2

Directory Structure

The **RTEMS** directory structure is designed to meet the following requirements:

- *encourage development of modular components.*
- *isolate processor and target dependent code, while allowing as much common source code as possible to be shared across multiple processors and targets.*
- *allow multiple **RTEMS** users to perform simultaneous compilation of **RTEMS** and its support facilities for different processors and targets.*

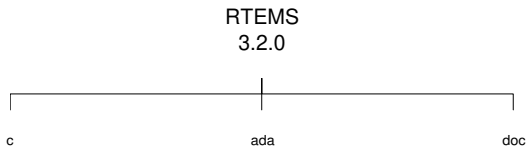
The resulting directory structure has processor and target dependent source files isolated from generic files. When **RTEMS** is built, object directories and an install point will be automatically created based upon the target BSP selected. The placement of object files based upon the selected **BSP** name insures that object files are not mixed across CPUs or targets. This in combination with the make files allows the specific compilation options to be tailored for a particular target board. For example, the efficiency of the memory subsystem for a particular target board may be sensitive to the alignment of data structures, while on another target board with the same processor memory may be very limited. For the first target, the options could specify very strict alignment requirements, while on the second the data structures could be "packed" to conserve memory. It is impossible to achieve this degree of flexibility without providing source code.

2.1 Suites

The **RTEMS** source tree is organized based on the following four variables:

- *language,*
- *target processor,*
- *target board, and*
- *compiler vendor (**Ada** only).*

The language may be either **C** or **Ada** and there is currently nothing shared between the source trees for these two implementations of **RTEMS**. The user generally selects the subdirectory for the implementation they are using and ignores that for the other implementation. The only exceptions to this normally occurs when comparing the source code for the two implementations or when porting both to a new CPU or target board. The following shows the top level **RTEMS** directory structure which includes directories for each language implementation and a language independent source documentation directory. The source documentation directory is currently not supported..



For the most part, the **C** and **Ada** source trees follow exactly the same layout. The primary differences are that there are some directories which exist only in one or the either implementation and that the **Ada** tree includes a "vendor" directory in a number of places. The vendor name is included because often vendor specific language features are used to optimize performance (such as the Machine_Code package) and that the compilation and library management utilities vary widely between vendors. The vendor directories are included in the "make" subsystem so there may be a unique set of make rules for each vendor and as the parent of "cpu" directories. A vendor may provide a feature across all of its compilers which aid in porting RTEMS. For example, many vendors of Ada83 compilers provided their own package for efficient bitwise operations.

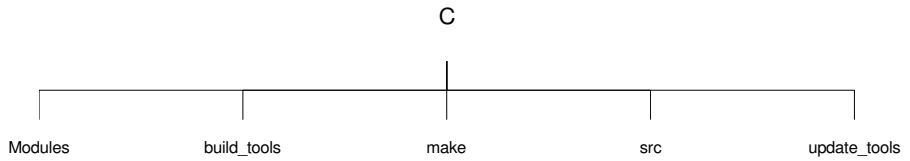
Each of the following sections will describe the contents of the directories in RTEMS for each language supported.

2.1.1 C Suites

The following suites are currently included with the **C** implementation of **RTEMS**:

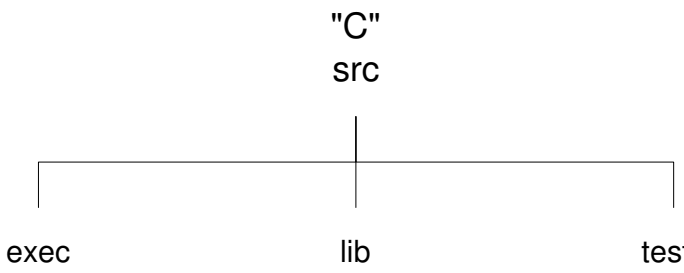
Suite Name	Base Directory
RTEMS Source Code	$RTEMS_SRC_BASE/c/src/exec
Support Libraries (BSPs, C library, CPU support)	$RTEMS_ROOT/c/src/lib
Single Processor Tests	$RTEMS_ROOT/c/src/tests/sptests
Timing Tests	$RTEMS_ROOT/c/src/tests/tmtests
Multiprocessor Tests	$RTEMS_ROOT/c/src/tests/mptests
Sample Applications	$RTEMS_ROOT/c/src/tests/samples
Modules	$RTEMS_ROOT/c/Modules
RTEMS Build Tools	$RTEMS_SRC_BASE/c/build_tools
Make Support	$RTEMS_ROOT/c/make

