RTI Connext Core Libraries

Getting Started Guide Addendum for Embedded Systems Version 7.2.0



© 2012-2022 Real-Time Innovations, Inc. All rights reserved. October 2023.

Trademarks

RTI, Real-Time Innovations, Connext, Connext Drive, NDDS, the RTI logo, 1RTI and the phrase, "Your Systems. Working as one." are registered trademarks, trademarks or service marks of Real-Time Innovations, Inc. All other trademarks belong to their respective owners.

Copy and Use Restrictions

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form (including electronic, mechanical, photocopy, and facsimile) without the prior written permission of Real-Time Innovations, Inc. The software described in this document is furnished solely under and subject to RTI's standard terms and conditions available at https://www.rti.com/terms and in accordance with your License Acknowledgement Certificate (LAC) and Maintenance and Support Certificate (MSC), except to the extent otherwise accepted in writing by a corporate officer of RTI.

Third-Party Software

RTI software may contain independent, third-party software or code that are subject to third-party license terms and conditions, including open source license terms and conditions. Copies of applicable third-party licenses and notices are located at <u>community.rti.com/documentation</u>. IT IS YOUR RESPONSIBILITY TO ENSURE THAT YOUR USE OF THIRD-PARTY SOFTWARE COMPLIES WITH THE CORRESPONDING THIRD-PARTY LICENSE TERMS AND CONDITIONS.

Notices

Deprecations and Removals

Any deprecations or removals noted in this document serve as notice under the Real-Time Innovations, Inc. Maintenance Policy #4220 and/or any other agreements by and between RTI and customer regarding maintenance and support of RTI's software.

Deprecated means that the item is still supported in the release, but will be removed in a future release. *Removed* means that the item is discontinued or no longer supported. By specifying that an item is deprecated in a release, RTI hereby provides customer notice that RTI reserves the right after one year from the date of such release and, with or without further notice, to immediately terminate maintenance (including without limitation, providing updates and upgrades) for the item, and no longer support the item, in a future release.

Technical Support

Real-Time Innovations, Inc. 232 E. Java Drive Sunnyvale, CA 94089 Phone: (408) 990-7444 Email: <u>support@rti.com</u> Website: <u>https://support.rti.com/</u>

Contents

Chapter 1 Addendum for Embedded Platforms	1
Chapter 2 Getting Started on QNX Embedded Systems	
2.1 Building and Running a Hello World Example	2
Chapter 3 Getting Started on VxWorks Systems	
3.1 Building the VSB	6
3.2 Building the Kernel	8
3.3 Building and Running a Hello World Example	12
3.3.1 Generate Example Code and Makefile with rtiddsgen	12
3.3.2 Building and Running an Application as a Kernel Task	13
3.3.3 Building and Running an Application as a Real-Time Process	23
3.4 Using DDS Ping and Spy	28

Chapter 1 Addendum for Embedded Platforms

In addition to enterprise-class platforms like Microsoft Windows and Linux, *RTI*® *Connext*® supports a wide range of embedded platforms. This document is especially for users of those platforms. It describes how to configure some of the most popular embedded systems for use with *Connext* and to get up and running as quickly as possible. The code examples covered in this document can be generated for your platform(s) using *RTI Code Generator (rtiddsgen)*, which accompanies *Connext*.

This document assumes at least minimal knowledge with the platforms it describes and is not a substitute for the documentation from the vendors of those platforms. For further instruction on the general operation of your embedded system, please consult the product documentation for your board and operating system.

Chapter 2 Getting Started on QNX Embedded Systems

This document provides instructions on building and running *Connext* applications on embedded systems such as QNX® systems. It will guide you through the process of generating, compiling, and running a Hello World application on an embedded QNX system by expanding on Hands-On 1 of *Introduction to Publish/Subscribe*, in the <u>RTI Connext Getting Started Guide</u>. Please read the following alongside that section.

In the following steps:

- All commands must be executed in a command shell that has all the required environment variables. For details, see Set Up Environment Variables (rtisetenv), in "Hands-On 1" of *Introduction to Publish/Subscribe*, in the <u>RTI Connext Getting Started Guide</u>.
- You need to know the name of your target architecture (look in your NDDSHOME/lib directory). Use it in place of *<architecture>* in the example commands. For example, your architecture might be 'armv8QNX7.1qcc_gpp8.3.0'.
- We assume that you have **make** installed. If you have **make**, you can use the generated makefile to compile. If you do not have **make**, use your normal compilation process. (Note: the generated makefile assumes the correct version of the compiler is already in your path and that **NDDSHOME** is set.)

2.1 Building and Running a Hello World Example

This section describes the basic steps for building and running an *rtiddsgen*-generated example on an embedded target such as QNX.

