RTEMS User Manual

Release 4.11.2-rc5

CONTENTS

Part I
TABLE OF CONTENTS

CHAPTER

ONE

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

1.1 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 catastophic 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.

1.2 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.

CHAPTER

TWO

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 accessable, useable and predicable.

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.

2.1 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, compontents 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.

2.2 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.

2.3 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.

CHAPTER

THREE

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 encoutnered.

The detailed procedure for installing RTEMS can be found in *Chapter 5 - Installation* (page 25).

The development host computer needs to be set up for this quick start procedure to complete successfully. *Chapter 4 - Host Computer* (page 13) details what is needed for the supported host operating systems. If Windows is being used it is recommended following the procedure in *Chapter 4 Section 4 - Microsoft Windows* (page 17) first.

Their 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 5 Section 1 - Prefixes* (page 26) and *Chapter 5 Section 1.1 - Project Sandboxing* (page 26) 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 5 Section 3.1 - POSIX and OS X Host Tools Chain* (page 36)):

```
$ cd
$ mkdir -p development/rtems
$ cd development/rtems
$ git clone git://git.rtems.org/rtems-source-builder.git rsb
  ...
$ cd rsb
$ ./source-builder/sb-check
  ...
$ cd rtems
$ ../source-builder/sb-set-builder \
    --prefix=/usr/home/chris/development/rtems/4.12 4.12/rtems-sparc
  ...
```

Build the RTEMS Kernel (*Chapter 5 Section 3.2.2 - Building the Kernel* (page 43)) by cloning the repository, running the `bootstrap` procecure, building and finally installing the kernel:

```
$ export PATH=$HOME/development/rtems/4.12/bin:$PATH
$ cd
$ cd development/rtems
$ mkdir kernel
$ cd kernel
```

```
$ git clone git://git.rtems.org/rtems.git rtems
  ...
$ cd rtems
$ ./bootstrap -c && ./bootstrap -p && \
            $HOME/development/rtems/rsb/source-builder/sb-bootstrap
  ...
$ cd ..
$ mkdir erc32
$ cd erc32
$ $HOME/development/rtems/kernel/rtems/configure --prefix=$HOME/development/rtems/4.12 \
                  --target=sparc-rtems4.12 --enable-rtemsbsp=erc32 --enable-posix
  ...
$ make -j 8
  ...
$ make install
```

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

CHAPTER

FOUR

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 distrbution. 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.

4.1 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.

4.2 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.

4.2.1 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 5 Section 1 - Prefixes (page 26) 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 5 Section 1.1 - Project Sandboxing* (page 26) explains this in more detail.

4.3 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. Xocde 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 5 Section 1 - Prefixes (page 26) details using Prefixes to manage the installation.

4.4 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*.

4.4.1 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 5 Section 2 - Releases* (page 29).

4.4.2 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.

4.4.3 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.

4.4.4 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 suitables 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.

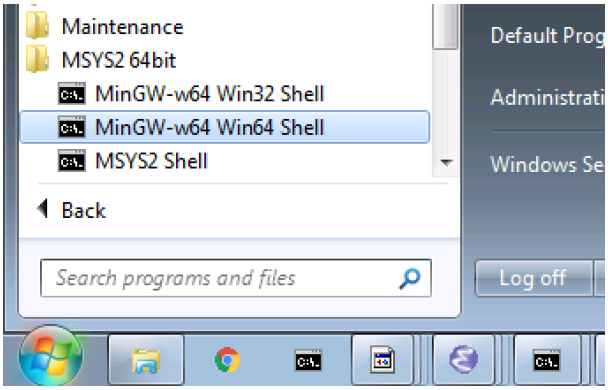
4.4.5 Installing MSYS2

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:

```
~
$ pacman -S python mingw-w64-x86_64-python2 mingw-w64-x86_64-gcc \
bison cvs diffutils git make patch tar texinfo unzip
resolving dependencies...
looking for conflicting packages...

Packages (74) db-5.3.28-2  expat-2.1.0-2  gdbm-1.11-3  heimdal-1.5.3-8
              libgdbm-1.11-3  mingw-w64-x86_64-binutils-2.26-21
              mingw-w64-x86_64-bzip2-1.0.6-5
              mingw-w64-x86_64-ca-certificates-20150426-2
              mingw-w64-x86_64-crt-git-5.0.0.4627.03684c4-1
              mingw-w64-x86_64-expat-2.1.0-6  mingw-w64-x86_64-gcc-libs-5.3.0-2
              mingw-w64-x86_64-gdbm-1.11-3  mingw-w64-x86_64-gettext-0.19.6-2
```

```
              mingw-w64-x86_64-gmp-6.1.0-1
              mingw-w64-x86_64-headers-git-5.0.0.4627.53be55d-1
              mingw-w64-x86_64-isl-0.15-1  mingw-w64-x86_64-libffi-3.2.1-3
              mingw-w64-x86_64-libiconv-1.14-5
              mingw-w64-x86_64-libsystre-1.0.1-2
              mingw-w64-x86_64-libtasn1-4.7-1
              mingw-w64-x86_64-libtre-git-r122.c2f5d13-4
              mingw-w64-x86_64-libwinpthread-git-5.0.0.4573.628fdbf-1
              mingw-w64-x86_64-mpc-1.0.3-2  mingw-w64-x86_64-mpfr-3.1.3.p0-2
              mingw-w64-x86_64-ncurses-6.0.20160220-2
              mingw-w64-x86_64-openssl-1.0.2.g-1
              mingw-w64-x86_64-p11-kit-0.23.1-3
              mingw-w64-x86_64-readline-6.3.008-1  mingw-w64-x86_64-tcl-8.6.5-1
              mingw-w64-x86_64-termcap-1.3.1-2  mingw-w64-x86_64-tk-8.6.5-1
              mingw-w64-x86_64-windows-default-manifest-6.4-2
              mingw-w64-x86_64-winpthreads-git-5.0.0.4573.628fdbf-1
              mingw-w64-x86_64-zlib-1.2.8-9  openssh-7.1p2-1  perl-5.22.0-2
              perl-Authen-SASL-2.16-2  perl-Convert-BinHex-1.123-2
              perl-Encode-Locale-1.04-1  perl-Error-0.17024-1
              perl-File-Listing-6.04-2  perl-HTML-Parser-3.71-3
              perl-HTML-Tagset-3.20-2  perl-HTTP-Cookies-6.01-2
              perl-HTTP-Daemon-6.01-2  perl-HTTP-Date-6.02-2
              perl-HTTP-Message-6.06-2  perl-HTTP-Negotiate-6.01-2
              perl-IO-Socket-SSL-2.016-1  perl-IO-stringy-2.111-1
              perl-LWP-MediaTypes-6.02-2  perl-MIME-tools-5.506-1
              perl-MailTools-2.14-1  perl-Net-HTTP-6.09-1
              perl-Net-SMTP-SSL-1.02-1  perl-Net-SSLeay-1.70-1
              perl-TermReadKey-2.33-1  perl-TimeDate-2.30-2  perl-URI-1.68-1
              perl-WWW-RobotRules-6.02-2  perl-libwww-6.13-1  vim-7.4.1468-1
              bison-3.0.4-1  cvs-1.11.23-2  diffutils-3.3-3  git-2.7.2-1
              make-4.1-4  mingw-w64-x86_64-gcc-5.3.0-2
              mingw-w64-x86_64-python2-2.7.11-4  patch-2.7.5-1  python-3.4.3-3
              tar-1.28-3  texinfo-6.0-1  unzip-6.0-2

Total Download Size:   114.10 MiB
Total Installed Size:  689.61 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
 mingw-w64-x86_64-gm...   477.1 KiB   681K/s 00:01 [######################] 100%
 mingw-w64-x86_64-li...    24.2 KiB   755K/s 00:00 [######################] 100%
 mingw-w64-x86_64-gc...   541.9 KiB   705K/s 00:01 [######################] 100%
 mingw-w64-x86_64-ex...   106.7 KiB   702K/s 00:00 [######################] 100%
 mingw-w64-x86_64-bz...    77.9 KiB   666K/s 00:00 [######################] 100%
 mingw-w64-x86_64-li...   600.2 KiB   703K/s 00:01 [######################] 100%
 mingw-w64-x86_64-ge...     3.0 MiB   700K/s 00:04 [######################] 100%
 mingw-w64-x86_64-gd...   151.8 KiB   483K/s 00:00 [######################] 100%
 mingw-w64-x86_64-li...    34.5 KiB   705K/s 00:00 [######################] 100%
 mingw-w64-x86_64-li...    69.2 KiB   713K/s 00:00 [######################] 100%
 mingw-w64-x86_64-li...     9.3 KiB   778K/s 00:00 [######################] 100%
 mingw-w64-x86_64-nc...  1800.5 KiB   701K/s 00:03 [######################] 100%
 mingw-w64-x86_64-li...   171.4 KiB   708K/s 00:00 [######################] 100%
 mingw-w64-x86_64-p1...   193.5 KiB   709K/s 00:00 [######################] 100%
 mingw-w64-x86_64-ca...   382.1 KiB   705K/s 00:01 [######################] 100%
 mingw-w64-x86_64-zl...   148.6 KiB   704K/s 00:00 [######################] 100%
 mingw-w64-x86_64-op...     3.3 MiB   624K/s 00:05 [######################] 100%
 mingw-w64-x86_64-te...    12.6 KiB  76.7K/s 00:00 [######################] 100%
```

