4

Sample Applications

4.1 Introduction

RTEMS is shipped with the following sample applications:

- *Hello World - C and Ada*
- *Clock Tick - C and Ada*
- *Base Single Processor - C and Ada*
- *Base Multiple Processor - C and Ada*
- *Constructor/Destructor C++ Test - C only if C++ enabled*
- *Paranoia Floating Point Test - C only*

These applications are intended to illustrate the basic format of **RTEMS** single and multiple processor applications. In addition, these relatively simple applications can be used to test locally developed board support packages and device drivers.

The reader should be familar with the terms used and material presented in the **RTEMS C Applications User's Guide** or the **RTEMS Ada Applications User's Guide**.

4.2 Hello World

This sample application is in the following directory:

```
$RTEMS_SRC_BASE/tests/samples/hello
```

It provides a rudimentary test of the BSP start up code and the console output routine. The C version of this sample application uses the printf function from the **RTEMS Standard C Library** to output messages. The Ada version of this sample use the TEXT_IO package to output the hello messages. The following messages are printed:

```
*** HELLO WORLD TEST ***
Hello World
*** END OF HELLO WORLD TEST ***
```

These messages are printed from the application's single initialization task. If the above messages are not printed correctly, then either the BSP start up code or the console output routine is not operating properly.

4.3 Clock Tick

This sample application is in the following directory:

```
$RTEMS_SRC_BASE/tests/samples/ticker
```

This application is designed as a simple test of the clock tick device driver. In addition, this application also tests the printf function from the **RTEMS Standard C Library** by using it to output the following messages:

```
 *** CLOCK TICK TEST ***
TA1 - tm_get - 09:00:00   12/31/1988
TA2 - tm_get - 09:00:00   12/31/1988
TA3 - tm_get - 09:00:00   12/31/1988
TA1 - tm_get - 09:00:05   12/31/1988
TA1 - tm_get - 09:00:10   12/31/1988
TA2 - tm_get - 09:00:10   12/31/1988
TA1 - tm_get - 09:00:15   12/31/1988
TA3 - tm_get - 09:00:15   12/31/1988
TA1 - tm_get - 09:00:20   12/31/1988
TA2 - tm_get - 09:00:20   12/31/1988
TA1 - tm_get - 09:00:25   12/31/1988
TA1 - tm_get - 09:00:30   12/31/1988
TA2 - tm_get - 09:00:30   12/31/1988
TA3 - tm_get - 09:00:30   12/31/1988
*** END OF CLOCK TICK TEST ***
```

When using the Ada version the TEXT_IO package is utilized to output the following messages:

```
 *** CLOCK TICK TEST ***
TA1 - tm_get -  9: 0: 0   12/31/1988
TA2 - tm_get -  9: 0: 0   12/31/1988
TA3 - tm_get -  9: 0: 0   12/31/1988
TA1 - tm_get -  9: 0: 5   12/31/1988
TA1 - tm_get -  9: 0:10   12/31/1988
TA2 - tm_get -  9: 0:10   12/31/1988
TA1 - tm_get -  9: 0:15   12/31/1988
TA3 - tm_get -  9: 0:15   12/31/1988
TA1 - tm_get -  9: 0:20   12/31/1988
TA2 - tm_get -  9: 0:20   12/31/1988
TA1 - tm_get -  9: 0:25   12/31/1988
TA1 - tm_get -  9: 0:30   12/31/1988
TA2 - tm_get -  9: 0:30   12/31/1988
TA3 - tm_get -  9: 0:30   12/31/1988
*** END OF CLOCK TICK TEST ***
```

The clock tick sample application utilizes a single initialization task and three copies of the single application task. The initialization task prints the test herald, sets the time and date, and creates and starts the three application tasks before deleting itself. The three application tasks generate the rest of the output. Every five seconds, one or more of the tasks will print the current time obtained via the tm_get directive. The first task, TA1, executes every five seconds, the second task, TA2, every ten seconds, and the third task, TA3, every fifteen seconds. If the time printed does not match the above output, then the clock device driver is not operating properly.