The top level directory structure for the **C** implementation of RTEMS is as follows:



This directory contains the subdirectories which contain the entire **C** implementation of the **RTEMS** executive. The "Modules" directory contains modules files which correspond to enviroments supported in the current version of **RTEMS**. The "build-tools" directory contains an assortment of support tools for the **RTEMS** development environment. Two subdirectories exist under build-tools" which contain scripts (executables) and source for the support tools. The "make" directory contains configuration files and subdirectories which provide a robust host and cross-target makefile system supporting the building of the executive for numerous application environments. The "update_tools" directory contains utilities which aid in the updating from a previous version to the current version of the **RTEMS** executive.

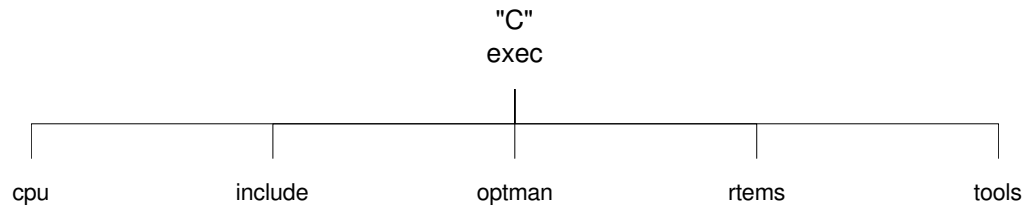
The "src" directory structure for the **C** implementation of **RTEMS** is as follows:



This directory contains all source files that comprises the **RTEMS** executive, supported target board support packages, and the **RTEMS** Test Suite.

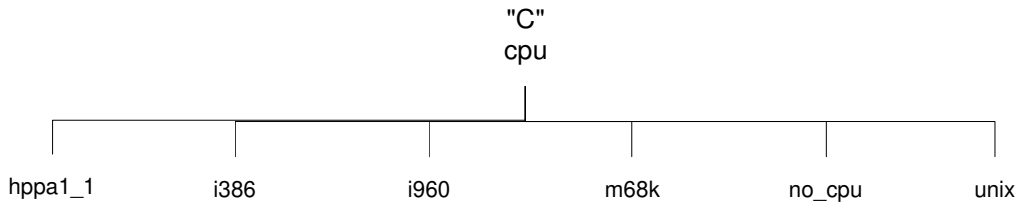
2.1.1.1 Executive Source Directory

The "exec" directory structure for the **C** implementation is as follows:



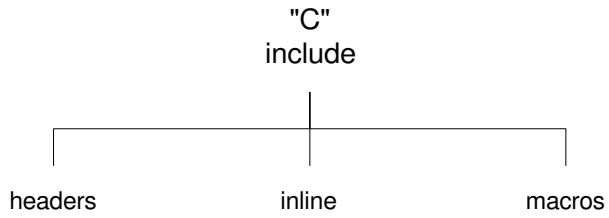
This directory contains a set of subdirectories which contains the source files comprising the executive portion of the RTEMS development environment. At this point the CPU dependent and independent source code files are separated into distinct directory trees. The "rtems" subdirectory contains the C language source files for each module found in the executive. Also included in this directory are the subdirectories containing the executive's header files, optional manager capability, and support tool "size_rtems" which determines the actual size of a built configuration of the **RTEMS** executive.

The "cpu" directory structure for the **C** implementation of **RTEMS** is as follows:



This directory contains a subdirectory for each target CPU supported by the 3.2.0 release of the RTEMS executive. Each processor directory contains the CPU dependent code necessary to host RTEMS. The "no_cpu" directory provides a starting point for developing a new port to an unsupported processor. The files contained within the "no_cpu" directory may also be used as a reference for the other ports to specific processors.