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

```
(74/74) checking package integrity                  [######################] 100%
(74/74) loading package files                       [######################] 100%
(74/74) checking for file conflicts                 [######################] 100%
(74/74) checking available disk space               [######################] 100%
:: Processing package changes...
( 1/74) installing python                           [######################] 100%
( 2/74) installing mingw-w64-x86_64-gmp             [######################] 100%
( 3/74) installing mingw-w64-x86_64-libwinpthr...   [######################] 100%
( 4/74) installing mingw-w64-x86_64-gcc-libs        [######################] 100%
( 5/74) installing mingw-w64-x86_64-expat           [######################] 100%
( 6/74) installing mingw-w64-x86_64-bzip2           [######################] 100%
( 7/74) installing mingw-w64-x86_64-libiconv        [######################] 100%
( 8/74) installing mingw-w64-x86_64-gettext         [######################] 100%
( 9/74) installing mingw-w64-x86_64-gdbm            [######################] 100%
(10/74) installing mingw-w64-x86_64-libffi          [######################] 100%
(11/74) installing mingw-w64-x86_64-libtre-git      [######################] 100%
(12/74) installing mingw-w64-x86_64-libsystre       [######################] 100%
(13/74) installing mingw-w64-x86_64-ncurses         [######################] 100%
(14/74) installing mingw-w64-x86_64-libtasn1        [######################] 100%
(15/74) installing mingw-w64-x86_64-p11-kit         [######################] 100%
(16/74) installing mingw-w64-x86_64-ca-certifi...   [######################] 100%
(17/74) installing mingw-w64-x86_64-zlib            [######################] 100%
(18/74) installing mingw-w64-x86_64-openssl         [######################] 100%
(19/74) installing mingw-w64-x86_64-termcap         [######################] 100%
(20/74) installing mingw-w64-x86_64-readline        [######################] 100%
(21/74) installing mingw-w64-x86_64-tcl             [######################] 100%
(22/74) installing mingw-w64-x86_64-tk              [######################] 100%
(23/74) installing mingw-w64-x86_64-python2         [######################] 100%
(24/74) installing mingw-w64-x86_64-binutils        [######################] 100%
(25/74) installing mingw-w64-x86_64-headers-git     [######################] 100%
(26/74) installing mingw-w64-x86_64-crt-git         [######################] 100%
(27/74) installing mingw-w64-x86_64-isl             [######################] 100%
(28/74) installing mingw-w64-x86_64-mpfr            [######################] 100%
(29/74) installing mingw-w64-x86_64-mpc             [######################] 100%
(30/74) installing mingw-w64-x86_64-windows-de...   [######################] 100%
(31/74) installing mingw-w64-x86_64-winpthread...   [######################] 100%
(32/74) installing mingw-w64-x86_64-gcc             [######################] 100%
(33/74) installing bison                            [######################] 100%
(34/74) installing heimdal                          [######################] 100%
(35/74) installing cvs                              [######################] 100%
(36/74) installing diffutils                        [######################] 100%
(37/74) installing expat                            [######################] 100%
(38/74) installing vim                              [######################] 100%
(39/74) installing openssh                          [######################] 100%
(40/74) installing db                               [######################] 100%
(41/74) installing libgdbm                          [######################] 100%
(42/74) installing gdbm                             [######################] 100%
(43/74) installing perl                             [######################] 100%
(44/74) installing perl-Error                       [######################] 100%
(45/74) installing perl-Authen-SASL                 [######################] 100%
(46/74) installing perl-Encode-Locale               [######################] 100%
(47/74) installing perl-HTTP-Date                   [######################] 100%
(48/74) installing perl-File-Listing                [######################] 100%
(49/74) installing perl-HTML-Tagset                 [######################] 100%
(50/74) installing perl-HTML-Parser                 [######################] 100%
(51/74) installing perl-LWP-MediaTypes              [######################] 100%
(52/74) installing perl-URI                         [######################] 100%
```

```
(53/74) installing perl-HTTP-Message                [######################] 100%
(54/74) installing perl-HTTP-Cookies                [######################] 100%
(55/74) installing perl-HTTP-Daemon                 [######################] 100%
(56/74) installing perl-HTTP-Negotiate              [######################] 100%
(57/74) installing perl-Net-HTTP                    [######################] 100%
(58/74) installing perl-WWW-RobotRules              [######################] 100%
(59/74) installing perl-libwww                      [######################] 100%
Optional dependencies for perl-libwww
    perl-LWP-Protocol-HTTPS: for https:// url schemes
(60/74) installing perl-TimeDate                    [######################] 100%
(61/74) installing perl-MailTools                   [######################] 100%
(62/74) installing perl-IO-stringy                  [######################] 100%
(63/74) installing perl-Convert-BinHex              [######################] 100%
module test... pass.
(64/74) installing perl-MIME-tools                  [######################] 100%
(65/74) installing perl-Net-SSLeay                  [######################] 100%
(66/74) installing perl-IO-Socket-SSL               [######################] 100%
(67/74) installing perl-Net-SMTP-SSL                [######################] 100%
(68/74) installing perl-TermReadKey                 [######################] 100%
(69/74) installing git                              [######################] 100%
Optional dependencies for git
    python2: various helper scripts
    subversion: git svn
(70/74) installing make                             [######################] 100%
(71/74) installing patch                            [######################] 100%
Optional dependencies for patch
    ed: for patch -e functionality
(72/74) installing tar                              [######################] 100%
(73/74) installing texinfo                          [######################] 100%
(74/74) installing unzip                            [######################] 100%
```

CHAPTER

FIVE

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 commonlly 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 distrbution. 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.

5.1 Prefixes

You will see the term *prefix* referred to thoughout 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 standand **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 it's 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`.

5.1.1 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 5 Section 2 - Releases* (page 29) 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 varation 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.

5.2 Releases

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

Please read *Chapter 4 - Host Computer* (page 13) 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 4 Section 4.1 - Windows Path Length* (page 17) provides more detail about path lengths on Windows.

The location used to install the tools and kernel is called the *prefix*. *Chapter 5 Section 1 - Prefixes* (page 26) 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 4.12 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 sis"`. 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.

5.2.1 RTEMS Tools and Kernel

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

```
$ cd
$ mkdir -p development/rtems/releases
$ cd development/rtems/releases
```

If building on Windows:

```
$ cd /c
$ mkdir -p opt/rtems
$ 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:

```
$  wget  https://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/rtems-source-builder-4.11.0.
 ↪tar.xz
--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
Resolving ftp.rtems.org (ftp.rtems.org)... 140.211.10.151
Connecting to ftp.rtems.org (ftp.rtems.org)|140.211.10.151|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 967056 (944K) [application/x-xz]
Saving to: 'rtems-source-builder-4.11.0.tar.xz'

rtems-source-builder-4.1 100%[===================================>] 944.39K 206KB/s   in␣
 ↪5.5s

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:

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

On Windows you need to shorten the path (See *Chapter 4 Section 4.1 - Windows Path Length* (page 17)) after you have unpacked the tar file:

```
$ tar Jxf rtems-source-builder-4.11.0.tar.xz
$ mv rtems-source-builder-4.11.0 4.110
$ cd 4.11.0
```

Build a tool chain for the SPARC architecure. 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 4 Section 4.2 - Parallel Builds with Make* (page 17).

