testsuites/samples/nsecs/init.c
59 sparc-rtems5-gcc -B../../../../../../erc32/lib/ -specs bsp_specs -qrtems -DHAVE_CONFIG_H -I.
↳-I/home/chris/development/rtems/kernel/rtems/c/src/../../../../testsuites/samples/nsecs -I.
↳-I/home/chris/development/rtems/kernel/rtems/c/src/../../../../testsuites/samples/./support/
↳include -mcpu=cypress -O2 -g -ffunction-sections -fdata-sections -Wall -Wmissing-
↳prototypes -Wimplicit-function-declaration -Wstrict-prototypes -Wnested-externs -MT.
↳empty.o -MD -MP -MF .deps/empty.Tpo -c -o empty.o /home/chris/development/rtems/kernel/
↳rtems/c/src/../../../../testsuites/samples/nsecs/empty.c
60 mv -f .deps/empty.Tpo .deps/empty.Po
61 mv -f .deps/init.Tpo .deps/init.Po
62 sparc-rtems5-gcc -B../../../../../../erc32/lib/ -specs bsp_specs -qrtems -mcpu=cypress -O2.
↳-g -ffunction-sections -fdata-sections -Wall -Wmissing-prototypes -Wimplicit-function-
↳declaration -Wstrict-prototypes -Wnested-externs -Wl,--gc-sections -mcpu=cypress -o.
↳nsecs.exe init.o empty.o
63 sparc-rtems5-nm -g -n nsecs.exe > nsecs.num
64 sparc-rtems5-size nsecs.exe
65 text data bss dec hex filename
66 121392 1888 6624 129904 1fb70 nsecs.exe
67 cp nsecs.exe nsecs.raft
68 make[6]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/ c/
↳erc32/testsuites/samples/nsecs'
69 make[5]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/ c/
↳erc32/testsuites/samples'
70 make[4]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/ c/
↳erc32/testsuites/samples'

```

```

71 make[4]: Entering directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/ c/
↪erc32/testsuites'
72 make[4]: Nothing to be done for 'all-am'.
73 make[4]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/ c/
↪erc32/testsuites'
74 make[3]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/ c/
↪erc32/testsuites'
75 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/ c/
↪erc32'
76 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/c'
77 make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32'
78 make[1]: Nothing to be done for 'all-am'.
79 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32'

```

### 7.4.5 Installing A BSP

All that remains to be done is to install the kernel. Installing RTEMS copies the API headers and architecture specific libraries to a location under the *prefix* you provide. You can install any number of BSPs under the same *prefix*. We recommend you have a separate *prefix* for different versions of RTEMS. Do not mix versions of RTEMS under the same *prefix*. Make installs RTEMS with the following command:

```

1 $ make install
2 Making install in tools/build
3 make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/build'
4 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/build'
5 /bin/mkdir -p '/home/chris/development/rtems/5/bin'
6 /usr/bin/install -c cklength eolstrip packhex unhex rtems-bin2c '/home/chris/development/
↪rtems/5/bin'
7 /bin/mkdir -p '/home/chris/development/rtems/5/bin'
8 /usr/bin/install -c install-if-change '/home/chris/development/rtems/5/bin'
9 make[2]: Nothing to be done for 'install-data-am'.
10 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/build'
11 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/build'
12 Making install in tools/cpu
13 make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
14 Making install in generic
15 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
16 make[3]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
17 make[3]: Nothing to be done for 'install-exec-am'.
18 make[3]: Nothing to be done for 'install-data-am'.
19 make[3]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
20 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
21 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
22 make[3]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
23 make[3]: Nothing to be done for 'install-exec-am'.
24 make[3]: Nothing to be done for 'install-data-am'.
25 make[3]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
26 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
27 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
28 .....
29 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems5/c'
30 make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32'
31 make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32'
32 make[2]: Nothing to be done for 'install-exec-am'.

```

```

33 /bin/mkdir -p '/home/chris/development/rtems/5/make'
34 /usr/bin/install -c -m 644 /home/chris/development/rtems/kernel/rtems/make/main.cfg /home/
↪chris/development/rtems/kernel/rtems/make/leaf.cfg '/home/chris/development/rtems/5/make
↪'
35 /bin/mkdir -p '/home/chris/development/rtems/5/share/rtems5/make/Templates'
36 /usr/bin/install -c -m 644 /home/chris/development/rtems/kernel/rtems/make/Templates/
↪Makefile.dir /home/chris/development/rtems/kernel/rtems/make/Templates/Makefile.leaf
↪/home/chris/development/rtems/kernel/rtems/make/Templates/Makefile.lib '/home/chris/
↪development/rtems/5/share/rtems5/make/Templates'
37 /bin/mkdir -p '/home/chris/development/rtems/5/make/custom'
38 /usr/bin/install -c -m 644 /home/chris/development/rtems/kernel/rtems/make/custom/default.
↪cfg '/home/chris/development/rtems/5/make/custom'
39 make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32'
40 make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32'

```

## 7.4.6 Contributing Patches

RTEMS welcomes fixes to bugs and new features. The RTEMS Project likes to have bugs fixed against a ticket created on our [Developer Site](#). Please raise a ticket if you have a bug. Any changes that are made can be tracked against the ticket. If you want to add a new a feature please post a message to [Developers Mailing List](#) describing what you would like to implement. The RTEMS maintainer will help decide if the feature is in the best interest of the project. Not everything is and the maintainers need to evaluate how much effort it is to maintain the feature. Once accepted into the source tree it becomes the responsibility of the maintainers to keep the feature updated and working.

Changes to the source tree are tracked using git. If you have not made changes and enter the source tree and enter a git status command you will see nothing has changed:

```

1 $ cd ../rtems
2 $ git status
3 On branch master
4 Your branch is up-to-date with 'origin/master'.
5 nothing to commit, working directory clean

```

We will make a change to the source code. In this example I change the help message to the RTEMS shell's halt command. Running the same git status command reports:

```

1 $ git status
2 On branch master
3 Your branch is up-to-date with 'origin/master'.
4 Changes not staged for commit:
5   (use "git add <file>..." to update what will be committed)
6   (use "git checkout -- <file>..." to discard changes in working directory)
7
8       modified:   cpukit/libmisc/shell/main_halt.c
9
10 no changes added to commit (use "git add" and/or "git commit -a")

```

As an example I have a ticket open and the ticket number is 9876. I commit the change with the follow git command:

```

1 $ git commit cpukit/libmisc/shell/main_halt.c

```

An editor is opened and I enter my commit message. The first line is a title and the following lines form a body. My message is:

```
1 shell: Add more help detail to the halt command.
2
3 Closes #9876.
4
5 # Please enter the commit message for your changes. Lines starting
6 # with '#' will be ignored, and an empty message aborts the commit.
7 # Explicit paths specified without -i or -o; assuming --only paths...
8 #
9 # Committer: Chris Johns <chrisj@rtems.org>
10 #
11 # On branch master
12 # Your branch is up-to-date with 'origin/master'.
13 #
14 # Changes to be committed:
15 #   modified:   cpukit/libmisc/shell/main_halt.c
```

When you save and exit the editor git will report the commit's status:

```
1 $ git commit cpukit/libmisc/shell/main_halt.c
2 [master 9f44dc9] shell: Add more help detail to the halt command.
3 1 file changed, 1 insertion(+), 1 deletion(-)
```

You can either email the patch to [Developers Mailing List](#) with the following git command, and it is *minus one* on the command line:

```
1 $ git send-email --to=devel@rtems.org -1
2 <add output here>
```

Or you can ask git to create a patch file using:

```
1 $ git format-patch -1
2 0001-shell-Add-more-help-detail-to-the-halt-command.patch
```

This patch can be attached to a ticket.





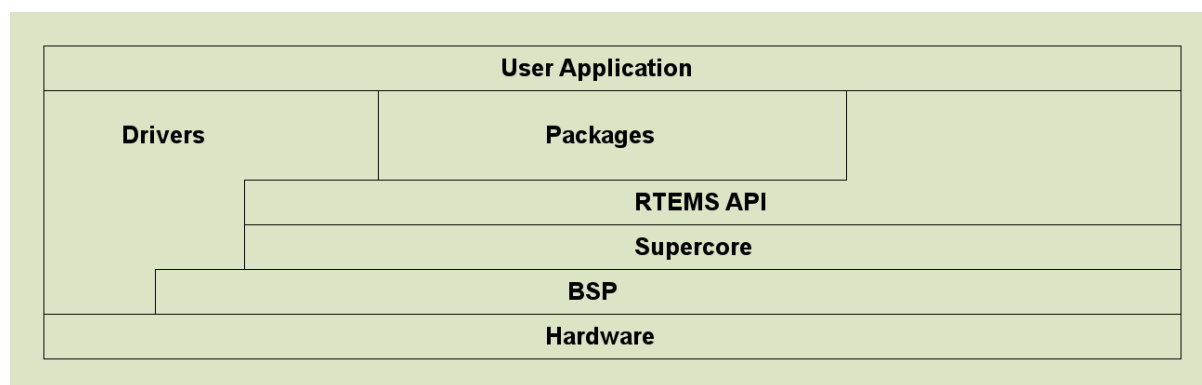
# HARDWARE

This section discusses supported Hardware, Architectures, Board Support Packages, and running RTEMS on real hardware and simulators.

## 8.1 Targets

Hardware that can run RTEMS is often referred to as a *target* because RTEMS is specifically aimed at that hardware or target. An RTEMS executable is statically linked and executes in a single address space on the target hardware. A statically linked executable means the RTEMS Kernel, drivers, third party packages and application code is linked into a single executable image. A single address space means no virtual memory and no memory protected process address space is running within the RTEMS arena and the RTEMS Kernel, drivers and application have unprotected access to the whole address space and all hardware.

Target hardware supported by RTEMS has a Board Support Package or BSP. A BSP is a specific instance of an RTEMS architecture that allows the creation of an RTEMS executable. You can view the layering as:



RTEMS Targets are grouped by architectures and within an architecture there are a number of Board Support Packages or BSPs. An architecture is a specific class or family of processors and can be large such as ARM or specific such as the NIOS-II or Microblaze.

RTEMS is designed to be ported to new target hardware easily and efficiently.

## 8.2 Architectures

An RTEMS architecture is a class or family of a processor that RTEMS supports. The RTEMS architecture model follows the architecture model of GCC. An architecture in GCC results in a specific RTEMS GCC compiler. This compiler may support a range of processors in the family that may have differences in instructions sets or floating point support. RTEMS configures GCC to create separate runtime libraries for each supported instruction set and floating point unit in the architecture. This is termed **multilib**. Multilibs are managed automatically by GCC by selecting a specific instruction set or specific device in a family.

RTEMS executables are statically linked for a specific target therefore a precise and exact match can be made for the hardware that extracts the best possible performance. The compiler supports the variants to the instruction set and RTEMS extends the specialization to specific processors in an architecture. This specialization gives RTEMS a finer resolution of features and capabilities a specific device may offer allowing the kernel, drivers and application to make the most of those resources. The trade off is portability however this is not important because the executables are statically linked for a single target.

---

**Note:** RTEMS support dynamically load code through the `dlopen` interface. Loading code via this interface results in an executable image that is equivalent to statically linked executable of the same code. Dynamic loading is a system level tool for system architects.

---

RTEMS supports 17 architectures:

- arm
- bfin
- epiphany
- i386
- lm32
- m32c
- m68k
- mips
- moxie
- nios2
- no\_cpu
- or1k
- powerpc
- sh
- sparc
- sparc64
- v850

### 8.3 Board Support Packages (BSP)

A Board Support Package is a historical term for a package of code, and supporting documentation for a target. The separation is still important today for users with custom hardware.

RTEMS includes 173 board support packages in its source tree and this is a small number of actual targets running because it does not take into account the custom targets.

You can see the BSP list in RTEMS by asking RTEMS with:

```
1 $ ./rtems-bsps
```

## 8.4 Tiers

RTEMS has a tiered structure for architecture and BSPs. It provides:

1. A way to determine the state of a BSP in RTEMS.
2. A quality measure for changes entering the RTEMS source code.

The tier structure in RTEMS is support by the Buildbot continuous integration server. Changes to RTEMS are automatically built and tested and the results indicate if a BSP currently meets it's tier status.

The rules for Tiers are:

1. A BSP can only be in one of the following tiers:

Tier	Description
1	<ul style="list-style-type: none"> <li>• The RTEMS Kernel must build without error.</li> <li>• Tests are run on target hardware.</li> </ul>
2	<ul style="list-style-type: none"> <li>• The RTEMS Kernel must build without error.</li> <li>• Tests can be run on simulation.</li> </ul>
3	<ul style="list-style-type: none"> <li>• The RTEMS Kernel must build without error.</li> <li>• There are no test results.</li> </ul>
4	<ul style="list-style-type: none"> <li>• The RTEMS Kernel does not build.</li> </ul>
5	<ul style="list-style-type: none"> <li>• The BSP is to be removed after the next release.</li> </ul>

2. An architecture's tier is set by the highest BSP tier reached.
3. The tier level for a BSP is set by the RTEMS Project team. Movement of BSP between tier level requires agreement. The Buildbot results indicate the current tier level.
4. Changes to RTEMS may result in a BSP not meeting it's tier are acceptable if the change is accompanied by an announcement and a plan on how this is to be resolved.
5. Test results are set on a per BSP basis by the RTEMS Project team. Changes to the test result values requires agreement. The test results are defined as:
  - (a) Passes
  - (b) Expected Failures

Expected failures must be explicitly listed. A BSP is required to have a valid test result entry to reach tier 1.



# BOARD SUPPORT PACKAGES (BSPS)

## 9.1 ARM Board Support Packages

### 9.1.1 NXP i.MX

This BSP offers only one variant, the *imx7*. This variant supports the i.MX 7Dual processor. The basic hardware initialization is not performed by the BSP. A boot loader with device tree support must be used to start the BSP, e.g. U-Boot.

#### 9.1.1.1 Build Configuration Options

The following options are available at the configure command line.

**BSP\_PRESS\_KEY\_FOR\_RESET**

If defined to a non-zero value, then print a message and wait until pressed before resetting board when application terminates.

**BSP\_RESET\_BOARD\_AT\_EXIT**

If defined to a non-zero value, then reset the board when the application terminates.

**BSP\_PRINT\_EXCEPTION\_CONTEXT**

If defined to a non-zero value, then print the exception context when an unexpected exception occurs.

**BSP\_FDT\_BLOB\_SIZE\_MAX**

The maximum size of the device tree blob in bytes (default is 262144).

**CONSOLE\_USE\_INTERRUPTS**

Use interrupt driven mode for console devices (enabled by default).

**IMX\_CCM\_IPG\_HZ**

The IPG clock frequency in Hz (default is 67500000).

**IMX\_CCM\_UART\_HZ**

The UART clock frequency in Hz (default is 24000000).

**IMX\_CCM\_AHB\_HZ**

The AHB clock frequency in Hz (default is 135000000).

#### 9.1.1.2 Boot via U-Boot

The application executable file (ELF file) must be converted to an U-Boot image. Use the following commands:

```
1 arm-rtems5-objcopy -O binary app.exe app.bin
2 gzip -9 -f -c app.bin > app.bin.gz
3 mkimage -A arm -O linux -T kernel -a 0x80200000 -e 0x80200000 -n RTEMS -d app.bin.gz app.
  ↪img
```

Use the following U-Boot commands to boot an application via TFTP download:

```
1 tftpboot ${loadaddr} app.img && run loadfdt && bootm ${loadaddr} - ${fdt_addr} ; reset
```



### 9.1.1.3 Clock Driver

The clock driver uses the *ARMv7-AR Generic Timer*.

### 9.1.1.4 Console Driver

The console driver supports up to seven on-chip UARTs. They are initialized according to the device tree. The console driver does not configure the pins.

### 9.1.1.5 I2C Driver

I2C drivers are registered by the `i2c_bus_register_imx()` function. The I2C driver does not configure the pins.

```
1 #include <assert.h>
2 #include <bsp.h>
3
4 void i2c_init(void)
5 {
6     int rv;
7
8     rv = i2c_bus_register_imx("/dev/i2c-0", "i2c0");
9     assert(rv == 0);
10 }
```

### 9.1.1.6 SPI Driver

SPI drivers are registered by the `spi_bus_register_imx()` function. The SPI driver configures the pins according to the `pinctrl-0` device tree property. SPI transfers with a continuous chip select are limited by the FIFO size of 64 bytes. The driver has no DMA support.