- 1. Create a directory to work in. In this example, we use a directory called myhello.
- 2. In the **myhello** directory, create a file called **HelloWorld.idl** that contains a user-defined data type:

```
struct HelloWorld {
    string<128> msg;
};
```

3. Use the *rtiddsgen* utility to generate sample code and a makefile as shown below. Substitute <architecture> with your target architecture string, such as armv8QNX7.1qcc gpp8.3.0.

For C++:

rtiddsgen -language C++ -example <architecture> HelloWorld.idl

Edit the example code to add this line:

sprintf(instance->msg, "Hello World! (%d)", count);

It should look like this:

```
for (count=0; (sample_count == 0) || (count < sample_count); ++count) {
    printf("Writing HelloWorld, count %d\n", count);
    /* Modify the data to be written here */
    sprintf(instance->msg, "Hello World! (%d)", count);
    /* Write data */
    retcode = HelloWorldDataWriter_write(
        HelloWorld_writer, instance, &instance_handle);
    if (retcode != DDS_RETCODE_OK) {
        fprintf(stderr, "write error %d\n", retcode);
    }
    NDDS_Utility_sleep(&send_period);
}
```

4. With the NDDSHOME environment variable set, build the Publisher and Subscriber modules using the generated makefile.

make -f makefile HelloWorld <architecture>

For details on setting up the NDDSHOME environment variable, see Set Up Environment Variables (rtisetenv), in "Hands-On 1" of *Introduction to Publish/Subscribe*, in the <u>RTI Connext Getting Started Guide</u>.

After compiling, find the application executables in myhello/objs/<architecture>.

 Connect to the QNX target (using ssh, for example) and start the subscriber application, HelloWorld_subscriber.

HelloWorld_subscriber

In this shell, you should see that the subscriber is waking up every 4 seconds to print a message. Here is a C++ example: No data after 1 second No data after 1 second No data after 1 second

6. Connect to the QNX target and start the publisher application, HelloWorld_publisher.

HelloWorld_publisher

In this second (publishing) shell, you should see:

Writing HelloWorld, count 0 Writing HelloWorld, count 1 Writing HelloWorld, count 2

7. Look back in the first (subscribing) shell. You should see that the subscriber is now receiving messages from the publisher.

For example, in C++:

Received data msg: "Hello World! (0)" Received data msg: "Hello World! (1)" Received data msg: "Hello World! (2)"

Chapter 3 Getting Started on VxWorks Systems

This section provides simple instructions to configure a kernel and run *Connext* applications on systems based on VxWorks 7, including VxWorks 22.09.

Please refer to the documentation provided by Wind River Systems for more information on this operating system.

This chapter will guide you through the process of generating, compiling, and running a Hello World application on a VxWorks 7 system by expanding on the <u>VxWorks section of the RTI</u> <u>Connext Core Libraries Platform Notes</u>; please read the following alongside that section. This chapter uses VxWorks 7 as an example. The steps for VxWorks 22.09 should be similar, since it is based on VxWorks 7.

The first two sections describe how to build a VxWorks Source Build (VSB) and the kernel:

- 3.1 Building the VSB on the next page
- 3.2 Building the Kernel on page 8

The next section guides you through the steps to generate, modify, build, and run the provided example HelloWorld application on a VxWorks target:

• 3.3 Building and Running a Hello World Example on page 12

For tips on using RTI DDS Ping and Spy, see 3.4 Using DDS Ping and Spy on page 28.

3.1 Building the VSB

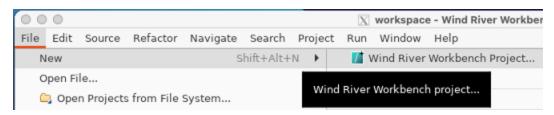
This section explains how to build a VxWorks Source Build (VSB), which is required in order to build your own kernels and applications with VxWorks 7.

The following steps use the VSB defaults. For further information and special customizations, please refer to Wind River's documentation: https://docs.windriver.com/bundle/Configuration_and_Build_Guide_Edition_9_1/page/1597954.html

https://docs.windriver.com/bundle/Configuration and Build Guide Edition 9 1/page/1597954.html

Before you start, you should be familiar with your hardware, as you will need to select a BSP and other hardware-specific settings. This document uses an Intel BSP as an example.

1. Launch Workbench and select File, New, Wind River Workbench Project.



2. For the Build type, select VxWorks Source Build.

000	X New Wind River Workbench Project
Build Type Select the build	type for the project.
Based on Categ	ory All 👻
Build type:	VxWorks Source Build
Description:	Creates a VxWorks source build project, which allows configuring and building operating system layers.
Setup informati	
Resulting proje	ct type: VSB Project
?	< Back Next > Cancel Finish

- 3. Set your project name and click Next.
- 4. Configure your VSB. Set your BSP, the CPU, addressing mode, compiler, SMP, etc., according to your platform. When you are done, click **Finish**.