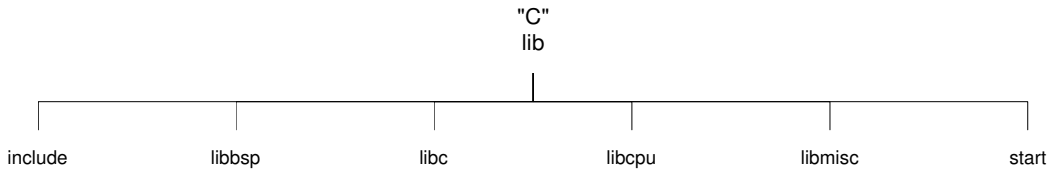
The "include" directory structure for the **C** implementaion is as follows:



This directory contains all header files necessary to define the executive portion of the **RTEMS** executive. The "include" subdirectory contains the headers for all the modules found in the executive portion. It is possible that the header files may reference an inline file (filename.inl) which contains an implementation of module related macros. Two subdirectories are provided to support compilers which allow inlining of macros and those that do not support this feature. Therefore, identical functions are found in the "inline" and "macros" subdirectories, but they differ in their implementation.

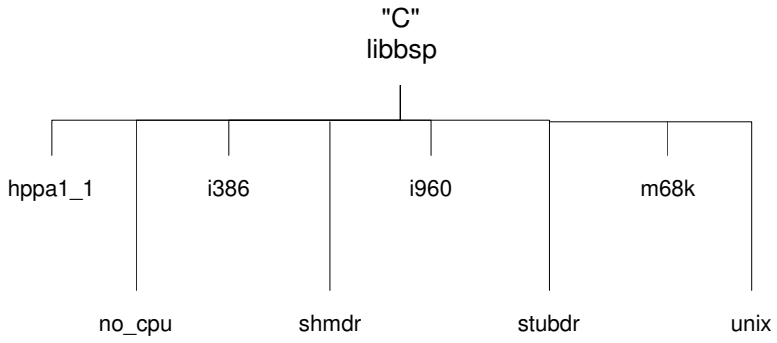
2.1.1.2 Support Library Source Directory

The "lib" directory structure for the **C** implementaion is as follows:



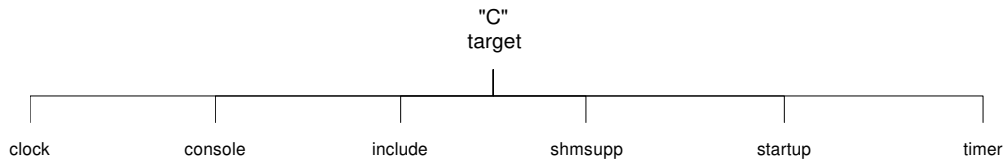
This directory contains the source code for libraries which support the development of applications using the RTEMS Development Environment. Board support packages (BSPs), processor environment start up code, C library support, common BSP header files, and miscellaneous support functions subdirectories are provided.

The "libbsp" directory structure for the **C** implementaion is as follows:



This directory provides all the BSPs provied with the 3.2.0 version of the **RTEMS** executive. The subdirectories are divided, as discussed previously, based on specific processor family, then further breaking down into specific target board environments. The "shmdr" subdirectory provides the implementation of a shared memory driver which supports the multiprocessing protion of the executive. In addition, two starting point subdirectories are provided for reference. The "no_cpu" subdirectory provides a template BSP which can be used to develop a specific BSP for an unsupported target board. The "stubdr" subdirectory provides stubbed out BSPs. These files may aid in preliminary testing of the RTEMS development environment that has been built for no particular target in mind.

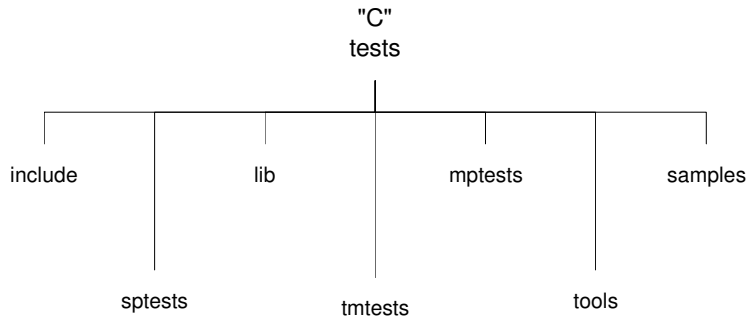
An example of a subdirectory containing BSPs for a family of processors is as follows:



This directory provides the typical modules which comprise an RTEMS BSP. These modules are separated into the subdirectories "clock", "console", "include", "shmsupp", "startup", and "timer".