```
1 #include <assert.h>
2 #include <bsp.h>
3
4 void spi_init(void)
5 {
6     int rv;
7
8     rv = spi_bus_register_imx("/dev/spi-0", "spi0");
9     assert(rv == 0);
10 }
```

### 9.1.1.7 Network Interface Driver

The network interface driver is provided by the *libbsd*. It is initialized according to the device tree. It supports checksum offload and interrupt coalescing. IPv6 transmit checksum offload is not implemented. The interrupt coalescing uses the MII/GMII clocks and can be controlled by the following system controls:

- `dev.ffec.<unit>.int_coal.rx_time`
- `dev.ffec.<unit>.int_coal.rx_count`

- `dev.ffec.<unit>.int_coal.tx_time`
- `dev.ffec.<unit>.int_coal.tx_count`

A value of zero for the time or count disables the interrupt coalescing in the corresponding direction.

#### 9.1.1.8 Caveats

The clock and pin configuration support is quite rudimentary and mostly relies on the boot loader. For a pin group configuration see `imx_iomux_configure_pins()`. There is no power management support.

## HOST TOOLS

The RTEMS kernel is developed on host computers cross-compiled and linking the kernel, language runtime libraries, 3rd party packages and application source code so it can run on target hardware. RTEMS and some of the hardware it support cannot self-host so we need a range of tools to support the wide range of available host computers users wish to develop on. This section details the tools available on the host computers to help support RTEMS users and developers.

## 10.1 RTEMS Linker

The RTEMS Linker is an RTEMS tool to help build RTEMS application that dynamically loader code into a running RTEMS system.

[ NOT COMPLETE ]

## 10.2 RTEMS Symbols

The RTEMS Symbols (**rtems-syms**) command is an RTEMS tool to generate symbol tables used by the RTEMS Runtime Loader (RTL). The symbol table contains the exported base kernel symbols user code dynamically loaded can reference.

The RTEMS Runtime Loader supports two methods of loading a symbol table, embedded and runtime loading. Embedding the table requires linking the symbol table with the base image and runtime loading loads the table using the dynamic loader when RTEMS is running.

### *Filtering Symbols*

Currently there is no filtering of symbols in the symbol table. This means all base kernel image symbols are present in the symbol table when only a sub-set of the symbols are referenced.

Embedding the symbol table creates self contained images. A target may not have any external media, for example RTEMS tests, or there is a requirement to avoid the management need to match the symbol table with the kernel base image. Embedding the symbol table requires a 2-pass link process making the application's build system more complicated.

A dynamically loadable symbol table is simpler to create however the symbol table and the kernel base image must match or the behaviour is undefined. There is currently no mechanism to ensure the symbol table and the kernel image match The **rtems-syms** command is run against the base kernel image and the generated symbol table is installed on to the target hardware and loaded before any other modules.

### 10.2.1 Symbol Table

The symbol table is an ELF object file in the target's ELF format and is built using the target's RTEMS C compiler. The **rtems-syms** command searches for the C compiler under the prefix this command is installed under or the system path. If the target's C compiler is not located in either of these paths use the option **-c** or **--cc** to specify the path to the compiler.

The **rtems-syms** command loads the base kernel image's ELF file and reads the global or public symbols, creates a temporary C file and then compiles it using the target's RTEMS C compiler. The command automatically detects the architecture from the base kernel image's ELF file and uses it to create the C compiler's name. The option **-E** or **--exec-prefix** can be used to override the executable prefix used.

It is important to supply suitable C compiler flags (**cflags**) that match the kernel image's so the symbol table can be linked or loaded.

### 10.2.2 2-Pass Linking

2-Pass linking is used to embed a symbol table in a base kernel image. The first link pass is a normal RTEMS kernel link process. The link output is passed to the **rtems-syms** command and the **-e** or **--embed** option is used. The symbol table object file created by **rtems-syms** is added to the linker command used in the first pass to create the second pass. The address map will change between the first pass and second pass without causing a problem, the symbol table embedded in the second link pass will adjust the symbol addresses to match.

### 10.2.3 Command

**rtms-syms** [options] kernel

- V, --version**  
Display the version information and then exit.
- v, --verbose**  
Increase the verbose level by 1. The option can be used more than once to get more detailed trace and debug information.
- w, --warn**  
Enable build warnings. This is useful when debugging symbol table generation.
- k, --keep**  
Do not delete temporary files on exit, keep them.
- e, --embed**  
Create a symbol table that can be embedded in the base kernel image using a 2-pass link process.
- S, --symc**  
Specify the symbol's C source file. The default is to use a temporary file name.
- o, --output**  
Specify the ELF output file name.
- m, --map**  
Create a map file using the provided file name.
- C, --cc**  
Specify the C compile executable file name. The file can be absolute and no path is search or relative and the environment's path is searched.
- E, --exec-prefix**  
Specify the RTEMS tool prefix. For example for RTEMS 5 and the SPARC architecture the prefix is `sparc-rtems5`.
- c, --cflags**  
Specify the C compiler flags used to build the symbol table with. These should be the same or compatible with the flags used to build the RTEMS kernel.
- , -h**  
Reort the usage help.

### 10.2.4 Examples

Create a dynamlically loaded symbol table for the `minimum.exe` sample program for the i386/pc686 BSP:

```
1 $ rtms-syms -o ms.o i386-rtems5/c/pc686/testsuites/samples/minimum/minimum.exe
2 $ file ms.o
3 ms.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not stripped
```

Run the same command, this time create a map file:

```

1 $ rtems-syms -o ms.o -m ms.map i386-rtems5/c/pc686/testsuites/samples/minimum/minimum.exe
2 $ head -10 ms.map
3 RTEMS Kernel Symbols Map
4 kernel: i386-rtems5/c/pc686/testsuites/samples/minimum/minimum.exe
5
6 Globals:
7 No.  Index Scope      Type      SHNDX  Address      Size      Name
8     0   931 STB_GLOBAL STT_OBJECT  11 0x0012df08    4 BSPBaseBaud (minimum.exe)
9     1  1124 STB_GLOBAL STT_OBJECT  11 0x0012d894    4 BSPPrintkPort (minimum.exe)
10    2   836 STB_GLOBAL STT_FUNC    1 0x00104b00   302 BSP_dispatch_isr (minimum.
↔exe)
11    3  1156 STB_GLOBAL STT_FUNC    1 0x001082d0    92 BSP_install_rtems_shared_irq_
↔handler (minimum.exe)
12    4   876 STB_GLOBAL STT_FUNC    1 0x00106500   138 BSP_outch (minimum.exe)

```

Run the same command with a raise verbose level to observe the stages the command performs:

```

1 $ rtems-syms -vvv -o ms.o i386-rtems5/c/pc686/testsuites/samples/minimum/minimum.exe
2 RTEMS Kernel Symbols 5.a72a462adc18
3 kernel: i386-rtems5/c/pc686/testsuites/samples/minimum/minimum.exe
4 cache:load-sym: object files: 1
5 cache:load-sym: symbols: 1043
6 symbol C file: /tmp/rld--X7paaa.c
7 symbol O file: ms.o
8 execute: i386-rtems5-gcc -O2 -c -o ms.o /tmp/rld--X7paaa.c
9 execute: status: 0

```

## 10.3 RTEMS Executable Information

The RTEMS Executable Information (**rtms-exeinfo**) command is an RTEMS tool to display some important parts of an RTEMS executable. RTEMS uses ELF as the final linker output and this tool displays useful RTEMS specific information held in the ELF executable. The tool does not replace tools like *readelf*, rather it focuses on reporting specific information RTEMS builds into the executable.

### 10.3.1 System Initialisation

Linker based system initialisation automatically lets RTEMS only link into the executable the system initialisation code referenced by the user's application or indirectly by RTEMS. The technique is a variation of the system initialisation process used in the FreeBSD kernel. It is also similar to the process used by the C++ language to run static constructors before entering *main* and destructors after *exit* is called.

Linker based system initialisation collects the address of referenced system initialisation functions in specially named sections. The system initialisation function's address is placed in a variable and the section attribute of the variable is set to a special section name. The linker is instructed via a special linker command file to collect these variables together to create a table. The start-up code in RTEMS loops over the table of addresses and calling each address or system initialisation function. Special section names given to the variables sorts the table placing the functions in a specific order.

A user places a call to an API function in their application and the linker pulls the API code from the RTEMS kernel library adding it to the executing. The API code the linker loads references the variable containing the address of that API's system initialisation function. The linker loads the API system initialisation code into the executable to resolve the external reference created by the variable. If the user does not reference the API the variable is loaded into the executable and no reference to the API system initialisation code is made so it is not linked into the executable.

The design automatically creates a unique system initialisation table for each executable and the code in RTEMS does not change, there is no special build system tricks, or stub libraries.

The RTEMS Executable Information reports the tables created and you can use this information to debug any initialisation issues.

### 10.3.2 Command

The **rtms-exeinfo** tool reports RTEMS specific information about the executable. The *init* and *fini* tables print the symbol referenced for each table entry and if the symbol is from the C++ language it is demangled.

#### **rtms-exeinfo**

**-v**

Display the version information and then exit.

**-v**

Increase the verbose level by 1. The option can be used more than once to get more detailed trace and debug information.



- a**  
Report all types of output data.
- I**  
Report the `init` or initialisation table.
- F**  
Report the `fini` or finalisation table.
- S**  
Report the sections.
- , -h**  
Report the usage help.

### 10.3.3 Examples

Prints all reports for the `hello.exe` for the `i386/pc686` BSP:

```

1 $ rtems-exeinfo -a i386-rtems5/c/pc686/testsuites/samples/hello/hello.exe
2 RTEMS Executable Info 5.6f5cfada964c
3 rtems-exeinfo -a i386-rtems5/c/pc686/testsuites/samples/hello/hello.exe
4 exe: i386-rtems5/c/pc686/testsuites/samples/hello/hello.exe
5 Sections: 22
6 ----- addr: 0x00000000 0x00000000 size:      0 align: 0_
↪relocs:      0
7 .bss      WA----- addr: 0x00135760 0x0013b300 size: 23456 align: 32_
↪relocs:      0
8 .comment  ---MS----- addr: 0x00000000 0x00000083 size:   131 align: 1_
↪relocs:      0
9 .ctors    WA----- addr: 0x0013322c 0x00133234 size:     8 align: 4_
↪relocs:      0
10 .data     WA----- addr: 0x00133240 0x0013574c size:  9484 align: 32_
↪relocs:      0
11 .debug_abbrev ----- addr: 0x00000000 0x0003c5ce size: 247246 align: 1_
↪relocs:      0
12 .debug_aranges ----- addr: 0x00000000 0x00003a18 size:  14872 align: 8_
↪relocs:      0
13 .debug_info ----- addr: 0x00000000 0x0032496d size: 3295597 align: 1_
↪relocs:      0
14 .debug_line ----- addr: 0x00000000 0x0006606b size:  417899 align: 1_
↪relocs:      0
15 .debug_loc  ----- addr: 0x00000000 0x0003b704 size:  243460 align: 1_
↪relocs:      0
16 .debug_ranges ----- addr: 0x00000000 0x00008128 size:   33064 align: 1_
↪relocs:      0
17 .debug_str  ---MS----- addr: 0x00000000 0x0001a9d7 size:  109015 align: 1_
↪relocs:      0
18 .dtors     WA----- addr: 0x00133234 0x0013323c size:     8 align: 4_
↪relocs:      0
19 .eh_frame  -A----- addr: 0x0012b884 0x0013222c size:  27048 align: 4_
↪relocs:      0
20 .fini     -AE----- addr: 0x00127fdd 0x00127fe5 size:     8 align: 1_
↪relocs:      0
21 .init     -AE----- addr: 0x00127fd0 0x00127fd0 size:    13 align: 1_
↪relocs:      0
22 .rodata   -A----- addr: 0x00128000 0x0012b884 size:  14468 align: 32_
↪relocs:      0

```

```

23  .rtemsroset    WA----- addr: 0x00127f94 0x00127fd0 size:      60 align:  4_
↳relocs:      0
24  .shstrtab     ----- addr: 0x00000000 0x000000c6 size:      198 align:  1_
↳relocs:      0
25  .strtab       ----- addr: 0x00000000 0x000068ca size:     26826 align:  1_
↳relocs:      0
26  .symtab       ----- addr: 0x00000000 0x00006290 size:     25232 align:  4_
↳relocs:      0
27  .text         WAE----- addr: 0x00100000 0x00127f91 size:    163729 align: 16_
↳relocs:      0
28
29  Init sections: 3
30  .ctors
31  0xffffffff RamSize
32  0x00000000 REG_EFLAGS
33  .init
34  0xfd81ebe8 no symbol
35  0xff86e8ff no symbol
36  0x00c2ffff no symbol
37  .rtemsroset
38  0x00100310 bsp_work_area_initialize
39  0x00100440 bsp_start_default
40  0x001160e0 _User_extensions_Handler_initialization
41  0x0010fe60 rtems_initialize_data_structures
42  0x0010fcf0 _RTEMS_tasks_Manager_initialization
43  0x0010f310 _Semaphore_Manager_initialization
44  0x0010ed90 _POSIX_Keys_Manager_initialization
45  0x00113af0 _Thread_Create_idle
46  0x0010c100 rtems_libio_init
47  0x0010bec0 rtems_filesystem_initialize
48  0x00100420 bsp_predriver_hook
49  0x0010bfb0 _Console_simple_Initialize
50  0x0010ff30 _IO_Initialize_all_drivers
51  0x0010fc10 _RTEMS_tasks_Initialize_user_tasks_body
52  0x0010ccb0 rtems_libio_post_driver
53
54  Fini sections: 2
55  .dtors
56  0xffffffff RamSize
57  0x00000000 REG_EFLAGS
58  .fini
59  0xfd815ee8 no symbol
60  0x0000c2ff no symbol

```

The Init section `.rtemsroset` shows the initialisation call order for the `hello.exe` sample application. The order is initialise the BSP work area, call the BSP start up, initialise the User extensions, initialise the RTEMS data structures, then call the various Classic API managers that have been linked into the application. Next any POSIX managers are initialisations, in this case the POSIX Keys manager which is used by the thread local storage (TLS) support. Finally the IO and file system is initialise followed by the drivers.

Print the Init section data for the `cdtest.exe` for the `i386/pc686` BSP:

```

1 $ rtems-exeinfo -I i386-rtems5/c/pc686/testsuites/samples/cdtest/cdtest.exe
2 RTEMS Executable Info 5.6f5cfada964c
3 rtems-exeinfo -I i386-rtems5/c/pc686/testsuites/samples/cdtest/cdtest.exe
4 exe: i386-rtems5/c/pc686/testsuites/samples/cdtest/cdtest.exe

```