	X New Va	Works Source Build Project		
Project Setup Base the new pr CPU type.	oject either on an existi	ng configuration, a board support package, or a		
Setup the project	t			
Based on a V	Works board support pa	ickage 👻		
BSP:	itl_generic_3_0_0_0	Show latest versions only		
Active CPU:	CORE			
Baseline:	21.11			
Address mode:	LP64 64-bit libraries	•		
Options:	Branch:	vxworks-21.11		
	Toolchain:	llvm 12.0.1.1		
	Linker to use:	LD (GNU Linker)		
	Floating point setting:	hard 👻		
	Endian setting:	little 👻		
	Processor mode:	SMP support in Libraries 👻		
	Debug mode:	Off, and normal compiler optimizations enabled $aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa$		
	IP version setting:	IPv6 and IPv4 enabled libraries		
	VSB profile:	None 👻		
	ces to project			
Setup informatio Base location: i	n itl_generic_3_0_0_0			
?	< Bac	k Next > Cancel Finish		

5. After you finish, build the VSB as you would any other project.

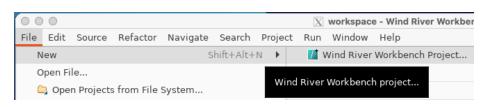
3.2 Building the Kernel

This section explains how to build a kernel capable of loading *Connext* libraries. *Connext* libraries require that certain components are added to the default list in the VxWorks kernel, as outlined in the following steps.

Before you start, you should be familiar with building and deploying a default working kernel on your target.

Note: The following steps might vary slightly depending on your chosen version of VxWorks.

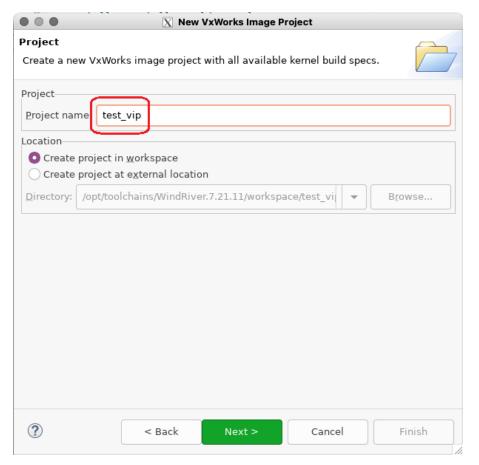
1. Launch Workbench and select File, New, Wind River Workbench Project.



2. When prompted for a **Build type**, select **VxWorks Source Build** (this may be **Kernel Image** or **VxWorks Image** depending on your version of VxWorks); click **Next.**

000	🔀 New Wind River Workbench Project	
Build Type Select the build ty	e for the project.	7
Based on Category	All	
Build type:	VxWorks Source Build)
Description:	Creates a VxWorks source build project, which allows configuring and building operating system layers.	
Setup information-		1
Resulting project	/pe: VSB Project	
?	< Back Next > Cancel Finish	

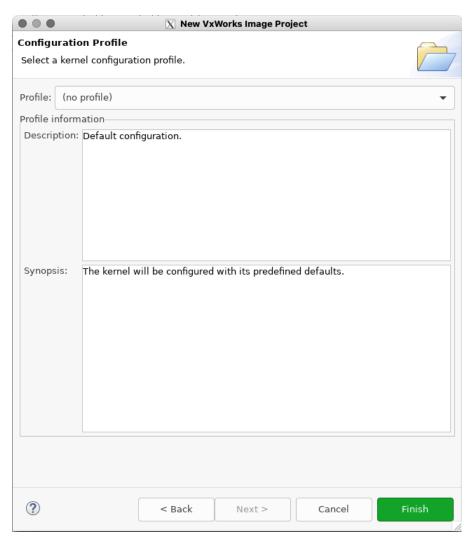
3. Give your project a name; click Next.



- 4. In Project Setup:
 - a. For the **Based on** field, choose a source build project.
 - b. For the **Project**, choose the VSB you created and built in 3.1 Building the VSB on page 6. The BSP, SMP support, and other options will be correctly populated from the VSB configuration.
 - c. For the Tool chain option, select LLVM.
 - d. In **Options**, select **SMP support in kernel** if your BSP supports it and you want to enable symmetric multi-processing capability in the kernel.
 - e. Select **IPv6 enabled kernel libraries** if your architecture supports IPv6 (See the <u>VxWorks</u> section of the <u>RTI Connext Core Libraries Platform Notes</u> to check if your architecture supports IPv6); click **Next**.