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

```
downloading: sources/autoconf-2.69.tar.gz - 1.8MB of 1.8MB (100%)
building: autoconf-2.69-x86_64-freebsd10.1-1
config: tools/rtems-automake-1.12.6-1.cfg
package: automake-1.12.6-x86_64-freebsd10.1-1
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
downloading: sources/automake-1.12.6.tar.gz - 2.0MB of 2.0MB (100%)
Creating source directory: patches
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
downloading: patches/automake-1.12.6-bugzilla.redhat.com-1239379.diff - 408.0 bytes of 408.
 ↪0 bytes (100%)
building: automake-1.12.6-x86_64-freebsd10.1-1
cleaning: autoconf-2.69-x86_64-freebsd10.1-1
cleaning: automake-1.12.6-x86_64-freebsd10.1-1
Build Set: Time 0:00:32.749337
Build Set: 4.11/rtems-autotools-base.bset
config: tools/rtems-autoconf-2.69-1.cfg
package: autoconf-2.69-x86_64-freebsd10.1-1
building: autoconf-2.69-x86_64-freebsd10.1-1
reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-freebsd10.1-1.txt
reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-freebsd10.1-1.xml
config: tools/rtems-automake-1.12.6-1.cfg
package: automake-1.12.6-x86_64-freebsd10.1-1
building: automake-1.12.6-x86_64-freebsd10.1-1
reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-freebsd10.1-1.txt
reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-freebsd10.1-1.xml
installing: autoconf-2.69-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
installing: automake-1.12.6-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
cleaning: autoconf-2.69-x86_64-freebsd10.1-1
cleaning: automake-1.12.6-x86_64-freebsd10.1-1
Build Set: Time 0:00:15.619219
Build Set: Time 0:00:48.371085
config: devel/expat-2.1.0-1.cfg
package: expat-2.1.0-x86_64-freebsd10.1-1
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
downloading: sources/expat-2.1.0.tar.gz - 549.4kB of 549.4kB (100%)
building: expat-2.1.0-x86_64-freebsd10.1-1
reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-freebsd10.1-1.txt
reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-freebsd10.1-1.xml
config: tools/rtems-binutils-2.26-1.cfg
package: sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1
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
downloading: sources/binutils-2.26.tar.bz2 - 24.4MB of 24.4MB (100%)
building: sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1
reporting: tools/rtems-binutils-2.26-1.cfg ->
sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1.txt
reporting: tools/rtems-binutils-2.26-1.cfg ->
sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1.xml
config: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg
package: sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1
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
downloading: sources/gcc-4.9.3.tar.bz2 - 85.8MB of 85.8MB (100%)
```

```
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
downloading: sources/newlib-2.2.0.20150423.tar.gz - 16.7MB of 16.7MB (100%)
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
downloading: sources/mpfr-3.0.1.tar.bz2 - 1.1MB of 1.1MB (100%)
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
downloading: sources/mpc-0.8.2.tar.gz - 535.5kB of 535.5kB (100%)
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
downloading: sources/gmp-5.0.5.tar.bz2 - 2.0MB of 2.0MB (100%)
building: sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1
reporting: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg ->
sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1.txt
reporting: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg ->
sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1.xml
config: tools/rtems-gdb-7.9-1.cfg
package: sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1
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
downloading: sources/gdb-7.9.tar.xz - 17.0MB of 17.0MB (100%)
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
downloading: patches/0001-sim-erc32-Disassembly-in-stand-alone-mode-did-not-wo.patch - 1.
↪9kB of 1.9kB (100%)
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
downloading: patches/0002-sim-erc32-Corrected-wrong-CPU-implementation-and-ver.patch - 827.
↪0 bytes of 827.0 bytes (100%)
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
downloading: patches/0003-sim-erc32-Perform-pseudo-init-if-binary-linked-to-no.patch - 2.
↪6kB of 2.6kB (100%)
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
downloading: patches/0004-sim-erc32-Use-fenv.h-for-host-FPU-access.patch - 4.9kB of 4.9kB␣
↪(100%)
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
downloading: patches/0005-sim-erc32-Remove-unused-defines-in-Makefile-and-swit.patch - 871.
↪0 bytes of 871.0 bytes (100%)
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
downloading: patches/0006-sim-erc32-Fix-incorrect-simulator-performance-report.patch - 5.
↪6kB of 5.6kB (100%)
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
downloading: patches/0007-sim-erc32-File-loading-via-command-line-did-not-work.patch - 1.
↪0kB of 1.0kB (100%)
```

```
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
downloading: patches/0008-sim-erc32-Added-v-command-line-switch-for-verbose-ou.patch - 3.6kB of 3.6kB (100%)
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
downloading: patches/0009-sim-erc32-Removed-type-mismatch-compiler-warnings.patch - 1.9kB of 1.9kB (100%)
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
downloading: patches/0010-sim-erc32-Switched-emulated-memory-to-host-endian-or.patch - 16.0kB of 16.0kB (100%)
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
downloading: patches/0011-sim-erc32-use-SIM_AC_OPTION_HOSTENDIAN-to-probe-for-.patch - 14.8kB of 14.8kB (100%)
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
downloading: patches/0012-sim-erc32-Use-memory_iread-function-for-instruction-.patch - 3.8kB of 3.8kB (100%)
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
downloading: patches/0013-sim-erc32-Fix-a-few-compiler-warnings.patch - 2.2kB of 2.2kB (100%)
download: ftp://ftp.rtems.org/pub/rtems/releases/4.11/4.11.0/sources/0014-sim-erc32-Use-gdb-callback-for-UART-I-O-when-linked-.patch -> patches/0014-sim-erc32-Use-gdb-callback-for-UART-I-O-when-linked-.patch
downloading: patches/0014-sim-erc32-Use-gdb-callback-for-UART-I-O-when-linked-.patch - 9.2kB of 9.2kB (100%)
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
downloading: patches/0015-sim-erc32-Access-memory-subsystem-through-struct-mem.patch - 22.9kB of 22.9kB (100%)
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
downloading: patches/0016-sim-erc32-Use-readline.h-for-readline-types-and-func.patch - 1.5kB of 1.5kB (100%)
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
downloading: patches/0017-sim-erc32-Move-local-extern-declarations-into-sis.h.patch - 5.8kB of 5.8kB (100%)
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
downloading: patches/0018-sim-erc32-Add-support-for-LEON3-processor-emulation.patch - 66.7kB of 66.7kB (100%)
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
```

```
downloading: patches/0019-sim-erc32-Add-support-for-LEON2-processor-emulation.patch - 26.1kB of 26.1kB (100%)
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
downloading: patches/0020-sim-erc32-Updated-documentation.patch - 16.1kB of 16.1kB (100%)
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
downloading: patches/0021-sim-erc32-Add-data-watchpoint-support.patch - 10.1kB of 10.1kB (100%)
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
downloading: patches/0022-Add-watchpoint-support-to-gdb-simulator-interface.patch - 25.5kB of 25.5kB (100%)
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
downloading: patches/0023-sim-erc32-ELF-loading-could-fail-on-unaligned-sectio.patch - 1.3kB of 1.3kB (100%)
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
downloading: patches/gdb-sim-arange-inline.diff - 761.0 bytes of 761.0 bytes (100%)
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
downloading: patches/gdb-sim-cgen-inline.diff - 706.0 bytes of 706.0 bytes (100%)
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
downloading: patches/patch-gdb-python-python-config.py - 449.0 bytes of 449.0 bytes (100%)
building: sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1
reporting: tools/rtems-gdb-7.9-1.cfg ->
sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1.txt
reporting: tools/rtems-gdb-7.9-1.cfg ->
sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1.xml
config: tools/rtems-tools-4.11-1.cfg
package: rtems-tools-4.11.0-1
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
downloading: sources/rtems-tools-4.11.0.tar.xz - 1.6MB of 1.6MB (100%)
building: rtems-tools-4.11.0-1
reporting: tools/rtems-tools-4.11-1.cfg -> rtems-tools-4.11.0-1.txt
reporting: tools/rtems-tools-4.11-1.cfg -> rtems-tools-4.11.0-1.xml
config: tools/rtems-kernel-4.11.cfg
package: sparc-rtems4.11-kernel-4.11.0-1
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
downloading: sources/rtems-4.11.0.tar.xz - 9.3MB of 9.3MB (100%)
building: sparc-rtems4.11-kernel-4.11.0-1
reporting: tools/rtems-kernel-4.11.cfg -> sparc-rtems4.11-kernel-4.11.0-1.txt
reporting: tools/rtems-kernel-4.11.cfg -> sparc-rtems4.11-kernel-4.11.0-1.xml
installing: expat-2.1.0-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
installing: sparc-rtems4.11-binutils-2.26-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
installing: sparc-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
installing: sparc-rtems4.11-gdb-7.9-x86_64-freebsd10.1-1 -> /opt/work/rtems/4.11.0
installing: rtems-tools-4.11.0-1 -> /opt/work/rtems/4.11.0
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 defailted in TBD.

5.3 Developer (Unstable)

RTEMS provides open access to it's development processes. We call this the developer set up. The project encouages 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 4 - Host Computer* (page 13) 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 4.12 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*.

5.3.1 POSIX and OS X Host Tools Chain

This procedure will build a SPARC tool chain.

Clone the RTEMS Source Builder (RSB) repository:

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