2.1.1.3 Test Suite Source Directory

The "tests" directory structure for the **C** implementaion is as follows:



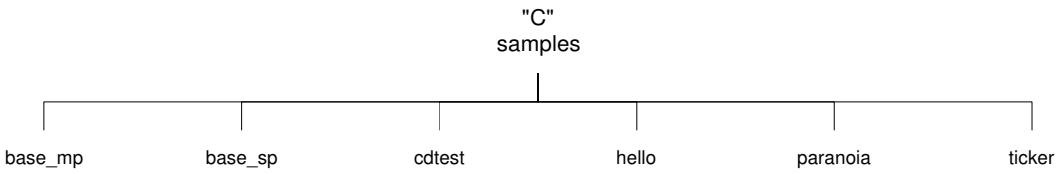
This directory provides the entire **RTEMS** Test Suite which includes the single processor tests, multiprocessor tests, timing tests, and sample tests. Additionally, subdirectories for support functions and test related header files are provided.

The "sptests" subdirectory consists of twenty-four tests designed to cover the entire executive code. The "spfatal" test will verify any code associated with the occurrence of a fatal error. Also provided is a test which will determine the size of the **RTEMS** executive.

The multiprocessor test are provided in "mptests". Fouteen tests are provided in this subdirectory which address two node configurations and cover the multiprocessor code found in RTEMS.

Tests that time each directive and a set of critical executive functions are provided in the "tmtests" subdirectory. Within this subdirectory thirty-one tests are provided along with a subdirectory to contain each targets timing results.

The "samples" directory structure for the **C** implementation is as follows:



This directory provides sample application tests which aid in the testing a newly built **RTEMS** environment, a new BSP, or as starting points for the development of an application using the **RTEMS** executive. A Hello World test is provided in the subdirectory "hello". This test is helpful when testing new versions of **RTEMS**, BSPs, or modifications to any portion of the **RTEMS** development environment. The "ticker" subdirectory provides a test for verifiaction of clock chip device drivers of BSPs. A simple single processor test similar to those in the single processor test suite is provided in "base_sp". A simple two node

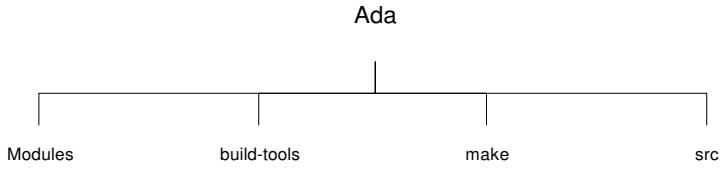
multiprocessor test capable of testing an newly developed MPCI layer is provided in "base_mp". The "cdtest" subdirectory provides a simple C++ application using constructors and destructors. The final sample test is a public domain floating point and math library toolset test is provided in "paranoia".

2.1.2 Ada Suites

The following suites are currently included with the **Ada** implementation of **RTEMS**:

Suite Name	Base Directory
RTEMS Source Code	$RTEMS_SRC_BASE/ada/src/exec
Support Libraries (BSPs, C library, CPU support)	$RTEMS_ROOT/ada/src/lib
Single Processor Tests	$RTEMS_ROOT/ada/src/tests/sptests
Timing Tests	$RTEMS_ROOT/ada/src/tests/tmtests
Multiprocessor Tests	$RTEMS_ROOT/ada/src/tests/mptests
Sample Applications	$RTEMS_ROOT/ada/src/tests/samples
Modules	$RTEMS_ROOT/ada/Modules
RTEMS Build Tools	$RTEMS_SRC_BASE/ada/build_tools
Make Support	$RTEMS_ROOT/ada/make

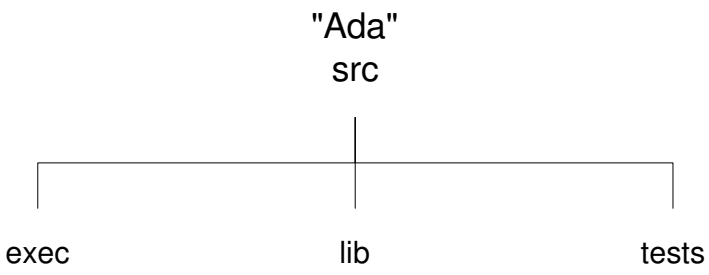
The top level directory structure for the **Ada** implementation of RTEMS is as follows:



This directory contains the subdirectories which contain the entire **Ada** implementation of the **RTEMS** executive. The "Modules" directory contains modules files which correspond to enviroments supported in the current version of **RTEMS**. The "build-tools" directory contains an assortment of support tools for the **RTEMS** development environment. Two subdirectories exist under build-tools" which contain scripts (executables) and source for the support tools. The "make" directory provides two subdirectories which support the Tartan and the GNU Ada Development Environments. Each subdirectory contains

configuration files and subdirectories which provide a robust host and cross-target makefile system supporting the building of the executive for numerous application environments.