		🔀 New VxWorks Image Project	
roject Setu	up		
		an existing project, or on a source build project ge and tool chain selections.	
etup the pro	oject		
Based on a	a source build proj	ect	
Project:	test_vsb	•	Browse.
BSP:	itl_generic_3_0_0	_0	
Tool chain:	llvm		
Options:	Branch:	vxworks-21.11	Ŧ
	Address mode:	LP64 64-bit libraries	•
	Processor mode:	SMP support in Libraries	•
	Debug mode:	Off, and normal compiler optimizations enabled	-
SP validatio	on test suite		
🗌 Add supp	port to project		Options.
etup inform	ation		
etup iniorm			

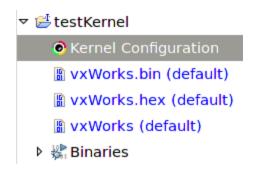
5. Optionally, select a configuration profile from the drop-down menu.



6. Leave everything else at its default setting. Click Finish.

Your project will be created at this time.

7. From the Project Explorer, open Kernel Configuration.



- 8. Add Operating System Components, Kernel Components, _thread variables support.
- 9. Make sure you have the following components enabled: INCLUDE_TIMESTAMP, INCLUDE_SHARED_DATA, INCLUDE_TLS.

Note: If you are unwilling or unable to build shared-memory support into your kernel, see the VxWorks section of the RTI Connext Core Libraries Platform Notes.

- If you plan to use any *Connext* C++ API, you will need to include the FOLDER_CPLUS section in your kernel (the underlying kernel components may vary depending on the VxWorks version). This includes Traditional and Modern C++ APIs and Request/Reply C++ APIs.
- 11. If you want support for RTP shared libraries, you need to add the component **INCLUDE_SHL**. Note that shared libraries are not supported in all VxWorks architectures.
- 12. If you plan on accessing your target via the network, you may need the following modules:
 - Telnet Server (under Network Components, Applications, Telnet Components)

This will allow you to telnet into the target.

• NFS client all (under Operating System Components, IO System Components, NFS components)

This will allow you to see networked file systems from the target (contact your system administrator to find out if you have them set up).

- 13. If you are running applications in RTP mode, you may increase Operating System components, Real Time Processes components, Number of entries in an RTP fd table from the default value of 20 to a higher value such as 256. This will enable you to open more sockets from an RTP application.
- 14. Compile the Kernel by right-clicking the project and selecting Build Project.

The Kernel and associated symbol file will be found in <your project directory>/default/.

3.3 Building and Running a Hello World Example

This section will guide you through the steps required to successfully run an *rtiddsgen*-generated example application on a VxWorks 7 target using kernel mode or RTP mode.

3.3.1 Generate Example Code and Makefile with rtiddsgen

To create the example applications:

1. Set up the environment on your development machine: set the NDDSHOME environment variable and update your PATH as described in Set Up Environment Variables (rtisetenv), in "Hands-On 1" of *Introduction to Publish/Subscribe*, in the <u>RTI Connext Getting Started Guide</u>.

- 2. Create a directory to work in. In this example, we use a directory called myhello.
- 3. In the **myhello** directory, create a file called **HelloWorld.idl** that contains a user-defined data type:

```
struct HelloWorld
{
    string<128> msg;
};
```

4. Use *RTI Code Generator (rtiddsgen)* to generate sample code and a makefile. Choose either C or C++.

Note: The architecture names for Kernel Mode and RTP Mode are different.

For C:

rtiddsgen -language C -example <architecture> HelloWorld.idl

For C++:

rtiddsgen -language C++ -example <architecture> HelloWorld.idl

Edit the generated example code as described in Hands-On 1 of *Introduction to Publish/Subscribe*, in the <u>RTI Connext Getting Started Guide</u>.

3.3.2 Building and Running an Application as a Kernel Task

There are two ways to build and run your Connext application:

- 3.3.2.1 Using the Command Line below
- 3.3.2.2 Using Workbench on the next page

3.3.2.1 Using the Command Line

1. Set up your environment with the **wrenv.sh** script or **wrenv.bat** batch file in the VxWorks base directory. Execute the script with the **-p** parameter. For example:

wrenv.sh -p vxworks

- Set the NDDSHOME environment variable as described in Set Up Environment Variables (rtisetenv), in "Hands-On 1" of *Introduction to Publish/Subscribe*, in the <u>RTI Connext Getting Started Guide</u>.
- 3. Build the Publisher and Subscriber modules using the generated makefile. You may have to modify the HOST_TYPE, compiler and linker paths to match your development setup.
- 4. To use dynamic linking, remove the *Connext* libraries from the link objects in the generated makefile.