```
Receiving objects: 100% (5837/5837), 2.48 MiB | 292.00 KiB/s, done.
Resolving deltas: 100% (4014/4014), done.
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:

```
$ cd rsb
$ ./source-builder/sb-check
RTEMS Source Builder - Check, 4.12 (e645642255cc)
Environment is ok
```

Build a tool chain for the SPARC architecure. 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 *4.12* and is on master:

```
$ cd rtems
$ ../source-builder/sb-set-builder \
    --prefix=/usr/home/chris/development/rtems/4.12 4.12/rtems-sparc
RTEMS Source Builder - Set Builder, 4.12 (e645642255cc)
Build Set: 4.12/rtems-sparc
Build Set: 4.12/rtems-autotools.bset
Build Set: 4.12/rtems-autotools-internal.bset
config: tools/rtems-autoconf-2.69-1.cfg
package: autoconf-2.69-x86_64-linux-gnu-1
Creating source directory: sources
download: ftp://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz -> sources/autoconf-2.69.tar.
 ↪gz
downloading: sources/autoconf-2.69.tar.gz - 1.8MB of 1.8MB (100%)
building: autoconf-2.69-x86_64-linux-gnu-1
config: tools/rtems-automake-1.12.6-1.cfg
package: automake-1.12.6-x86_64-linux-gnu-1
download: ftp://ftp.gnu.org/gnu/automake/automake-1.12.6.tar.gz -> sources/automake-1.12.6.
 ↪tar.gz
downloading: sources/automake-1.12.6.tar.gz - 2.0MB of 2.0MB (100%)
Creating source directory: patches
download:     https://git.rtems.org/rtems-tools/plain/tools/4.12/automake/automake-1.12.6-
 ↪bugzilla.redhat.com-1239379.diff -> patches/automake-1.12.6-bugzilla.redhat.com-1239379.
 ↪diff
downloading: patches/automake-1.12.6-bugzilla.redhat.com-1239379.diff - 408.0 bytes of 408.
 ↪0 bytes (100%)
building: automake-1.12.6-x86_64-linux-gnu-1
cleaning: autoconf-2.69-x86_64-linux-gnu-1
cleaning: automake-1.12.6-x86_64-linux-gnu-1
Build Set: Time 0:00:17.465024
Build Set: 4.12/rtems-autotools-base.bset
config: tools/rtems-autoconf-2.69-1.cfg
package: autoconf-2.69-x86_64-linux-gnu-1
building: autoconf-2.69-x86_64-linux-gnu-1
reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-linux-gnu-1.txt
reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-linux-gnu-1.xml
config: tools/rtems-automake-1.12.6-1.cfg
package: automake-1.12.6-x86_64-linux-gnu-1
building: automake-1.12.6-x86_64-linux-gnu-1
reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-linux-gnu-1.txt
reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-linux-gnu-1.xml
installing: autoconf-2.69-x86_64-linux-gnu-1 -> /usr/home/chris/development/rtems/4.12
```

```
installing: automake-1.12.6-x86_64-linux-gnu-1 -> /usr/home/chris/development/rtems/4.12
cleaning: autoconf-2.69-x86_64-linux-gnu-1
cleaning: automake-1.12.6-x86_64-linux-gnu-1
Build Set: Time 0:00:05.358624
Build Set: Time 0:00:22.824422
config: devel/expat-2.1.0-1.cfg
package: expat-2.1.0-x86_64-linux-gnu-1
download: http://downloads.sourceforge.net/project/expat/expat/2.1.0/expat-2.1.0.tar.gz -> sources/expat-2.1.0.tar.gz
  redirect: http://internode.dl.sourceforge.net/project/expat/expat/2.1.0/expat-2.1.0.tar.gz
downloading: sources/expat-2.1.0.tar.gz - 549.4kB of 549.4kB (100%)
building: expat-2.1.0-x86_64-linux-gnu-1
reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-linux-gnu-1.txt
reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-linux-gnu-1.xml
config: tools/rtems-binutils-2.26-1.cfg
package: sparc-rtems4.12-binutils-2.26-x86_64-linux-gnu-1
download: ftp://ftp.gnu.org/gnu/binutils/binutils-2.26.tar.bz2 -> sources/binutils-2.26.tar.bz2
downloading: sources/binutils-2.26.tar.bz2 - 24.4MB of 24.4MB (100%)
download: https://git.rtems.org/rtems-tools/plain/tools/4.12/binutils/binutils-2.26-rtems-aarch64-x86_64.patch -> patches/binutils-2.26-rtems-aarch64-x86_64.patch
downloading: patches/binutils-2.26-rtems-aarch64-x86_64.patch - 3.2kB of 3.2kB (100%)
building: sparc-rtems4.12-binutils-2.26-x86_64-linux-gnu-1
reporting: tools/rtems-binutils-2.26-1.cfg -> sparc-rtems4.12-binutils-2.26-x86_64-linux-gnu-1.txt
reporting: tools/rtems-binutils-2.26-1.cfg -> sparc-rtems4.12-binutils-2.26-x86_64-linux-gnu-1.xml
config: tools/rtems-gcc-6-20160228-newlib-2.3.0.20160226-1.cfg
package: sparc-rtems4.12-gcc-6-20160228-newlib-2.3.0.20160226-x86_64-linux-gnu-1
download: ftp://gcc.gnu.org/pub/gcc/snapshots/6-20160228/gcc-6-20160228.tar.bz2 -> sources/gcc-6-20160228.tar.bz2
downloading: sources/gcc-6-20160228.tar.bz2 - 90.8MB of 90.8MB (100%)
download: ftp://sourceware.org/pub/newlib/newlib-2.3.0.20160226.tar.gz -> sources/newlib-2.3.0.20160226.tar.gz
downloading: sources/newlib-2.3.0.20160226.tar.gz - 16.9MB of 16.9MB (100%)
download: http://www.mpfr.org/mpfr-2.4.2/mpfr-2.4.2.tar.bz2 ->
sources/mpfr-2.4.2.tar.bz2
downloading: sources/mpfr-2.4.2.tar.bz2 - 1.0MB of 1.0MB (100%)
download: http://www.multiprecision.org/mpc/download/mpc-0.8.1.tar.gz -> sources/mpc-0.8.1.tar.gz
downloading: sources/mpc-0.8.1.tar.gz - 532.2kB of 532.2kB (100%)
download: ftp://ftp.gnu.org/gnu/gmp/gmp-4.3.2.tar.bz2 -> sources/gmp-4.3.2.tar.bz2
downloading: sources/gmp-4.3.2.tar.bz2 - 1.8MB of 1.8MB (100%)
building: sparc-rtems4.12-gcc-6-20160228-newlib-2.3.0.20160226-x86_64-linux-gnu-1
reporting: tools/rtems-gcc-6-20160228-newlib-2.3.0.20160226-1.cfg -> sparc-rtems4.12-gcc-6-20160228-newlib-2.3.0.20160226-x86_64-linux-gnu-1.txt
reporting: tools/rtems-gcc-6-20160228-newlib-2.3.0.20160226-1.cfg -> sparc-rtems4.12-gcc-6-20160228-newlib-2.3.0.20160226-x86_64-linux-gnu-1.xml
config: tools/rtems-gdb-7.9-1.cfg
package: sparc-rtems4.12-gdb-7.9-x86_64-linux-gnu-1
download: http://ftp.gnu.org/gnu/gdb/gdb-7.9.tar.xz -> sources/gdb-7.9.tar.xz
downloading: sources/gdb-7.9.tar.xz - 17.0MB of 17.0MB (100%)
download:    https://git.rtems.org/rtems-tools/plain/tools/4.12/gdb/gdb-sim-arange-inline.diff -> patches/gdb-sim-arange-inline.diff
downloading: patches/gdb-sim-arange-inline.diff - 761.0 bytes of 761.0 bytes (100%)
download: https://git.rtems.org/rtems-tools/plain/tools/4.12/gdb/gdb-sim-cgen-inline.diff -> patches/gdb-sim-cgen-inline.diff
```

```
downloading: patches/gdb-sim-cgen-inline.diff - 706.0 bytes of 706.0 bytes (100%)
download:   https://git.rtems.org/rtems-tools/plain/tools/4.12/gdb/gdb-7.9-aarch64-x86_64.
 ↪patch -> patches/gdb-7.9-aarch64-x86_64.patch
downloading: patches/gdb-7.9-aarch64-x86_64.patch - 1.7kB of 1.7kB (100%)
building: sparc-rtems4.12-gdb-7.9-x86_64-linux-gnu-1
reporting: tools/rtems-gdb-7.9-1.cfg -> sparc-rtems4.12-gdb-7.9-x86_64-linux-gnu-1.txt
reporting: tools/rtems-gdb-7.9-1.cfg -> sparc-rtems4.12-gdb-7.9-x86_64-linux-gnu-1.xml
config: tools/rtems-tools-4.12-1.cfg
package: rtems-tools-HEAD-1
Creating source directory: sources/git
git: clone: git://git.rtems.org/rtems-tools.git -> sources/git/rtems-tools.git
git: reset: git://git.rtems.org/rtems-tools.git
git: fetch: git://git.rtems.org/rtems-tools.git -> sources/git/rtems-tools.git
git: checkout: git://git.rtems.org/rtems-tools.git => HEAD
git: pull: git://git.rtems.org/rtems-tools.git
building: rtems-tools-HEAD-1
reporting: tools/rtems-tools-4.12-1.cfg -> rtems-tools-HEAD-1.txt
reporting: tools/rtems-tools-4.12-1.cfg -> rtems-tools-HEAD-1.xml
installing: expat-2.1.0-x86_64-linux-gnu-1 -> /usr/home/chris/development/rtems/4.12
installing:   sparc-rtems4.12-binutils-2.26-x86_64-linux-gnu-1   ->   /usr/home/chris/
 ↪development/rtems/4.12
installing:  sparc-rtems4.12-gcc-6-20160228-newlib-2.3.0.20160226-x86_64-linux-gnu-1  -> /
 ↪usr/home/chris/development/rtems/4.12
installing:  sparc-rtems4.12-gdb-7.9-x86_64-linux-gnu-1  ->  /usr/home/chris/development/
 ↪rtems/4.12
installing: rtems-tools-HEAD-1 -> /usr/home/chris/development/rtems/4.12
cleaning: expat-2.1.0-x86_64-linux-gnu-1
cleaning: sparc-rtems4.12-binutils-2.26-x86_64-linux-gnu-1
cleaning: sparc-rtems4.12-gcc-6-20160228-newlib-2.3.0.20160226-x86_64-linux-gnu-1
cleaning: sparc-rtems4.12-gdb-7.9-x86_64-linux-gnu-1
cleaning: rtems-tools-HEAD-1
Build Set: Time 0:31:09.754219
```