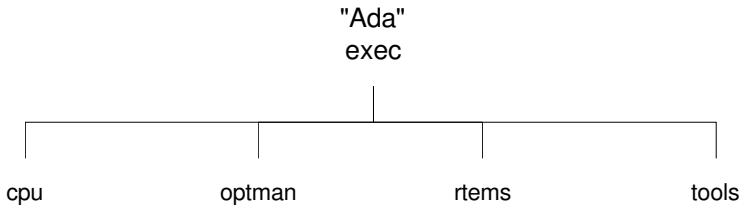
The "src" directory structure for the **Ada** implementation of **RTEMS** is as follows:



This directory contains all source files that comprises the **RTEMS** executive, supported target board support packages, and the **RTEMS** Test Suite.

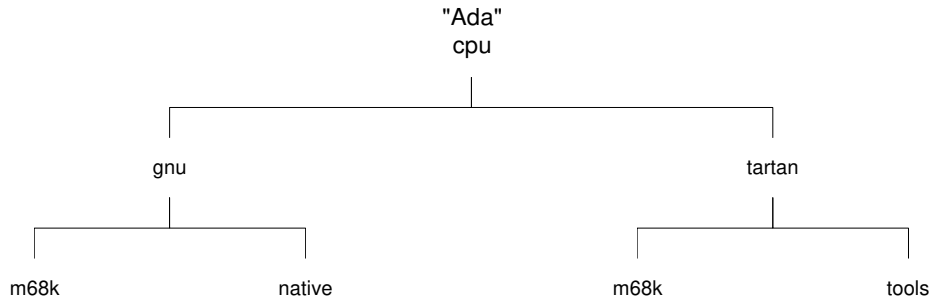
2.1.2.1 Executive Source Directory

The "exec" directory structure for the **Ada** implementation is as follows:



This directory contains a set of subdirectories which contains the source files comprising the executive portion of the RTEMS development environment. At this point the CPU dependent and independent source code files are separated into distinct directory trees. The "rtems" subdirectory contains the Ada language source files for each module found in the executive. Also included in this directory are the subdirectories containing the executive's optional manager capability, and support tool "size_rtems" which determines the actual size of a built configuration of the **RTEMS** executive.

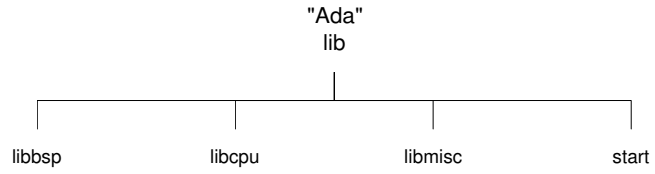
The "cpu" directory structure for the **Ada** implementation of **RTEMS** is as follows:



This directory contains a subdirectory for each target CPU supported by the 3.2.0 release of the RTEMS executive. Each processor directory contains the CPU dependent code necessary to host RTEMS. The "no_cpu" directory provides a starting point for developing a new port to an unsupported processor. The files contained within the "no_cpu" directory may also be used as a reference for the other ports to specific processors.

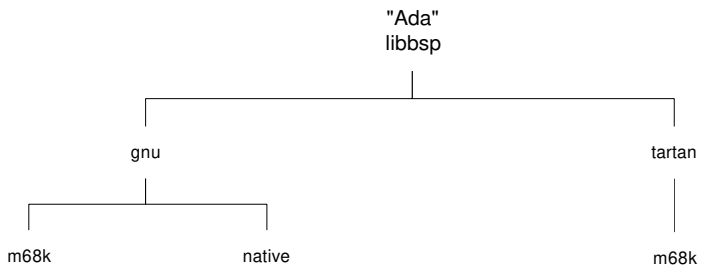
2.1.2.2 Support Library Source Directory

The "lib" directory structure for the **Ada** implementaion is as follows:



This directory contains the source code for libraries which support the development of applications using the RTEMS Development Environment. Board support packages (BSPs), processor environment start up code, and miscellaneous support functions subdirectories are provided.