(Note: steps 5-7 can be replaced by establishing a telnet connection to the VxWorks target. In that case, Workbench does not need to be used and both the Host Shell and Target Console will be redirected to the telnet connection. Once in the C interpreter (you will see the prompt '->' in the shell) you can type **cmd** and then **help** for more information on how to load and run applications on your target.)

- 5. Launch Workbench.
- Make sure your target is running VxWorks and is added to the Remote Systems panel. (To add a new target, click the New Connection button on the Remote System panel, select Wind River VxWorks 7 Target Server Connection, click Next, enter the Target name or address, and click Finish).
- 7. Connect to the target and open a host shell by right-clicking the connected target in the **Target Tools** sub-menu.
- 8. In the shell:

If you are using static linking: Load the .so file produced by the build:

```
>cd "directory"
>ld 0 < HelloWorld subscriber.so</pre>
```

(Where 'directory' refers to the location of the generated object files.) If you are using dynamic linking: load the libraries first, in this order: **libnddscore.so**, **libnddsc.so**, **libnddscpp.so**; *then* load the **.so** file produced by the build.

Note: If you are statically linking, and you try to load both the publisher and subscriber into the kernel, you will run into duplication of symbols due to the *Connext* libraries being statically linked in both modules. To overcome that situation, see the "Notes for VxWorks 7 Platforms" section in the *RTI Connext Core Libraries Platform Notes*, for an explanation about how to create a single Downloadable Kernel Module (DKM) containing both applications.

9. Run the run_subscriber_application or run_publisher_application function. For example:

```
>taskSpawn "sub", 255, <floating_point_option>, 150000, run_subscriber_application,
38, 10
```

Where <floating_point_option> is a numeric value that varies depending on the hardware. See <u>Enabling Floating Point Coprocessor in Kernel Tasks, in the VxWorks chapter of the RTI Con-</u> next Core Libraries Platform Notes.

In this example, 38 is the domain ID and 10 is the number of samples.

3.3.2.2 Using Workbench

Note: The following steps might vary slightly depending on your chosen version of VxWorks.

- 1. Start Workbench.
- 2. Select File, New, Wind River Workbench Project.

😣 🖃 🗉 🛛 Basic Device Development V	Vind River Workbench	
<u>File</u> <u>E</u> dit <u>S</u> ource Refac <u>t</u> or <u>N</u> avigate	Se <u>a</u> rch <u>P</u> roject <u>R</u> un	<u>W</u> indow <u>H</u> elp
<u>N</u> ew	Shift+Alt+N ▶	Wind River Workbench Project
Open File <u>.</u>		P <u>r</u> oject
Close	Ctrl+W	Build Target
Save	Ctrl+S	Folder
Refresh	F5	File
Convert Line Delimiters To	•	File from Template
-	Challen	E <u>x</u> ample
Print	Ctrl+P	Other Ctrl+N

3. Select the desired Target operating system; click Next.

😣 🗉 New Wind River Wo	rkbench Proje	ct		
Target Operating System	n			
Select the target operat	ing system for	the project.		
Target operating system		VxWorks 6.9		
	Host OS (Na	ative Developmen	nt)	
?	< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

4. When prompted to choose a Build type, select Downloadable Kernel Module; click Next.

😣 🗊 🛛 New W	ind River Workben	ch Project		
Build Type Select the bu	uild type for the pr	oject.		
Build type: Description:	Boot Loader / BSR Downloadable Ker Kernel Libraries ROMFS File System Real Time Process Relinkable Kernel Shared User Libra Static Kernel Libra	rnel Module m s Application Module ary ary		
Setup inforn Resulting pr Uses build t	System Image User-Defined None	, 		
?	< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

5. Give your project a name; click Next.

😣 🗉 New Wind	River Workben	ch Project		
Project Create a new pr	oject with the s	specified data.		
<u>P</u> roject name: t	estApplication			
Location				
• Create proje	ct in <u>w</u> orkspace	2		
○ Create proje	ct at e <u>x</u> ternal l	ocation		
○ Create proje	ct in workspace	e with <u>c</u> ontent a	t external locati	on
Directory: /hom	ne/local/WindRi	ver/workspace/t	testApplicat 🔻	Browse
			<u> </u>	4
?	< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

6. Leave everything else at its default setting; click Finish.

Your project will be created at this time.