5.3.2 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 4 Section 4 - Microsoft Windows* (page 17) 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.

5.3.2.1 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.

```
 ~
$ 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 subsitute `/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:

```
 ~
$ cd /c/opt/rtems
 /c/opt/rtems
$ git clone git://git.rtems.org/rtems-source-builder.git rsb
Cloning into 'rsb'...
remote: Counting objects: 5716, done.
remote: Compressing objects: 100% (2183/2183), done.
remote: Total 5716 (delta 3919), reused 5071 (delta 3494)
Receiving objects: 100% (5716/5716), 2.46 MiB | 656.00 KiB/s, done.
Resolving deltas: 100% (3919/3919), done.
Checking connectivity... done.
Checking out files: 100% (630/630), done.
 /c/opt/rtems
$ cd rsb
```

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

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

Check the RSB has a valid environment:

```
 /c/opt/rtems/rsb
$ cd rtems
 /c/opt/rtems/rsb/rtems
$ ../source-builder/sb-check
RTEMS Source Builder - Check, 4.11 (01ac76f2f90f)
Environment is ok
 /c/opt/rtems/rsb/rtems
$
```

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:

```
 /c/opt/rtems/rsb/rtems
$ ../source-builder/sb-set-builder --prefix=/c/opt/rtems/4.11 \
                                   --jobs=none 4.11/rtems-i386
```

```
RTEMS Source Builder - Set Builder, 4.11 (01ac76f2f90f)
Build Set: 4.11/rtems-i386
Build Set: 4.11/rtems-autotools.bset
Build Set: 4.11/rtems-autotools-internal.bset
config: tools/rtems-autoconf-2.69-1.cfg
package: autoconf-2.69-x86_64-w64-mingw32-1
Creating source directory: sources
download: ftp://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz -> sources/autoconf-2.69.tar.gz
downloading: sources/autoconf-2.69.tar.gz - 1.8MB of 1.8MB (100%)
 building: autoconf-2.69-x86_64-w64-mingw32-1
config: tools/rtems-automake-1.12.6-1.cfg
package: automake-1.12.6-x86_64-w64-mingw32-1
download: ftp://ftp.gnu.org/gnu/automake/automake-1.12.6.tar.gz -> sources/automake-1.12.6.tar.gz
downloading: sources/automake-1.12.6.tar.gz - 2.0MB of 2.0MB (100%)
 building: automake-1.12.6-x86_64-w64-mingw32-1
cleaning: autoconf-2.69-x86_64-w64-mingw32-1
cleaning: automake-1.12.6-x86_64-w64-mingw32-1
Build Set: Time 0:00:42.515625
Build Set: 4.11/rtems-autotools-base.bset
config: tools/rtems-autoconf-2.69-1.cfg
package: autoconf-2.69-x86_64-w64-mingw32-1
building: autoconf-2.69-x86_64-w64-mingw32-1
reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-w64-mingw32-1.txt
reporting: tools/rtems-autoconf-2.69-1.cfg -> autoconf-2.69-x86_64-w64-mingw32-1.xml
config: tools/rtems-automake-1.12.6-1.cfg
package: automake-1.12.6-x86_64-w64-mingw32-1
building: automake-1.12.6-x86_64-w64-mingw32-1
reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-w64-mingw32-1.txt
reporting: tools/rtems-automake-1.12.6-1.cfg -> automake-1.12.6-x86_64-w64-mingw32-1.xml
tarball: tar/rtems-4.11-autotools-x86_64-w64-mingw32-1.tar.bz2
installing: autoconf-2.69-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
installing: automake-1.12.6-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
cleaning: autoconf-2.69-x86_64-w64-mingw32-1
cleaning: automake-1.12.6-x86_64-w64-mingw32-1
Build Set: Time 0:00:37.718750
Build Set: Time 0:01:20.234375
config: devel/expat-2.1.0-1.cfg
package: expat-2.1.0-x86_64-w64-mingw32-1
download: http://downloads.sourceforge.net/project/expat/expat/2.1.0/expat-2.1.0.tar.gz -> sources/expat-2.1.0.tar.gz
 redirect: http://iweb.dl.sourceforge.net/project/expat/expat/2.1.0/expat-2.1.0.tar.gz
downloading: sources/expat-2.1.0.tar.gz - 549.4kB of 549.4kB (100%)
building: expat-2.1.0-x86_64-w64-mingw32-1
reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-w64-mingw32-1.txt
reporting: devel/expat-2.1.0-1.cfg -> expat-2.1.0-x86_64-w64-mingw32-1.xml
config: tools/rtems-binutils-2.24-1.cfg
package: i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1
download: ftp://ftp.gnu.org/gnu/binutils/binutils-2.24.tar.bz2 -> sources/binutils-2.24.tar.bz2
downloading: sources/binutils-2.24.tar.bz2 - 21.7MB of 21.7MB (100%)
building: i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1
reporting: tools/rtems-binutils-2.24-1.cfg -> i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1.txt
reporting: tools/rtems-binutils-2.24-1.cfg -> i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1.xml
```