```
5 Init sections: 3
6 .ctors
7 0xffffffff RamSize
8 0x00100ea0 _GLOBAL__sub_I_rtems_test_name
9 0x001015d0 __gnu_cxx:___freeres()
10 0x00101df0 __cxxabiv1:___terminate(void (*)())
11 0x00102ac0 _GLOBAL__sub_I___cxa_get_globals_fast
12 0x00103260 std::nothrow
13 0x00000000 REG_EFLAGS
14 .init
15 0xfcb3dbe8 no symbol
16 0xff86e8ff no symbol
17 0x00c2ffff no symbol
18 .rtemsroset
19 0x001112c0 bsp_work_area_initialize
20 0x001113f0 bsp_start_default
21 0x001276c0 _User_extensions_Handler_initialization
22 0x00121260 rtems_initialize_data_structures
23 0x001210f0 _RTEMS_tasks_Manager_initialization
24 0x00120710 _Semaphore_Manager_initialization
25 0x0011ff70 _POSIX_Keys_Manager_initialization
26 0x001250d0 _Thread_Create_idle
27 0x0011d220 rtems_libio_init
28 0x0011cfe0 rtems_filesystem_initialize
29 0x001113d0 bsp_predriver_hook
30 0x0011d0d0 _Console_simple_Initialize
31 0x00121310 _IO_Initialize_all_drivers
32 0x00121010 _RTEMS_tasks_Initialize_user_tasks_body
33 0x0011ddd0 rtems_libio_post_driver
```

The C++ constructor section `.ctors` shows you the C++ static objects the RTEMS kernel will construct before calling `main`.

## 10.4 RTEMS BSP Builder

The RTEMS BSP Builder is an RTEMS developer tool to build RTEMS in ways users do not test changes to RTEMS. RTEMS has large number of architectures, board support packages and configuration options. This tool provides a standard way to test a change.

### 10.4.1 Developer Workflows

There are a number of RTEMS developers each with a different view or expertise in RTEMS. Developers can work in the generic areas such as scheduling, file systems or the shell, or users can become developers adding a new BSP, or even a new port to a new architecture. A common approach for all these developers is to select a BSP and to work with that BSP. Developers working in a generic areas of RTEMS tend to select a BSP that has good simulator support with good debugging such as QEMU, while developers of a new BSP or a new port tend to work on target hardware. This type of development does not check the other architectures, BSP, and build options and a change may change the number of warnings or introduce build errors. It is important for the RTEMS project to have developers fix these issues before pushing the changes to the master repository to avoid breaking the code for other developers. It is best for a developer to resolve as many issues as they work on changes because coming back to a problem often proves difficult.

The RTEMS BSP Builder forms part of a developers workflow where patches are tested before being pushed to the repository.

### 10.4.2 Build Characteristics

Build characteristic are the various parts of a build that can varied changing what is built. RTEMS can vary builds based on:

1. Architecture
2. Board Support Package (BSP)
3. Build options
4. BSP Options

The BSP Builder provides a template of builds to try and reduce the possible combinations to something manageable. It is not realistic to build all possible combinations on a single machine in reasonable time.

The RTEMS BSP Builder specifies it builds in terms of:

1. Profiles
2. Architectures
3. BSPs
4. Builds

The RTEMS BSP Builder builds are created by user options that vary these parameters.

### 10.4.2.1 Profiles

A profile is named collection of architectures and board support packages. When the RTEMS BSP Builder is asked to build a specific profile it builds the BSPs in the specified architectures.

The default configuration provides standard profiles for *Chapter 8 Section 4 - Tiers*. They are:

1. tier-1 (default)
2. tier-2
3. tier-3
4. tier-4

The everythings profile allows all BSPs to be built.

### 10.4.2.2 Builds

A build is a list of builds or a build set and each BSP in a profile, architecture of BSP is built with.

The default configuration provides standard builds based around the commonly varied configure options.

The builds are:

1. all (default)
2. tests
3. standard, also no-tests
4. debug
5. profiling
6. smp
7. smp-debug
8. posix
9. no-posix
10. posix-debug
11. posix-profiling
12. network
13. no-network
14. network-debug
15. smp-network
16. smp-network-debug

## All Build

The all build is:

- debug
- profiling
- smp
- smp-debug
- posix
- no-posix
- posix-debug
- posix-profiling
- network
- no-network
- network-debug
- smp-network
- smp-network-debug

A build maps to specific configuration options. The mappings are:

debug	config:base, config:debug
profiling	config:base, config:profiling
smp	config:base, config:smp
smp-debug	config:base, config:smp, config:debug
posix	config:base, config:posix
no-posix	config:base, config:no-posix
posix-debug	config:base, config:posix, config:debug
posix-profiling	config:base, config:posix, config:profiling
network	config:base, config:network
no-network	config:base, config:no-network
network-debug	config:base, config:network, config:debug
smp-network	config:base, config:smp, config:network
smp-network-debug	config:base, config:smp, config:network, config:debug

### 10.4.3 Build Configurations

Build configurations are configure options. These are mapped to the various builds. The configurations are:

base	--target=@ARCH@-rtems@RTEMS_VERSION@ --enable-rtemsbSP=@BSP@ --prefix=@PREFIX@
tests	--enable-tests
debug	--enable-debug
no-debug	--disable-debug
profiling	--enable-profiling
no-profiling	--disable-profiling
smp	--enable-smp
no-smp	--disable-smp
posix	--enable-posix
no-posix	--disable-posix
network	--enable-networking
no-network	--disable-networking

#### 10.4.4 Performance

The RTEMS BSP Builder is designed to extract the maximum performance from your hardware when building RTEMS. The RTEMS build system is based on autoconf, automake and GNU make. Building consists of two phases:

1. Configuring
2. Building

The Configuring phase and the start of the Build phase runs autoconf's configure scripts. These execute as a single linear process and are not run in parallel even if you specify more than one job to make. The configure part of a build is approximately 30% of the total time and higher if building the tests. Performing a single build at a time will not fully utilize a multi-core machine because of the large amount of time the system is idle.

The RTEMS BSP Builder can run more than one build in parallel. A build can also request make run its build with more than one job. The --jobs option lets a user specify the number of build jobs to run at once and the number of make jobs each build runs with. Together these options can fully load a system and can overload a machine.

Tuning the best ratio of build jobs to make jobs requires running some builds and observing the system's performance. If the build job count is too low the system will show idle periods and if you have too many build jobs with too many make jobs the system will have too many processes running and the operating system's overheads in administering too processes at once lowers the overall performance.

A fast eight core machine where the operating system shows sixteen cores can support a build option of --jobs=5/10. The machine will be fully loaded the average build time is around 18 seconds.

The type of build selected effects the optimum jobs option. For example building the tests changes the percentage of time spent configuring compared to building so the make jobs parameter becomes a dominant factor. Lowering the make jobs value avoids having too many active processes running at once.

### 10.4.5 Command

**rtems-bsp-builder** [options]

**-?**

Display a compact help.

**-h, --help**

Display the full help.

**--prefix**

Prefix to pass to configure when building a BSP.

**--rtems-tools**

The path the RTEMS tools such as the C compiler. This option avoid polluting your path. This path is to the tool's prefix used to build and install the tools and not exact path to an executable.

**--rtems**

The path the RTEMS source tree to build.

**--build-path**

The path to build the BSP and place the build output. This can be any path and away from your current directory or the RTEMS source code. The storage does not need to be fast like an SSD.

**--log**

The log file.

**--config-report**

Print a configuration report and exit.

**--warnings-report**

Create a warnings report once all builds have finished.

**--stop-on-error**

Stop the build on an error. The default is to build all the builds for a profile.

**--no-clean**

Do not remove the build once finished. This option lets you inspect the built output. The amount of output can be large and disks can fill with this option.

**--profiles**

Build the comma separated list of profiles. The default is tier-1.

**--arch**

A comma separated list of architectures to build using the selected build.

**--bsp**

A comma separated list of BSPs to build where a BSP is of the format arch/bsp using the selected build.

**--build**

The build to be used. The default is all. See --config-report for a list of valid builds.

**--jobs**

The jobs options where the format is build-jobs/make-jobs. The default is 1/num-cores where num-cores is the operating system reported number of cores.



**--dry-run**

Do not do the actual builds just show what would be built.

## 10.4.5.1 Examples

The following is a *tier-1* profile build of *all* on a machine where all the source and tools are located on fast SSD disks and the build happens on a spinning disk mounted under *build*. The build uses a development source tree that is bootstrapped and ready to build. The source can have local patches that need to be regression tested:

```

1 $ /opt/rtems/5/bin/rtems-bsp-builder --build-path=/build/rtems \
2     --rtems-tools=/opt/work/rtems/5 \
3     --rtems=/opt/work/chris/rtems/kernel/rtems.git \
4     --profiles=tier-1 \
5     --jobs=5/10
6 RTEMS Tools Project - RTEMS Kernel BSP Builder, 5.not_released
7 Profile(s): tier-1
8 Cleaning: bsp-builds
9 [ 1/655] arm/altcycv_devkit (debug)          Start
10 [ 1/655] arm/altcycv_devkit (debug)        Creating: bsp-builds/arm/
11 ↪ altcycv_devkit.debug
12 [ 2/655] arm/altcycv_devkit (no-posix)     Start
13 [ 2/655] arm/altcycv_devkit (no-posix)     Creating: bsp-builds/arm/
14 ↪ altcycv_devkit.no-posix
15 [ 3/655] arm/altcycv_devkit (posix)        Start
16 [ 1/655] arm/altcycv_devkit (debug)        Configuring
17 [ 3/655] arm/altcycv_devkit (posix)        Creating: bsp-builds/arm/
18 ↪ altcycv_devkit.posix
19 [ 2/655] arm/altcycv_devkit (no-posix)     Configuring
20 [ 4/655] arm/altcycv_devkit (posix-debug)  Start
21 [ 1/655] arm/altcycv_devkit (debug)        Building
22 [ 3/655] arm/altcycv_devkit (posix)        Configuring
23 [ 4/655] arm/altcycv_devkit (posix-debug)  Creating: bsp-builds/arm/
24 ↪ altcycv_devkit.posix-debug
25 [ 2/655] arm/altcycv_devkit (no-posix)     Building
26 [ 5/655] arm/altcycv_devkit (posix-profiling) Start
27 [ 4/655] arm/altcycv_devkit (posix-debug)  Configuring
28 [ 3/655] arm/altcycv_devkit (posix)        Building
29 . . . .
30 [654/655] sparc/ngmp (posix-profiling)     PASS
31 [654/655] sparc/ngmp (posix-profiling)     Warnings:0 exes:0 objs:0
32 ↪ libs:0
33 [654/655] sparc/ngmp (posix-profiling)     Finished (duration:0:01:49.
34 ↪ 002189)
35 [654/655] sparc/ngmp (posix-profiling)     Status: Pass: 655 Fail:
36 ↪ 0 (configure:0 build:0)
37 [655/655] sparc/ngmp (profiling)          PASS
38 [655/655] sparc/ngmp (profiling)          Warnings:0 exes:0 objs:0
39 ↪ libs:0
40 [655/655] sparc/ngmp (profiling)          Finished (duration:0:01:260.
41 ↪ 002098)
42 [655/655] sparc/ngmp (profiling)          Status: Pass: 655 Fail:
43 ↪ 0 (configure:0 build:0)
44 [651/655] sparc/ngmp (no-posix)           Cleaning: bsp-builds/sparc/
45 ↪ ngmp.no-posix
46 [652/655] sparc/ngmp (posix)             Cleaning: bsp-builds/sparc/
47 ↪ ngmp.posix

```

```

36 [653/655] sparc/ngmp (posix-debug)           Cleaning: bsp-builds/sparc/
↪ngmp.posix-debug
37 [654/655] sparc/ngmp (posix-profiling)      Cleaning: bsp-builds/sparc/
↪ngmp.posix-profiling
38 [655/655] sparc/ngmp (profiling)           Cleaning: bsp-builds/sparc/
↪ngmp.profiling
39 Total: Warnings:31689  exes:6291  objs:793839  libs:37897
40 Failures:
41   No failure(s)
42 Average BSP Build Time: 0:00:18.165000
43 Total Time 3:41:48.075006
44 Passes: 655   Failures: 0

```

To build a couple of BSPs you are interested in with tests:

```

1 $ /opt/rtems/5/bin/rtems-bsp-builder --build-path=/build/rtems \
2     --rtems-tools=/opt/work/rtems/5 \
3     --rtems=/opt/work/chris/rtems/kernel/rtems.git \
4     ----log=lpc-log \
5     --bsp=arm/lpc2362,arm/lpc23xx_tli800 \
6     --build=tests \
7     --jobs=5/12
8 RTEMS Tools Project - RTEMS Kernel BSP Builder, 5.not_released
9 BSPS(s): arm/lpc2362, arm/lpc23xx_tli800
10 Cleaning: bsp-builds
11 [1/2] arm/lpc2362 (tests)           Start
12 [1/2] arm/lpc2362 (tests)           Creating: bsp-builds/arm/lpc2362.tests
13 [2/2] arm/lpc23xx_tli800 (tests)    Start
14 [2/2] arm/lpc23xx_tli800 (tests)    Creating: bsp-builds/arm/lpc23xx_tli800.tests
15 [1/2] arm/lpc2362 (tests)           Configuring
16 [2/2] arm/lpc23xx_tli800 (tests)    Configuring
17 [1/2] arm/lpc2362 (tests)           Building
18 [2/2] arm/lpc23xx_tli800 (tests)    Building
19 [1/2] arm/lpc2362 (tests)           FAIL
20 [1/2] arm/lpc2362 (tests)           Warnings:74  exes:58  objs:1645  libs:74
21 [1/2] arm/lpc2362 (tests)           Finished (duration:0:01:31.708252)
22 [1/2] arm/lpc2362 (tests)           Status: Pass:  0  Fail:  2 (configure:0 build:2)
23 [2/2] arm/lpc23xx_tli800 (tests)    FAIL
24 [2/2] arm/lpc23xx_tli800 (tests)    Warnings:74  exes:51  objs:1632  libs:74
25 [2/2] arm/lpc23xx_tli800 (tests)    Finished (duration:0:01:31.747582)
26 [2/2] arm/lpc23xx_tli800 (tests)    Status: Pass:  0  Fail:  2 (configure:0 build:2)
27 [1/2] arm/lpc2362 (tests)           Cleaning: bsp-builds/arm/lpc2362.tests
28 [2/2] arm/lpc23xx_tli800 (tests)    Cleaning: bsp-builds/arm/lpc23xx_tli800.tests
29 Total: Warnings:74  exes:109  objs:3277  libs:148
30 Failures:
31   1 tests arm/lpc2362 build:
32     configure: /opt/work/chris/rtems/kernel/rtems.git/configure --target\
33     =arm-rtems5 --enable-rtemsbsp=lpc2362 --prefix=/opt/rtems/5\
34     --enable-tests
35     error: ld/collect2:0 error: math.exe section '.rodata' will not fit
36     in region 'ROM_INT'; region 'ROM_INT' overflowed by 7284 bytes
37
38   2 tests arm/lpc23xx_tli800 build:
39     configure: /opt/work/chris/rtems/kernel/rtems.git/configure --target\
40     =arm-rtems5 --enable-rtemsbsp=lpc23xx_tli800\
41     --prefix=/opt/rtems/5 --enable-tests
42     error: ld/collect2:0 error: math.exe section '.text' will not fit in

```

```
43         region 'ROM_INT'; region 'ROM_INT' overflowed by 13972 bytes
44
45 Average BSP Build Time: 0:00:46.658257
46 Total Time 0:01:33.316514
47 Passes: 0   Failures: 2
```

The summary report printed shows both BSP builds failed with the error detail shown. In this case both are linker related errors where the test do not fit into the target's available resources.

## 10.5 RTEMS Tester and Run Commands

The RTEMS Tester is a test tool that provides a command line interface to run test executable on supported targets. The tool provides back-end support for common simulators, debuggers and boot loaders. Board support package (BSP) configurations for RTEMS are provided and can be used to run all the tests in the RTEMS test suite. The tool and its framework is not specific to RTEMS and can be configured to run any suitable application.

RTEMS is an embedded operating system and is cross-compiled on a range of host machines. The executables run on the target hardware and this can vary widely from open source simulators, commercial simulators, debuggers with simulators, debuggers with hardware specific pods and devices to target boot loaders. Testing RTEMS requires the cross-compiled test executable is transferred to the target hardware, executed and the output captured and returned to the test host where it is analyzed to determine the test result.

Running the RTEMS tests on your target is very important. It provides you with a traceable record showing that your RTEMS version and its tools are working at the level the RTEMS development team expect when releasing RTEMS. Being able to easily run the tests and verify the results is critical in maintaining a high standard.

### 10.5.1 Available BSP testers

You can list the available BSP testers with:

```
1 $ rtems-test --list-bsps
2 arm920
3 beagleboardxm
4 beagleboneblack
5 jmr3904-run
6 jmr3904
7 mcf5235
8 pc
9 psim-run
10 psim
11 realview_pbx_a9_qemu
12 sis-run
13 sis
14 xilinx_zynq_a9_qemu
15 xilinx_zynq_a9_qemu_smp
16 xilinx_zynq_zc706
17 xilinx_zynq_zc706_qemu
18 xilinx_zynq_zedboard
```

---

**Note:** The list is growing all the time and if your BSP is not supported we encourage you to add it and submit the configuration back to the project.

---

Some of the BSPs may appear more than once in the list. These are aliased BSP configurations that may use a different back-end. An example is the erc32 BSP. There is the erc32 tester which uses the GDB back-end and the erc32-run tester which uses the run command for erc32. We will show how to use **rtems-test** command with the erc32 BSP because it is easy to build an use.

## 10.5.2 Building RTEMS Tests

Build the RTEMS Kernel (See *Chapter 7 Section 4 - RTEMS Kernel*) by cloning the repository, running the bootstrap procedure, building and finally installing the kernel. Be sure to enable tests by using `--enable-tests` option with `configure` after running `bootstrap`.

```
1 $ ../../rtems.git/configure --target=sparc-rtems5 \
2     --enable-tests --enable-rtemsbsp=erc32
3 $ make
```

Add the `-j` option to the `make` command with the number of cores to run a parallel build.

Building all the tests takes time and it uses more disk so be patient. When finished all the tests will have been built. Some BSPs may require a post-build process to be run on the RTEMS ELF executable to create an image suitable for execution. This can be built into the configuration script and the tester will perform a pre-test command to convert the executable to a suitable format for your target.

Before running all the tests it is a good idea to run the hello test. The hello test is an RTEMS version of the classic “Hello World” example and running it shows you have a working tool chain and build of RTEMS ready to run the tests. Using the run with the ERC32 BSP the command is:

```
1 $ sparc-rtems5-run sparc-rtems5/c/erc32/testsuites/samples/hello/hello.exe
2
3 *** BEGIN OF TEST HELLO WORLD ***
4 Hello World
5 *** END OF TEST HELLO WORLD ***
```

The `run` command is the GDB simulator without the GDB part.

Running the example using GDB:

```
1 $ sparc-rtems5-gdb sparc-rtems5/c/erc32/testsuites/samples/hello/hello.exe
2 GNU gdb (GDB) 7.12
3 Copyright (C) 2016 Free Software Foundation, Inc.
4 License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
5 This is free software: you are free to change and redistribute it.
6 There is NO WARRANTY, to the extent permitted by law. Type "show copying"
7 and "show warranty" for details.
8 This GDB was configured as "--host=x86_64-linux-gnu --target=sparc-rtems5".
9 Type "show configuration" for configuration details.
10 For bug reporting instructions, please see:
11 <http://www.gnu.org/software/gdb/bugs/>.
12 Find the GDB manual and other documentation resources online at:
13 <http://www.gnu.org/software/gdb/documentation/>.
14 For help, type "help".
15 Type "apropos word" to search for commands related to "word"...
16 Reading symbols from
17 sparc-rtems5/c/erc32/testsuites/samples/hello/hello.exe...done.
18 (gdb) target sim
19 Connected to the simulator.
20 (gdb) load
21 (gdb) r
22 Starting program: sparc-rtems5/c/erc32/testsuites/samples/hello/hello.exe
23
24
25 *** BEGIN OF TEST HELLO WORLD ***
```

```

26 Hello World
27 *** END OF TEST HELLO WORLD ***
28 [Inferior 1 (process 42000) exited normally]
29 (gdb) q

```

The command `r` is used to debug set break points before issuing the GDB run command.

There are currently close to 500 separate tests and you can run them all with a single RTEMS Tester command.

### 10.5.3 Running the Tests

The `rtems-test` command line accepts a range of options. These are discussed later in the manual. Any command line argument without a `-` prefix is a test executable. You can pass more than one executable on the command line. If the executable is a path to a directory the directories under that path are searched for any file with a `.exe` extension. This is the default extension for RTEMS executables built within RTEMS.

To run the erc32 tests enter the following command from the top of the erc32 BSP build tree:

```

1 $ ~/development/rtems/test/rtems-tools.git/tester/rtems-test \
2   --log=log_erc32_run \
3   --rtems-bsp=erc32-run \
4   --rtems-tools=$HOME/development/rtems/5 \
5     sparc-rtems5/c/erc32/testsuites/samples
6 RTEMS Testing - Tester, 5.not_released
7 [ 1/13] p:0 f:0 u:0 e:0 I:0 B:0 t:0 i:0 | sparc/erc32: base_sp.exe
8 [ 2/13] p:0 f:0 u:0 e:0 I:0 B:0 t:0 i:0 | sparc/erc32: capture.exe
9 [ 3/13] p:0 f:0 u:0 e:0 I:0 B:0 t:0 i:0 | sparc/erc32: cdtest.exe
10 [ 4/13] p:0 f:0 u:0 e:0 I:0 B:0 t:0 i:0 | sparc/erc32: fileio.exe
11 [ 5/13] p:2 f:0 u:0 e:0 I:0 B:0 t:0 i:0 | sparc/erc32: hello.exe
12 [ 6/13] p:2 f:0 u:0 e:0 I:0 B:0 t:0 i:0 | sparc/erc32: cxx_iostream.exe
13 [ 8/13] p:2 f:0 u:0 e:0 I:0 B:0 t:2 i:0 | sparc/erc32: minimum.exe
14 [ 7/13] p:2 f:0 u:0 e:0 I:0 B:0 t:2 i:0 | sparc/erc32: loopback.exe
15 [ 9/13] p:3 f:0 u:0 e:0 I:0 B:0 t:3 i:0 | sparc/erc32: nsecs.exe
16 [10/13] p:3 f:0 u:0 e:0 I:0 B:0 t:3 i:0 | sparc/erc32: paranoia.exe
17 [11/13] p:4 f:0 u:0 e:0 I:0 B:0 t:3 i:0 | sparc/erc32: pppd.exe
18 [12/13] p:6 f:0 u:0 e:0 I:0 B:0 t:3 i:0 | sparc/erc32: ticker.exe
19 [13/13] p:6 f:0 u:0 e:0 I:0 B:0 t:3 i:0 | sparc/erc32: unlimited.exe
20 Passed:          7
21 Failed:          0
22 User Input:      0
23 Expected Fail:  0
24 Indeterminate:  0
25 Benchmark:      0
26 Timeout:        5
27 Invalid:        1
28 Total:          13
29 Average test time: 0:00:27.963000
30 Testing time    : 0:06:03.519012

```

- The RTEMS Tester's test command. In this example we are using an absolute path.
- The `--log` option sends the output to a log file. By default only failed tests log the complete output.

- Select the erc32 BSP and use GDB.
- Path to the RTEMS tools so GDB can be found.
- Path to the erc32 BSP built with all tests to run. If you add subdirectories to the path specific tests can be run.
- The output has been shortened so it fits nicely here.
- The test results shows passes, fails, timeouts, and invalid results. In this run 13 tests passed and 5 tests timed out and 1 is invalid. The timeouts are probably due to the tests not having enough execute time to complete. The default timeout is 180 seconds and some of the interrupt tests need longer. The amount of time depends on the performance of your host CPU running the simulations.
- The output shows the average time per test and the total time taken to run all the tests.
- If the path to the testsuites was put to `sparc-rtems5/c/erc32/testsuites` instead of `sparc-rtems5/c/erc32/testsuites/samples` then all the executables would have been tested and not just those in samples.

This BSP requires the `--rtems-tools` option because the SPARC GDB is the `sparc-rtems4.11-gdb` command that is part of the RTEMS tools. Not every BSP will require this option so you will need to check the specifics of the BSP configuration to determine if it is needed.

The output you see is each test starting to run. The `rtems-test` command by default runs multiple tests in parallel so you will see a number start quickly and then new tests start as others finish. The output shown here is from an 8 core processor so the first 8 are started in parallel and the status shows the order in which they actually started, which is not 1 to 8.

The test start line shows the current status of the tests. The status reported is when the test starts and not the result of that test. A fail, timeout or invalid count changing means a test running before this test started failed, not the starting test. The status here has 7 tests passed, no failures, 5 timeouts and 1 invalid test.

```
1 [ 5/13] p:2 f:0 u:0 e:0 I:0 B:0 t:0 i:0 | sparc/erc32: hello.exe
```

- [ 5/13] indicates the test number, in this case test 5 of 13 tests.
- p is the passed test count (2 in this case)
- f is the failed test count (0 in this case)
- u is the count for test marked as “user-input” as they expect input from user
- e is the expected-fail count (tests that are expected to fail)
- I is the count for tests the results of which are indeterminate
- B is the count for benchmarked tests
- t is the timeout test count
- i is the invalid test count.
- `sparc/erc32` is the architecture and BSP names.
- `hello.exe` is the executable name.

The test log records all the tests and results. The reporting mode by default only provides the output history if a test fails, times out, or is invalid. The time taken by each test is also recorded.

The tests must complete in a specified time or the test is marked as timed out. The default timeout is 3 minutes and can be globally changed using the `--timeout` command line option. The time required to complete a test can vary. When simulators are run in parallel the time taken depends on the specifics of the host machine being used. A test per core is the most stable method even though more tests can be run than available cores. If your machine needs longer or you are using a VM you may need to lengthen the timeout.

#### 10.5.4 Test Status

Tests can be marked with one of the following:

- Pass
- Fail
- User-input
- Expected-fail
- Indeterminate
- Benchmark
- Timeout
- Invalid

The RTEMS console or `stdout` output from the test is needed to determine the result of the test.

##### 10.5.4.1 Pass

A test passes if the start and end markers are seen in the test output. The start marker is `***` and the end mark is `*** END OF TEST`. All tests in the RTEMS test suite have these markers.

##### 10.5.4.2 Fail

A test fails if the start marker is seen and there is no end marker.

##### 10.5.4.3 User-input

A test marked as “user-input” as it expects input from user

##### 10.5.4.4 Expected-fail

A test that is expected to fail.

##### 10.5.4.5 Indeterminate

A test the results of which are indeterminate.



#### 10.5.4.6 Benchmark

A benchmarked test.

#### 10.5.4.7 Timeout

If the test does not complete within the timeout setting the test is marked as having timed out.

#### 10.5.4.8 Invalid

If no start marker is seen the test is marked as invalid. If you are testing on real target hardware things can sometimes go wrong and the target may not initialize or respond to the debugger in an expected way.

### 10.5.5 Reporting

The report written to the log has the following modes:

- All (all)
- Failures (failures)
- None (none)

The mode is controlled using the command line option `--report-mode` using the values listed above.

#### 10.5.5.1 All

The output of all tests is written to the log.

#### 10.5.5.2 Failures

The output of the all tests that do not pass is written to the log.

#### 10.5.5.3 None

No output is written to the log.

The output is tagged so you can determine where it comes from. The following is the complete output for the In Memory File System test `imfs_fslink.exe` running on a Coldfire MCF5235 using GDB and a BDM pod:

```
1 [ 11/472] p:9  f:0  t:0  i:1  | m68k/mcf5235: imfs_fslink.exe
2 > gdb: .../bin/m68k-rtems4.11-gdb -i=mi --nx --quiet .../imfs_fslink.exe
3 > Reading symbols from .../fstests/imfs_fslink/imfs_fslink.exe...
4 > done.
5 > target remote | m68k-bdm-gdbserver pipe 003-005
6 > Remote debugging using | m68k-bdm-gdbserver pipe 003-005
7 > m68k-bdm: debug module version 0
8 > m68k-bdm: detected MCF5235
```

```

9 > m68k-bdm: architecture CF5235 connected to 003-005
10 > m68k-bdm: Coldfire debug module version is 0 (5206(e)/5235/5272/5282)
11 > Process 003-005 created; pid = 0
12 > 0x00006200 in ?? ()
13 > thb *0xffe254c0
14 > Hardware assisted breakpoint 1 at 0xffe254c0
15 > continue
16 > Continuing.
17 ]
18 ]
19 ] External Reset
20 ]
21 ] ColdFire MCF5235 on the BCC
22 ] Firmware v3b.1a.1a (Built on Jul 21 2004 17:31:28)
23 ] Copyright 1995-2004 Freescale Semiconductor, Inc. All Rights Reserved.
24 ]
25 ] Enter 'help' for help.
26 ]
27 > Temporary breakpoint
28 > 1, 0xffe254c0 in ?? ()
29 > load
30 > Loading section .text, size 0x147e0 lma 0x40000
31 > Loading section .data, size 0x5d0 lma 0x547e0
32 > Start address 0x40414, load size 85424
33 > Transfer rate: 10 KB/sec, 1898 bytes/write.
34 > b bsp_reset
35 > Breakpoint 2 at 0x41274: file ../shared/bspreset_loop.c, line 14.
36 > continue
37 > Continuing.
38 ] dBUG>
39 ]
40 ] *** FILE SYSTEM TEST ( IMFS ) ***
41 ] Initializing filesystem IMFS
42 ]
43 ]
44 ] *** LINK TEST ***
45 ] link creates hardlinks
46 ] test if the stat is the same
47 ] chmod and chown
48 ] unlink then stat the file
49 ] *** END OF LINK TEST ***
50 ]
51 ]
52 ] Shutting down filesystem IMFS
53 ] *** END OF FILE SYSTEM TEST ( IMFS ) ***
54 > Breakpoint
55 > 2, bsp_reset () at ../m68k/mcf5235/../../shared/bspreset_loop.c:14
56 > 14 {
57 Result: passed      Time: 0:00:10.045447

```

- GDB command line (Note: paths with ‘...’ have been shortened)
- Lines starting with > are from GDB’s console.
- Line starting with ] are from the target’s console.
- The result with the test time.

## 10.5.6 Running Tests in Parallel

The RTEMS Tester supports parallel execution of tests by default. This only makes sense if the test back-end can run in parallel without resulting in resource contention. Simulators are an example of back-ends that can run in parallel. A hardware debug tool like a BDM or JTAG pod can manage only a single test at once so the tests need to be run one at a time.

The test framework manages the test jobs and orders the output in the report log in test order. Output is held for completed tests until the next test to be reported has finished.

## 10.5.7 Command Line Help

The `rtems-test` command line accepts a range of options. You can review the available option by the `--help` option:

```
1 RTEMS Tools Project (c) 2012-2014 Chris Johns
2 Options and arguments:
3 --always-clean           : Always clean the build tree, even with an error
4 --debug-trace           : Debug trace based on specific flags
5 --dry-run               : Do everything but actually run the build
6 --force                 : Force the build to proceed
7 --jobs=[0..n,none,half,full] : Run with specified number of jobs, default: num CPUs.
8 --keep-going            : Do not stop on an error.
9 --list-bsps             : List the supported BSPs
10 --log file              : Log file where all build output is written to
11 --macros file[,file]   : Macro format files to load after the defaults
12 --no-clean              : Do not clean up the build tree
13 --quiet                 : Quiet output (not used)
14 --report-mode           : Reporting modes, failures (default),all,none
15 --rtems-bsp             : The RTEMS BSP to run the test on
16 --rtems-tools           : The path to the RTEMS tools
17 --target                : Set the target triplet
18 --timeout                : Set the test timeout in seconds (default 180 seconds)
19 --trace                 : Trace the execution
20 --warn-all              : Generate warnings
```



# SUPPORT

RTEMS offers a variety of support options.

This chapter covers all options available to both users and developers. If you believe this is a bug report please submit it to the bug tracker otherwise the developers mailing list is the default location to send the report.



## RTEMS PROJECT SUPPORT

The following support channels are provided by the RTEMS Project and provide direct access to the RTEMS community.

## 12.1 Bug Tracker

The bug tracker can be found at the [Bugs Database](#).

See the [Submission Guidelines](#) for details on submitting a ticket.

Be sure to do a cursory search for any tickets that may be relevant to your problem.

If you are unsure about your issue status submit a ticket and we will help you sort it out.



## 12.2 Documentation

The latest user documentation can always be found at the [Documentation Site](#).

## 12.3 Mailing Lists

We have several mailing lists for RTEMS users and developers.

- **Announce Mailing List**
  - Announcements for major and other project-related issues.
- **Bugs Mailing List**
  - Bugs email from [Bugs Database](#).
- **Developers Mailing List**
  - Developers list, this is for developers of RTEMS itself.
- **Build logs**
  - Results from the testing and building of RTEMS.
- **Users Mailing List**
  - Users of RTEMS.
- **Version Control Mailing List**
  - Commits to the RTEMS master repository.

## 12.4 IRC

RTEMS IRC is available on the Freenode network. See the [Freenode](#) web site for details on connecting, selecting a nickname, and general usage tips. If you are new to IRC it is recommended reading.

These are the current IRC channels.

### **#rtems**

This is a general channel for all things RTEMS. You can just hang out with other RTEMS users and developers to talk about RTEMS, using RTEMS or to make contact with other RTEMS users.

The #rtems channel is logged. You can find the logs at <http://www.rtems.org/irclogs/>. You can search the logs using Google by adding:

**site:rtems.org inurl:irclogs**

to your search terms.

## 12.5 Developers

Developers can find help and support on the mailing lists, see *Chapter 12 Section 3 - Mailing Lists*.

Technical documents including design, [Google Summer of Code](#), [ESA SOCIS](#) can be found on the [Developer Site](#).

## COMMERCIAL SUPPORT SERVICES

The wider RTEMS community has developers and organizations who can provide commercial support services. These services range from training, implementing new features in RTEMS, deployment of RTEMS< helping establish a new project environment for a team, to application and system design.

The RTEMS Project does not endorse or promote any provider of these services and we recommend you use a search engine to locate a suitable provider. If you are unsure please contact a provider and see what is available.

If you develop a new feature or you have someone do this for you we recommend you have the work submitted to the project and merged. Once accepted into the project the work will be maintained as part of the development process within the project and this is a benefit for.



# GLOSSARY

## **Architecture**

Family or class of processor based around a common instruction set. RTEMS architectures follow the GCC architecture model as RTEMS needs an GCC architecture compiler for each support RTEMS architecture.

## **Binutils**

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

## **BSP**

Board Support Package is a specific configuration RTEMS can be built for. An RTEMS install process installs specific library and headers files for a single BSP. A BSP optimises RTEMS to a specific target hardware.

## **Buildbot**

A continuous integration build server.

## **Crosscompiler**

A compiler built to run on a Host that generate code for another architecture.

## **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

## **Host**

The computer and operating system that hosts the RTEMS development tools such as the compiler, linker and 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 simplified 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.

**Target**

A target is the hardware or simulator a BSP built executable runs on.

**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/>



**INDEX**

# INDEX

- arch
  - command line option, 82
- bsp
  - command line option, 82
- build
  - command line option, 82
- build-path
  - command line option, 82
- config-report
  - command line option, 82
- dry-run
  - command line option, 82
- jobs
  - command line option, 82
- log
  - command line option, 82
- no-clean
  - command line option, 82
- prefix
  - command line option, 82
- profiles
  - command line option, 82
- rtems
  - command line option, 82
- rtems-tools
  - command line option, 82
- stop-on-error
  - command line option, 82
- warnings-report
  - command line option, 82
- ?
  - command line option, 82
- , -h
  - command line option, 72, 75
- C, -cc
  - command line option, 72
- E, -exec-prefix
  - command line option, 72
- F
  - command line option, 75
- I
  - command line option, 75
- S
  - command line option, 75
- S, -symc
  - command line option, 72
- V
  - command line option, 74
- V, -version
  - command line option, 72
- a
  - command line option, 74
- c, -cflags
  - command line option, 72
- e, -embed
  - command line option, 72
- h, -help
  - command line option, 82
- k, -keep
  - command line option, 72
- m, -map
  - command line option, 72
- o, -output
  - command line option, 72
- v
  - command line option, 74
- v, -verbose
  - command line option, 72
- w, -warn
  - command line option, 72
- Architecture, **105**
- Architectures, 61
- Binutils, **105**
- Board Support Package, 62
- Board Support Packages (BSPs), 65
- BSP, 62, **105**
- Buildbot, **105**
- command line option

- arch, 82
- bsp, 82
- build, 82
- build-path, 82
- config-report, 82
- dry-run, 82
- jobs, 82
- log, 82
- no-clean, 82
- prefix, 82
- profiles, 82
- rtems, 82
- rtems-tools, 82
- stop-on-error, 82
- warnings-report, 82
- , 82
- , -h, 72, 75
- C, -cc, 72
- E, -exec-prefix, 72
- F, 75
- I, 75
- S, 75
- S, -symc, 72
- V, 74
- V, -version, 72
- a, 74
- c, -cflags, 72
- e, -embed, 72
- h, -help, 82
- k, -keep, 72
- m, -map, 72
- o, -output, 72
- v, 74
- v, -verbose, 72
- w, -warn, 72
- Crosscompiler, **105**
- DLL, **105**
- Ecosystem, 9
- GCC, **105**
- GDB, **105**
- Git, 38
- Hardware, 59
- Host, **105**
- Host Computer, 15
- Installation, 25
- linker, 70
- Microsoft Windows Installation, 42
- MinGW, **105**
- MinGW64, **105**
- MSYS2, **105**
- POSIX, **106**
- prefix, **106**
- Prefixes, 27
- Quick Start, 12
- release, 31
- RSB, **106**
- RTEMS, **106**
- rtems-bsp-builder, 78
- rtems-exeinfo, 74
- rtems-ld, 70
- rtems-run, 86
- rtems-syms, 71
- rtems-test, 86
- tarball, 31
- Target, **106**
- Targets, 60
- Test Suite, **106**
- Testsuite, **106**
- Tools, 27, 70, 71, 74, 78, 86
- Waf, **106**