😣 🗇 🗊 🛛 Basic Device Development Wind River Workbeng	:h	
<u>File Edit Source Refactor N</u> avigate Search Project B	3un <u>W</u> indow <u>H</u> elp	
│ M 🗂 - 💷 👒 🚔 │ 🖬 🗯 🗱 🔋 │ 🎭 - O - │ 🖋 │ 🏢 🏢	劉・智・登 令・令・ 叱	😭 📷 Basic De
ြဲ Project Explorer 🛛 📄 😫 ညိုး 🔛 🖉 🖓 🖓 🖓 🖓	🛿 🚱 Getting Started 🛛 📄 creation.log 📄 🗟 usrAppInit.c 📄 creation.log 📄 creation.log	
NetworkImage (vxworks-cert-6.6.4.1 [not installed] Im	a 🟠 🗇 🗘	An outline is not available.
ti-slo-6.9.4.2-sbc8641d-v1 (Wind River VxWorks 6.9 In		
Samsung-iso-v1 (wrlinux-4 [not installed] Platform Proj	e 🚺 Wind River Workbench 3.3 🛨 What's New	
Samsung-iso-v2 (wrlinux-4 [not installed] Platform Proj		
Samsung-iso-v3 (wrlinux-4 [not installed] Platform Proj	e A A A	
Samsung-v1 (wrlinux-4 [not installed] Platform Project)		
Est (vxworks-cert-6.6.4.1 [not installed] Image Project		
Test_Vx6ImageProject (vxworks-cert-6.6.4.1 [not instal	Getting Started	
 # testApplication (Wind River VxWorks 6.9 Downloadable) 		
Build Targets (SIMLINUXdiab - debug)	Create	
Sewind River Launches		
▶ 🔊 Includes	Debug	
EstKernel (Wind River VxWorks 6.9 Image Project)	Analyze	
Scalibur1700-micro-cert-v1 (vxworks-cert-6.6.4.1 [not		
Calibur1700-Test_Image (vxworks-cert-6.6.4.1 [not instant)	Set-up & Share	
	Show Welcome Splash	
A Remote Systems 🕱		
Wind River Registries	🔯 Help 🖳 Build Console 🕴 🖉 Tasks 🚼 Problems 🔗 Search 🛛 🗯 📄 🔍 💌	
Local Files		<u>^</u>
🛱 Local Shells		
Image: Signature State Stat		
		Ŧ
L		•
📑 😢 testApplication (Wind River VxWorks 6.9 Down	loadable Kernel Module Project) 108M of 157M 👔	

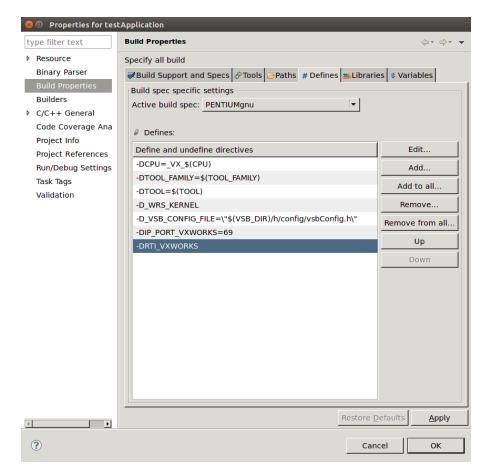
- 7. Copy the source and header files generated by *rtiddsgen* in 3.3.1 Generate Example Code and Makefile with rtiddsgen on page 12 into the project directory.
- 8. View the added files by right-clicking on the project in Project Explorer, then selecting **Refresh** to see the files.