```
config: tools/rtems-gcc-4.9.3-newlib-2.2.0-20150423-1.cfg
package: i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1
download: ftp://ftp.gnu.org/gnu/gcc/gcc-4.9.3/gcc-4.9.3.tar.bz2 -> sources/gcc-4.9.3.tar.bz2
downloading: sources/gcc-4.9.3.tar.bz2 - 85.8MB of 85.8MB (100%)
download: ftp://sourceware.org/pub/newlib/newlib-2.2.0.20150423.tar.gz -> sources/newlib-2.2.0.20150423.tar.gz
downloading: sources/newlib-2.2.0.20150423.tar.gz - 16.7MB of 16.7MB (100%)
download: http://www.mpfr.org/mpfr-3.0.1/mpfr-3.0.1.tar.bz2 -> sources/mpfr-3.0.1.tar.bz2
downloading: sources/mpfr-3.0.1.tar.bz2 - 1.1MB of 1.1MB (100%)
download: http://www.multiprecision.org/mpc/download/mpc-0.8.2.tar.gz -> sources/mpc-0.8.2.tar.gz
downloading: sources/mpc-0.8.2.tar.gz - 535.5kB of 535.5kB (100%)
download: ftp://ftp.gnu.org/gnu/gmp/gmp-5.0.5.tar.bz2 -> sources/gmp-5.0.5.tar.bz2
downloading: sources/gmp-5.0.5.tar.bz2 - 2.0MB of 2.0MB (100%)
building: i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1
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
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
config: tools/rtems-gdb-7.9-1.cfg
package: i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1
download: http://ftp.gnu.org/gnu/gdb/gdb-7.9.tar.xz -> sources/gdb-7.9.tar.xz
downloading: sources/gdb-7.9.tar.xz - 17.0MB of 17.0MB (100%)
download: https://git.rtems.org/rtems-tools/plain/tools/4.11/gdb/gdb-sim-arange-inline.diff -> patches/gdb-sim-arange-inline.diff
downloading: patches/gdb-sim-arange-inline.diff - 761.0 bytes of 761.0 bytes (100%)
download: https://git.rtems.org/rtems-tools/plain/tools/4.11/gdb/gdb-sim-cgen-inline.diff -> patches/gdb-sim-cgen-inline.diff
downloading: patches/gdb-sim-cgen-inline.diff - 706.0 bytes of 706.0 bytes (100%)
building: i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1
reporting: tools/rtems-gdb-7.9-1.cfg ->
i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1.txt
reporting: tools/rtems-gdb-7.9-1.cfg ->
i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1.xml
config: tools/rtems-tools-4.11-1.cfg
package: rtems-tools-4.11-1
Creating source directory: sources/git
git: clone: git://git.rtems.org/rtems-tools.git -> sources/git/rtems-tools.git
git: reset: git://git.rtems.org/rtems-tools.git
git: fetch: git://git.rtems.org/rtems-tools.git -> sources/git/rtems-tools.git
git: checkout: git://git.rtems.org/rtems-tools.git => 4.11
git: pull: git://git.rtems.org/rtems-tools.git
building: rtems-tools-4.11-1
reporting: tools/rtems-tools-4.11-1.cfg -> rtems-tools-4.11-1.txt
reporting: tools/rtems-tools-4.11-1.cfg -> rtems-tools-4.11-1.xml
config: tools/rtems-kernel-4.11.cfg
installing: expat-2.1.0-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
installing: i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
installing: i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
installing: i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1 -> C:\opt\rtems\4.11
installing: rtems-tools-4.11-1 -> C:\opt\rtems\4.11
cleaning: expat-2.1.0-x86_64-w64-mingw32-1
cleaning: i386-rtems4.11-binutils-2.24-x86_64-w64-mingw32-1
cleaning: i386-rtems4.11-gcc-4.9.3-newlib-2.2.0.20150423-x86_64-w64-mingw32-1
cleaning: i386-rtems4.11-gdb-7.9-x86_64-w64-mingw32-1
```

```
cleaning: rtems-tools-4.11-1
Build Set: Time 1:32:58.972919
 /c/opt/rtems/rsb/rtems
$
```

5.3.2.2 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:

```
 /c
$ cd /c/opt/rtems
 /c/opt/rtems
$ export PATH=/c/opt/rtems/4.11/bin:$PATH
 /c/opt/rtems
$
```

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

```
 /c/opt/rtems
$ mkdir kernel
 /c/opt/rtems
$ cd kernel
 /c/opt/rtems/kernel
$ git clone git://git.rtems.org/rtems.git rtems
Cloning into 'rtems'...
remote: Counting objects: 482766, done.
remote: Compressing objects: 100% (88781/88781), done.
remote: Total 482766 (delta 389610), reused 475155 (delta 383437)
Receiving objects: 100% (482766/482766), 69.77 MiB | 697.00 KiB/s, done.
Resolving deltas: 100% (389610/389610), done.
Checking connectivity... done.
Checking out files: 100% (10626/10626), done.
 /c/opt/rtems/kernel
$ cd rtems
 /c/opt/rtems/kernel/rtems
$ git checkout 4.11
Checking out files: 100% (2553/2553), done.
Branch 4.11 set up to track remote branch 4.11 from origin.
Switched to a new branch '4.11'
 /c/opt/rtems/kernel
$
```

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:

```
 /c/opt/rtems/kernel/rtems
$ ./bootstrap -c
removing automake generated Makefile.in files
removing configure files
removing aclocal.m4 files
```

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

```
 /c/opt/rtems/kernel/rtems
$ ./bootstrap -p
Generating ./c/src/ada/preinstall.am
Generating ./c/src/lib/libbsp/arm/altera-cyclone-v/preinstall.am
Generating ./c/src/lib/libbsp/arm/atsam/preinstall.am
Generating ./c/src/lib/libbsp/arm/beagle/preinstall.am
Generating ./c/src/lib/libbsp/arm/csb336/preinstall.am
Generating ./c/src/lib/libbsp/arm/csb337/preinstall.am
Generating ./c/src/lib/libbsp/arm/edb7312/preinstall.am
Generating ./c/src/lib/libbsp/arm/gdbarmsim/preinstall.am
  .......
Generating ./cpukit/score/cpu/mips/preinstall.am
Generating ./cpukit/score/cpu/moxie/preinstall.am
Generating ./cpukit/score/cpu/nios2/preinstall.am
Generating ./cpukit/score/cpu/no_cpu/preinstall.am
Generating ./cpukit/score/cpu/or1k/preinstall.am
Generating ./cpukit/score/cpu/powerpc/preinstall.am
Generating ./cpukit/score/cpu/sh/preinstall.am
Generating ./cpukit/score/cpu/sparc/preinstall.am
Generating ./cpukit/score/cpu/sparc64/preinstall.am
Generating ./cpukit/score/cpu/v850/preinstall.am
Generating ./cpukit/score/preinstall.am
Generating ./cpukit/telnetd/preinstall.am
Generating ./cpukit/wrapup/preinstall.am
Generating ./cpukit/zlib/preinstall.am
 /c/opt/rtems/kernel/rtems
```

Finally we run the RSB's parallel `bootstrap` command:

```
$ /c/opt/rtems/rsb/source-builder/sb-bootstrap
RTEMS Source Builder - RTEMS Bootstrap, 4.11 (76188ee494dd)
  1/139: autoreconf: configure.ac
  2/139: autoreconf: c/configure.ac
  3/139: autoreconf: c/src/configure.ac
  4/139: autoreconf: c/src/ada-tests/configure.ac
  5/139: autoreconf: c/src/lib/libbsp/arm/configure.ac
  6/139: autoreconf: c/src/lib/libbsp/arm/altera-cyclone-v/configure.ac
  7/139: autoreconf: c/src/lib/libbsp/arm/atsam/configure.ac
  8/139: autoreconf: c/src/lib/libbsp/arm/beagle/configure.ac
  9/139: autoreconf: c/src/lib/libbsp/arm/csb336/configure.ac
 10/139: autoreconf: c/src/lib/libbsp/arm/csb337/configure.ac
 11/139: autoreconf: c/src/lib/libbsp/arm/edb7312/configure.ac
  .......
129/139: autoreconf: testsuites/samples/configure.ac
130/139: autoreconf: testsuites/smptests/configure.ac
131/139: autoreconf: testsuites/sptests/configure.ac
132/139: autoreconf: testsuites/tmtests/configure.ac
133/139: autoreconf: testsuites/tools/configure.ac
134/139: autoreconf: testsuites/tools/generic/configure.ac
135/139: autoreconf: tools/build/configure.ac
136/139: autoreconf: tools/cpu/configure.ac
137/139: autoreconf: tools/cpu/generic/configure.ac
138/139: autoreconf: tools/cpu/nios2/configure.ac
139/139: autoreconf: tools/cpu/sh/configure.ac
Bootstrap time: 0:20:38.759766
 /c/opt/rtems/kernel/rtems
$
```

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:

```
 /c/opt/rtems/kernel/rtems
$ cd ..
 /c/opt/rtems/kernel
$ mkdir pc686
 /c/opt/rtems/kernel
$ cd pc686
 /c/opt/rtems/kernel/pc686
$
```

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

```
 /c/opt/rtems/kernel/pc686
$ /c/opt/rtems/kernel/rtems/configure --prefix=/c/opt/rtems/4.11 \
       --target=i386-rtems4.11 --disable-networking --enable-rtemsbsp=pc686
checking for gmake... no
checking for make... make
checking for RTEMS Version... 4.11.99.0
checking build system type... x86_64-pc-mingw64
checking host system type... x86_64-pc-mingw64
checking target system type... i386-pc-rtems4.11
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking that generated files are newer than configure... done
configure: creating ./config.status
configure: configuring in ./tools/build
  .......
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile

target architecture: i386.
available BSPs: pc686.
'make all' will build the following BSPs: pc686.
other BSPs can be built with 'make RTEMS_BSP="bsp1 bsp2 ..."'

config.status: creating Makefile
 /c/opt/rtems/kernel/pc686
$
```

Build the kernel:

```
 /c/opt/rtems/kernel/pc686
$ make
Making all in tools/build
make[1]: Entering directory '/c/opt/rtems/kernel/pc686/tools/build'
make  all-am
```

```
make[2]: Entering directory '/c/opt/rtems/kernel/pc686/tools/build'
gcc -DHAVE_CONFIG_H -I. -I/c/opt/rtems/kernel/rtems/tools/build     -g -O2 -MT
cklength.o -MD -MP -MF .deps/cklength.Tpo -c -o cklength.o
/c/opt/rtems/kernel/rtems/tools/build/cklength.c
gcc -DHAVE_CONFIG_H -I. -I/c/opt/rtems/kernel/rtems/tools/build     -g -O2 -MT
eolstrip.o -MD -MP -MF .deps/eolstrip.Tpo -c -o eolstrip.o
/c/opt/rtems/kernel/rtems/tools/build/eolstrip.c
  ..........
i386-rtems4.11-objcopy -O binary nsecs.nxe nsecs.bin
../../../../../pc686/build-tools/bin2boot -v nsecs.ralf 0x00097E00
../../../../../pc686/lib/start16.bin 0x00097C00 0 nsecs.bin 0x00100000 0
header address       0x00097e00, its memory size 0xzx
first  image address 0x00097c00, its memory size 0x00000200
second image address 0x00100000, its memory size 0x0003d800
rm -f nsecs.nxe
make[6]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites/
 ↪samples/nsecs'
make[5]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites/
 ↪samples'
make[4]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites/
 ↪samples'
make[4]: Entering directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites'
make[4]: Nothing to be done for 'all-am'.
make[4]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites'
make[3]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686/testsuites'
make[2]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c/pc686'
make[1]: Leaving directory '/c/opt/rtems/kernel/pc686/i386-rtems4.11/c'
make[1]: Entering directory '/c/opt/rtems/kernel/pc686'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/c/opt/rtems/kernel/pc686'
 /c/opt/rtems/kernel/pc696
$
```

Install the kernel to our prefix:

```
 /c/opt/rtems/kernel/pc686
$ make install
Making install in tools/build
make[1]: Entering directory '/c/opt/rtems/kernel/pc686/tools/build'
make[2]: Entering directory '/c/opt/rtems/kernel/pc686/tools/build'
 /usr/bin/mkdir -p '/c/opt/rtems/4.11/bin'
  /usr/bin/install -c cklength.exe eolstrip.exe packhex.exe unhex.exe
  rtems-bin2c.exe '/c/opt/rtems/4.11/bin'
 /usr/bin/mkdir -p '/c/opt/rtems/4.11/bin'
 /usr/bin/install -c install-if-change '/c/opt/rtems/4.11/bin'
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/c/opt/rtems/kernel/pc686/tools/build'
make[1]: Leaving directory '/c/opt/rtems/kernel/pc686/tools/build'
Making install in tools/cpu
make[1]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu'
Making install in generic
make[2]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu/generic'
make[3]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu/generic'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/c/opt/rtems/kernel/pc686/tools/cpu/generic'
make[2]: Leaving directory '/c/opt/rtems/kernel/pc686/tools/cpu/generic'
```

```
make[2]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu'
make[3]: Entering directory '/c/opt/rtems/kernel/pc686/tools/cpu'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
  ..........
make[2]: Entering directory '/c/opt/rtems/kernel/pc686'
make[2]: Nothing to be done for 'install-exec-am'.
 /usr/bin/mkdir -p '/c/opt/rtems/4.11/make'
 /usr/bin/install -c -m 644 /c/opt/rtems/kernel/rtems/make/main.cfg
 /c/opt/rtems/kernel/rtems/make/leaf.cfg '/c/opt/rtems/4.11/make'
 /usr/bin/mkdir -p '/c/opt/rtems/4.11/share/rtems4.11/make/Templates'
 /usr/bin/install -c -m 644
 /c/opt/rtems/kernel/rtems/make/Templates/Makefile.dir
 /c/opt/rtems/kernel/rtems/make/Templates/Makefile.leaf
 /c/opt/rtems/kernel/rtems/make/Templates/Makefile.lib
 '/c/opt/rtems/4.11/share/rtems4.11/make/Templates'
 /usr/bin/mkdir -p '/c/opt/rtems/4.11/make/custom'
 /usr/bin/install -c -m 644 /c/opt/rtems/kernel/rtems/make/custom/default.cfg
 '/c/opt/rtems/4.11/make/custom'
make[2]: Leaving directory '/c/opt/rtems/kernel/pc686'
make[1]: Leaving directory '/c/opt/rtems/kernel/pc686'
 /c/opt/rtems/kernel/pc686
$
```

5.4 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.

5.4.1 Development Sources

Create a new location to build the RTEMS kernel:

```
$ cd
$ cd development/rtems
$ mkdir kernel
$ cd kernel
```

Clone the RTEMS respository:

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

5.4.2 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:

```
$ export PATH=$HOME/development/rtems/4.12/bin:$PATH
```

5.4.3 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 bootsrap commands:

```
$ cd rtems
$ ./bootstrap -c && ./bootstrap -p && \
            $HOME/development/rtems/rsb/source-builder/sb-bootstrap
removing automake generated Makefile.in files
removing configure files
removing aclocal.m4 files
Generating ./cpukit/dtc/libfdt/preinstall.am
Generating ./cpukit/zlib/preinstall.am
Generating ./cpukit/libdl/preinstall.am
```

```
Generating ./cpukit/posix/preinstall.am
Generating ./cpukit/pppd/preinstall.am
Generating ./cpukit/librpc/preinstall.am
Generating ./cpukit/preinstall.am
Generating ./cpukit/sapi/preinstall.am
Generating ./cpukit/score/preinstall.am
Generating ./cpukit/score/cpu/mips/preinstall.am
Generating ./cpukit/score/cpu/sh/preinstall.am
Generating ./cpukit/score/cpu/sparc/preinstall.am
Generating ./cpukit/score/cpu/no_cpu/preinstall.am
Generating ./cpukit/score/cpu/arm/preinstall.am
Generating ./cpukit/score/cpu/m32c/preinstall.am
Generating ./cpukit/score/cpu/moxie/preinstall.am
Generating ./cpukit/score/cpu/v850/preinstall.am
Generating ./cpukit/score/cpu/sparc64/preinstall.am
Generating ./cpukit/score/cpu/or1k/preinstall.am
Generating ./cpukit/score/cpu/i386/preinstall.am
Generating ./cpukit/score/cpu/nios2/preinstall.am
Generating ./cpukit/score/cpu/epiphany/preinstall.am
Generating ./cpukit/score/cpu/m68k/preinstall.am
Generating ./cpukit/score/cpu/lm32/preinstall.am
Generating ./cpukit/score/cpu/powerpc/preinstall.am
Generating ./cpukit/score/cpu/bfin/preinstall.am
Generating ./cpukit/libpci/preinstall.am
Generating ./cpukit/libcrypt/preinstall.am
Generating ./cpukit/rtems/preinstall.am
Generating ./cpukit/telnetd/preinstall.am
Generating ./cpukit/libnetworking/preinstall.a
 ......
Generating ./c/src/lib/libbsp/powerpc/gen5200/preinstall.am
Generating ./c/src/lib/libbsp/powerpc/mpc55xxevb/preinstall.am
Generating ./c/src/lib/libbsp/bfin/TLL6527M/preinstall.am
Generating ./c/src/lib/libbsp/bfin/bf537Stamp/preinstall.am
Generating ./c/src/lib/libbsp/bfin/eZKit533/preinstall.am
Generating ./c/src/librtems++/preinstall.am
Generating ./c/src/libchip/preinstall.am
Generating ./c/src/wrapup/preinstall.am
Generating ./c/src/ada/preinstall.am
RTEMS Source Builder - RTEMS Bootstrap, 4.12 (e645642255cc modified)
  1/139: autoreconf: configure.ac
  2/139: autoreconf: cpukit/configure.ac
  3/139: autoreconf: tools/cpu/configure.ac
  4/139: autoreconf: tools/cpu/generic/configure.ac
  5/139: autoreconf: tools/cpu/sh/configure.ac
  6/139: autoreconf: tools/cpu/nios2/configure.ac
  7/139: autoreconf: tools/build/configure.ac
  8/139: autoreconf: doc/configure.ac
 ......
124/139: autoreconf: c/src/make/configure.ac
125/139: autoreconf: c/src/librtems++/configure.ac
126/139: autoreconf: c/src/ada-tests/configure.ac
127/139: autoreconf: testsuites/configure.ac
128/139: autoreconf: testsuites/libtests/configure.ac
129/139: autoreconf: testsuites/mptests/configure.ac
130/139: autoreconf: testsuites/fstests/configure.ac
131/139: autoreconf: testsuites/sptests/configure.ac
132/139: autoreconf: testsuites/tmtests/configure.ac
```