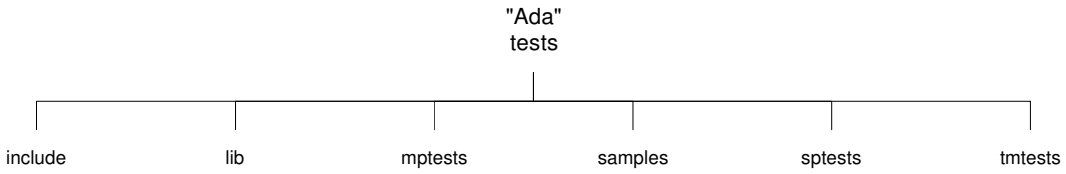
The "libbsp" directory structure for the **Ada** implementaion is as follows:



This directory provides all the BSPs provied with the 3.2.0 version of the **RTEMS** executive. The subdirectories are divided, as discussed previously, based on specific compilers, then based on aprocessor families, then further breaking down into specific target board environments. Each target board subdirectory may provide the implementation of a shared memory driver which supports the multiprocessing protion of the executive, a template stubbed out BSP for testing , a clock driver, timer driver, micellaneous I/O support, and RTEMS test support.

2.1.2.3 Test Suite Source Directory

The "tests" directory structure for the **Ada** implementaion is as follows:



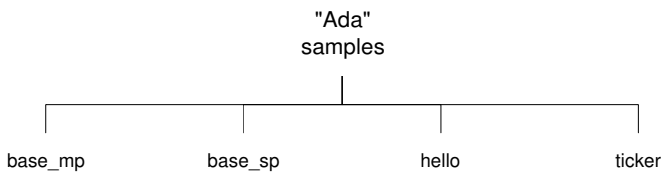
This directory provides the entire **RTEMS** Test Suite which includes the single processor tests, multiprocessor tests, timing tests, and sample tests. Additionally, subdirectories for support functions and test related header files are provided.

The "sptests" subdirectory consists of twenty-four tests designed to cover the entire executive code. The "spfatal" test will verify any code associated with the occurrence of a fatal error. Also provided is a test which will determine the size of the **RTEMS** executive.

The multiprocessor test are provided in "mptests". Fouteen tests are provided in this subdirectory which address two node configurations and cover the multiprocessor code found in RTEMS.

Tests that time each directive and a set of critical executive functions are provided in the "tmtests" subdirectory. Within this subdirectory thirty-one tests are provided along with a subdirectory to contain each targets timing results.

The "samples" directory structure for the **Ada** implementation is as follows:



This directory provides sample application tests which aid in the testing a newly built **RTEMS** environment, a new BSP, or as starting points for the development of an application using the **RTEMS** executive. A Hello World test is provided in the subdirectory "hello". This test is helpful when testing new versions of **RTEMS**, BSPs, or modifications to any portion of the **RTEMS** development environment. The "ticker" subdirectory provides a test for verifiaction of clock chip device drivers of BSPs. A simple single processor test similar to those in the single processor test suite is provided in "base_sp". The final sample test is a simple two node multiprocessor test capable of testing an newly developed MPCI layer is provided in "base_mp".

2.2 Makefiles and Target Directories

The **RTEMS** Makefile tree has been developed to support multiple target projects, effectively simplifying the administration of these development environments. Two types of Makefiles are present in the **RTEMS** directories and differentiated based on whether they control a "directory" or a "leaf" portion of the tree. The directory Makefiles control a source directory with sub-directories by recursing into each sub-directory propagating the make process. The leaf Makefiles control directories that contain source code for libraries, application programs, or any other related source code. Makefiles contained in leaf directories define the rules necessary to build the components of the leaf.

Variant directories are created to contain the binaries associated with a target. These directories assume the name as defined by ($ARCH) and are placed at the appropriate levels within the RTEMS tree by the leaf Makefiles. The Modules files are responsible for setting the current target architecture or it may be set by the user explicitly. Variant directories may be created to contain the binaries for the RTEMS executive and board support package libraries, the RTEMS Test Suites, and for the user application programs.