😣 🔵 💿 Basic Device Development Wind River Workbenc	1	
<u>File Edit Source Refactor Navigate Search Project R</u>		
M □ • · · · · · · · · · · · · · · · · · ·	劉・堂・谷 今・今・ 	😭 🔤 Basic De
🕒 Project Explorer 🛛 📄 😫 🗞 - 🎽 🖬	🚱 Getting Started 🕱 📄 creation.log 📄 usrAppInit.c 📄 creation.log 📄 creation.log 🦳 🗖 🗖	🗄 Outline 🛛 🦳 🗖 🗖
▶ ² samsung-iso-v2 (wrlinux-4 [not installed] Platform Pr		An outline is not available.
Samsung-iso-v3 (wrlinux-4 [not installed] Platform Press		
samsung-v1 (wrlinux-4 [not installed] Platform Projec	Wind River Workbench 3.3 🖈 What's New	
test (vxworks-cert-6.6.4.1 [not installed] Image Proje		
East_Vx6ImageProject (vxworks-cert-6.6.4.1 [not inst	AAA	
🝷 🕮 testApplication (Wind River VxWorks 6.9 Downloadat		
Build Targets (PPC32gnu_SMP - debug)	Catting Started	
🕼 Wind River Launches	Getting Started	
Includes	Create	
HelloWorld_publisher.cxx	Create	
HelloWorld_subscriber.cxx	Debug	
le HelloWorld.cxx		
🖻 HelloWorld.h	Analyze	
le HelloWorldPlugin.cxx		
A HelloWorldPlugin.h	Set-up & Share	
A HelloWorldSupport.cxx	Show Welcome Splach 🖻	
HelloWorldSupport.h		
HelloWorld.idl		
makefile_HelloWorld_ppc604Vx6.9gcc4.3.3		
☑ USER_QOS_PROFILES.xml		
EstKernel (Wind River VxWorks 6.9 Image Project)	the second se	
▶ ﷺ xcalibur1700-micro-cert-v1 (vxworks-cert-6.6.4.1 [no ♥		
📲 Remote Systems 🛱 🗖 🗖		
🚽 🗤 🗤 😻 🕲 (🗢 🔶 🖉 🔁 🤹		
🕨 🗳 Local		
LinuxHost_local (Host OS (Native Development))		
ト 楊 vxsim0 (Wind River VxWorks 6.9)		
Karring (Wind River VxWorks 6.9)		
	(buttle (2 puille Coursele m) (2 public publ	& & & ↓ <u> </u>
	🔭 Help 💷 Build Console 😫 🖉 Tasks 🖺 Problems 🔗 Search 🛛 😫 🖶 💽	
		_
🕐 ڬ testApplication (Wind River VxWorks 6.9 Downle	badable Kernel Module Project) 31M of 149M 🗓	

- 9. Open the project Properties by right-clicking on the project in Project Explorer and selecting **Properties**.
- 10. In the dialog box that appears, select **Build Properties** in the navigation pane on the left.
- 11. In the **Build Support and Specs** tab, select the desired build spec from the **Active build spec** drop-down menu; click **Apply** to save the changes.

😣 🗉 Properties for test	tApplication	
type filter text	Build Properties	(⇒ + ⇒ + +
 Resource Binary Parser Build Properties Builders C/C++ General Code Coverage Ana Project Info 	Specify all build Image: Specify all Support and Specs Image: Specify all Support Image: Specify all Specify all Specify all Specify and Specify all	ariables
Project References Run/Debug Settings Task Tags		Variables
Validation	Available and enabled build specs: ARMARCH4diab ARMARCH4diabbe ARMARCH4diabbe ARMARCH4gnu ARMARCH4gnube ARMARCH5diab ARMARCH5diab ARMARCH5diabbe ARMARCH5diabbe ARMARCH5diabbe ARMARCH5diabbe ARMARCH5gnu ARMARCH6diab ARMARCH6diab ARMARCH6diabbe Debuild spec: PPC604gnu_SMP V Debug mode	Enable All Disable All New Import Copy Delete
Image: A the second	Restore Defaults	<u>A</u> pply
?	Cancel	ОК

12. In the **Build Macros** or **Defines** tab, add **-DRTI_VXWORKS** to DEFINES in the Build macro definitions; click **Apply** to save the changes.



13. In the Variables tab, add to LIBPATH:

-L/(NDDSHOME)/lib/<architecture>

If you are using *static* linking, add to LIBS:

-Inddscppz -Inddscz -Inddscorez (in that order)

If you are using *dynamic* linking, add to LIBS:

-Inddscpp -Inddsc -Inddscore (in that order)

Click **Apply** to save the changes.