4.4 Base Single Processor Application

This sample application is in the following directory:

```
$RTEMS_SRC_BASE/tests/samples/base_sp
```

It provides a framework from which a single processor **RTEMS** application can be developed. The use of the task argument is illustrated. This sample application uses the printf function from the **RTEMS Standard C Library** or TEXT_IO functions when using the Ada version to output the following messages:

```
*** SAMPLE SINGLE PROCESSOR APPLICATION ***
Creating and starting an application task
Application task was invoked with argument (0) and has id of
0x10002
*** END OF SAMPLE SINGLE PROCESSOR APPLICATION ***
```

The first two messages are printed from the application's single initialization task. The final messages are printed from the single application task.

4.5 Base Multiple Processor Application

This sample application is in the following directory:

```
$RTEMS_SRC_BASE/tests/samples/base_mp
```

It provides a framework from which a multiprocessor **RTEMS** application can be developed. This directory has a subdirectory for each node in the multiprocessor system. The task argument is used to distinguish the node on which the application task is executed. The first node will print the following messages:

```
*** SAMPLE MULTIPROCESSOR APPLICATION ***
Creating and starting an application task
This task was invoked with the node argument (1)
This task has the id of 0x10002
*** END OF SAMPLE MULTIPROCESSOR APPLICATION ***
```

The second node will print the following messages:

```
*** SAMPLE MULTIPROCESSOR APPLICATION ***
Creating and starting an application task
This task was invoked with the node argument (2)
This task has the id of 0x20002
*** END OF SAMPLE MULTIPROCESSOR APPLICATION ***
```

The herald is printed from the application's single initialization task on each node. The final messages are printed from the single application task on each node.

In this sample application, all source code is shared between the nodes except for the node dependent configuration files. These files contains the definition of the node number used in the intiialization of the **RTEMS Multiprocessor Configuration Table.** This file is not shared because the node number field in the **RTEMS Multiprocessor Configuration Table** must be unique on each node.

4.6 Constructor/Destructor C++ Application

This sample application is in the following directory:

```
$RTEMS_SRC_BASE/tests/samples/cdtest
```

This sample application demostrates that **RTEMS** is compatible with C++ applications. It uses contructors, destructors, and I/O stream output in testing these various capabilities. The board support package responsible for this application must support a C++ environment.

This sample application uses the printf function from the **RTEMS Standard C Library** to output the following messages:

```
Hey I'M in base class constructor number 1 for 0x400010cc.
Hey I'M in base class constructor number 2 for 0x400010d4.
Hey I'M in derived class constructor number 3 for 0x400010d4.

*** CONSTRUCTOR/DESTRUCTOR TEST ***
Hey I'M in base class constructor number 4 for 0x4009ee08.
Hey I'M in base class constructor number 5 for 0x4009ee10.
Hey I'M in base class constructor number 6 for 0x4009ee18.
Hey I'M in base class constructor number 7 for 0x4009ee20.
Hey I'M in derived class constructor number 8 for 0x4009ee20.
Testing a C++ I/O stream
Hey I'M in derived class constructor number 8 for 0x4009ee20.
Derived class - Instantiation order 8
Hey I'M in base class constructor number 7 for 0x4009ee20.
Instantuation order 8
Hey I'M in base class constructor number 6 for 0x4009ee18.
Instantuation order 6
Hey I'M in base class constructor number 5 for 0x4009ee10.
Instantuation order 5
Hey I'M in base class constructor number 4 for 0x4009ee08.
Instantuation order 5
*** END OF CONSTRUCTOR/DESTRUCTOR TEST ***

Hey I'M in base class constructor number 3 for 0x400010d4.
Hey I'M in base class constructor number 2 for 0x400010d4.
Hey I'M in base class constructor number 1 for 0x400010cc.
```

4.7 Paranoia Floating Point Application

This sample application is in the following directory:

```
$RTEMS_SRC_BASE/tests/samples/paranoia
```

This sample application uses a public domain floating point and math library test to verify these capabilities of the **RTEMS** executive. Deviations between actual and expected results are reported to the screen. This is a very extensive test which tests all mathematical and number conversion functions. Paranoia is also very large and requires a long period of time to run.