```
133/139: autoreconf: testsuites/smptests/configure.ac
134/139: autoreconf: testsuites/tools/configure.ac
135/139: autoreconf: testsuites/tools/generic/configure.ac
136/139: autoreconf: testsuites/psxtests/configure.ac
137/139: autoreconf: testsuites/psxtmtests/configure.ac
138/139: autoreconf: testsuites/rhealstone/configure.ac
139/139: autoreconf: testsuites/samples/configure.ac
Bootstrap time: 0:02:47.398824
```

5.4.4 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:

```
$ cd ..
$ mkdir erc32
$ 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:

```
$ $HOME/development/rtems/kernel/rtems/configure --prefix=$HOME/development/rtems/4.12 \
                  --target=sparc-rtems4.12 --enable-rtemsbsp=erc32 --enable-posix \
                  --disable-networking
checking for gmake... no
checking for make... make
checking for RTEMS Version... 4.11.99.0
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking target system type... sparc-unknown-rtems4.12
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking that generated files are newer than configure... done
 ......
checking target system type... sparc-unknown-rtems4.12
checking rtems target cpu... sparc
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for sparc-rtems4.12-strip... sparc-rtems4.12-strip
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking that generated files are newer than configure... done
configure: creating ./config.status
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'
```

```
make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
make[2]: Nothing to be done for 'all-am'.
make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
Making all in testsuites/tools
make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools'
Making all in generic
make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools/
 ↪generic'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools/
 ↪generic'
make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools'
make[2]: Nothing to be done for 'all-am'.
make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools'
make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/testsuites/tools'
Making all in sparc-rtems4.12/c
make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/c'
Making all in .
make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/c'
Configuring RTEMS_BSP=erc32
checking for gmake... no
checking for make... make
checking build system type... x86_64-pc-linux-gnu
checking host system type... sparc-unknown-rtems4.12
 ......
sparc-rtems4.12-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
sparc-rtems4.12-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
mv -f .deps/empty.Tpo .deps/empty.Po
mv -f .deps/init.Tpo .deps/init.Po
sparc-rtems4.12-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
sparc-rtems4.12-nm -g -n nsecs.exe > nsecs.num
sparc-rtems4.12-size nsecs.exe
   text    data     bss     dec     hex filename
 121392    1888    6624  129904   1fb70 nsecs.exe
cp nsecs.exe nsecs.ralf
make[6]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/ c/
 ↪erc32/testsuites/samples/nsecs'
make[5]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/ c/
 ↪erc32/testsuites/samples'
make[4]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/ c/
 ↪erc32/testsuites/samples'
```

```
make[4]: Entering directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/ c/
 ↪erc32/testsuites'
make[4]: Nothing to be done for 'all-am'.
make[4]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/ c/
 ↪erc32/testsuites'
make[3]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/ c/
 ↪erc32/testsuites'
make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/ c/
 ↪erc32'
make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/c'
make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32'
```

5.4.5 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 locaiton 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:

```
$ make install
Making install in tools/build
make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/build'
make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/build'
/bin/mkdir -p '/home/chris/development/rtems/4.12/bin'
/usr/bin/install -c cklength eolstrip packhex unhex rtems-bin2c '/home/chris/development/
 ↪rtems/4.12/bin'
/bin/mkdir -p '/home/chris/development/rtems/4.12/bin'
/usr/bin/install -c install-if-change '/home/chris/development/rtems/4.12/bin'
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/build'
make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/build'
Making install in tools/cpu
make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
Making install in generic
make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
make[3]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu/generic'
make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
make[3]: Entering directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu'
make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/tools/cpu
 ......
make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32/sparc-rtems4.12/c'
make[1]: Entering directory '/home/chris/development/rtems/kernel/erc32'
make[2]: Entering directory '/home/chris/development/rtems/kernel/erc32'
make[2]: Nothing to be done for 'install-exec-am'.
```

```
/bin/mkdir -p '/home/chris/development/rtems/4.12/make'
/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/4.12/
 ↪make'
/bin/mkdir -p '/home/chris/development/rtems/4.12/share/rtems4.12/make/Templates'
/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/4.12/share/rtems4.12/make/Templates'
/bin/mkdir -p '/home/chris/development/rtems/4.12/make/custom'
/usr/bin/install -c -m 644 /home/chris/development/rtems/kernel/rtems/make/custom/default.
 ↪cfg '/home/chris/development/rtems/4.12/make/custom'
make[2]: Leaving directory '/home/chris/development/rtems/kernel/erc32'
make[1]: Leaving directory '/home/chris/development/rtems/kernel/erc32'
```

5.4.6 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 evalulate 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:

```
$ cd ../rtems
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
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:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   cpukit/libmisc/shell/main_halt.c

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:

```
$ 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:

```
shell: Add more help detail to the halt command.

Closes #9876.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# Explicit paths specified without -i or -o; assuming --only paths...
#
# Committer: Chris Johns <chrisj@rtems.org>
#
# On branch master
# Your branch is up-to-date with 'origin/master'.
#
# Changes to be committed:
#       modified:   cpukit/libmisc/shell/main_halt.c
```

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

```
$ git commit cpukit/libmisc/shell/main_halt.c
[master 9f44dc9] shell: Add more help detail to the halt command.
 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:

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

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

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

This patch can be attached to a ticket.

CHAPTER

SIX

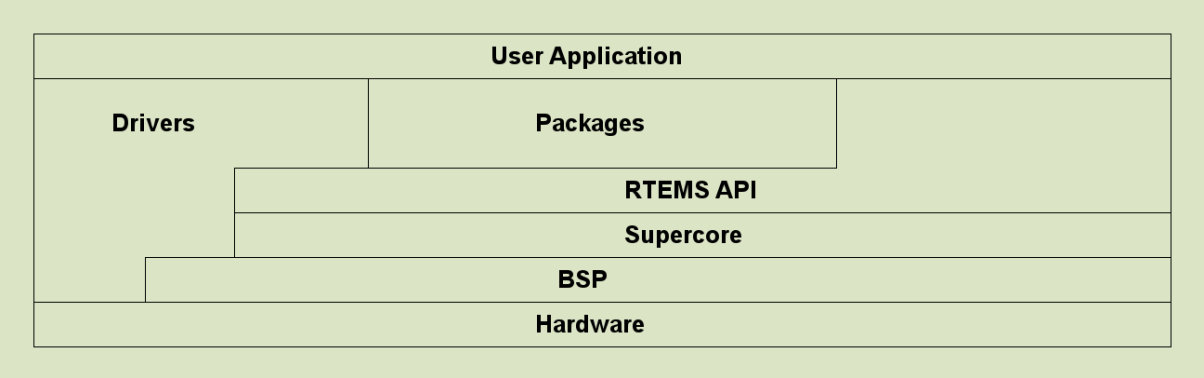
HARDWARE

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

6.1 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 BPSs. 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.

6.2 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 **multlib**. Multlibs are manage 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 capabilites 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 executable 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

6.3 Board Support Packages (BSP)

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

RTEMS includes 173 board support packages in it's 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
```

6.4 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 quaility 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	• The RTEMS Kernel must build without error. • Tests are run on target hardware.
2	• The RTEMS Kernel must build without error. • Tests can be run on simulation.
3	• The RTEMS Kernel must build without error. • There are no test results.
4	• The RTEMS Kernel does not build.
5	• The BSP is to be removed after the next release.

2. An architecuture'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.

CHAPTER

SEVEN

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.

7.1 RTEMS Project Support

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

7.1.1 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.

7.1.2 Documentation

The latest user documentation can always be found at the Documentation Site.

7.1.3 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.

7.1.4 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.

7.1.5 Developers

Developers can find help and support on the mailing lists, see *Chapter 7 Section 1.3 - Mailing Lists* (page 64).

Technical documents including design, Google Summer of Code, ESA SOCIS can be found on the Developer Site.

7.2 Commercial Support Services

The wider RTEMS community has developers and orgainizations 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 unsrue 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.

CHAPTER

EIGHT

ADDITIONAL INFORMATION

Information and details that do not belong in other sections go here.

8.1 Important Terms

8.1.1 Waf

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

8.1.2 Test Suite

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

8.2 Command List

List of all available commands in RTEMS

8.3 Program List

List of all available programs in RTEMS

CHAPTER

NINE

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 inteagration 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 simplied path structure to make it easier to building native Windows programs.

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

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

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

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

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/

- genindex
- search

INDEX