Properties for test	Application		
type filter text	Build Properties		(□ + □) + - +
Resource	Specify all build		
Binary Parser	Build Support and Spe	cs 🖗 Tools 🗁 Paths 🗰 Defines 🛋 Libraries 💲 Va	riables
Build Properties	Name	Value	New
Builders	PROJECT TYPE	DKM	
C/C++ General	DEFINES		Edit
Code Coverage Ana	EXPAND_DBG	0	a
Project Info	EXTAND_DDG	0	Rename
Project References			Сору
Run/Debug Settings			
Task Tags			Delete
Validation			
	Build spec specific settin	-	
	Active build spec: PEN	TIUMgnu 🔽	▲ _{New}
	Active build spec: PEN Name (Common)	TIUMgnu ▼ Value (Build spec specific)	New
	Active build spec: PEN Name (Common) TOOL_FAMILY	TIUMgnu 🔍	Edit
	Active build spec: PEN Name (Common) TOOL_FAMILY TOOL	TIUMgnu ▼ Value (Build spec specific)	New
	Active build spec: PEN Name (Common) TOOL_FAMILY TOOL TOOL_PATH	TIUMgnu 🔍	Edit
	Active build spec: PEN Name (Common) TOOL_FAMILY TOOL TOOL_PATH CC_ARCH_SPEC	TIUMgnu Value (Build spec specific) gnu gnu -mtune=pentium -march=pentium -nostdl	Edit
	Active build spec: PEN Name (Common) TOOL_FAMILY TOOL TOOL_PATH	TIUMgnu 🔍	Edit Edit Rename Copy
	Active build spec: PEN Name (Common) TOOL_FAMILY TOOL TOOL_PATH CC_ARCH_SPEC VSB_DIR	TIUMgnu Value (Build spec specific) gnu gnu -mtune=pentium -march=pentium -nostdl \$(WIND_BASE)/target/lib	Edit
	Active build spec: PEN Name (Common) TOOL_FAMILY TOOL TOOL_PATH CC_ARCH_SPEC VSB_DIR VSB_CONFIG_FILE	TIUMgnu Value (Build spec specific) gnu gnu -mtune=pentium -march=pentium -nostdl \$(WIND_BASE)/target/lib \$(VSB_DIR)/h/config/vsbConfig.h	Edit Edit Rename Copy
	Active build spec: PEN Name (Common) TOOL_FAMILY TOOL TOOL_PATH CC_ARCH_SPEC VSB_DIR VSB_CONFIG_FILE LIBPATH	TIUMgnu ▼ Value (Build spec specific) gnu gnu -mtune=pentium -march=pentium -nostdl \$(WIND_BASE)/target/lib \$(VSB_DIR)/h/config/vsbConfig.h -L\${NDDSHOME}/lib/pentiumVx6.9gcc4.3.	Rename Copy Delete
?)	Active build spec: PEN Name (Common) TOOL_FAMILY TOOL TOOL_PATH CC_ARCH_SPEC VSB_DIR VSB_CONFIG_FILE LIBPATH	TIUMgnu Value (Build spec specific) gnu gnu mtune=pentium -march=pentium -nostdl \$(WIND_BASE)/target/lib \$(VSB_DIR)/h/config/vsbConfig.h -L\${NDDSHOME}/lib/pentiumVx6.9gcc4.3. -Inddscppz -Inddscz -Inddscorez Inddscorez Inddscore	Rename Copy Delete

14. In the **Build Paths** or **Paths** tab, add both of these:

-I\$(NDDSHOME)/include

-I\$(NDDSHOME)/include/ndds

Click **Apply** to save the changes.

😣 🗉 Properties for tes	tApplication	
type filter text	Build Properties	(⇒ + ⇒ + +
 Resource Binary Parser Build Properties Builders C/C++ General Code Coverage Ana 	Specify all build Build Support and Specs Tools Paths # Defines Librarie Build spec specific settings Active build spec: PENTIUMgnu	es 🕏 Variables
Project Info Project References	Include directories	Generate
Run/Debug Settings Task Tags	-l\$(WIND_BASE)/target/h -l\$(WIND_BASE)/target/h/wrn/coreip	Edit
Validation	-I\${NDDSHOME}/include -I\${NDDSHOME}/include/ndds	Add Add to all
		Remove
	Redirection directory: PENTIUMgnu	Remove from all Up Down Down Default Browse
	Note: Leave this field blank to store build output together with t an absolute path (environment variables are permissible)	
4	Restore D	efaults <u>A</u> pply
?	Cano	cel OK

- 15. Click **OK** to exit the Properties menu.
- 16. Build the project by right-clicking on the project in Project Explorer, then selecting **Build Project**.

😣 🚍 🗉 🛛 Basic Device Development Wind River	r Workbench			
<u>File Edit Source Refactor Navigate Search</u>	<u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp			
] 🖬 😂 🖩 🛍 🚔 🗍 📾 😫 🕷 🏇 • O •] 🛷	?▼] ■ ■ ∯ + ₩ + ₩ + \$\$			😭 🔤 Basic De
🕒 Project Explorer 🛛 🛛 😑 😓 🖬 🗧	🜌 • 🍷 🗖 🕥 Getting Started 🛿 📄 creation.log	🗈 usrAppInit.c 📄 creation.log	📄 creation.log 📃 🗖	🗄 Outline 🛿 🗖 🗖
SetworkImage (vxworks-cert-6.6.4.1 [not inspective]	stalled] II 🚔 🏠 🧔 🔿			An outline is not available.
▶ ﷺ rti-slo-6.9.4.2-sbc8641d-v1 (Wind River VxW	orks 6.9			1
Samsung-iso-v1 (wrlinux-4 [not installed] Pla	atform Pro Wind River	Workbench 3.3 \star 🕷	/hat's New	
samsung-iso-v2 (wrlinux-4 [not installed] Pla	atform Pro			
samsung-iso-v3 (wrlinux-4 [not installed] Pla	itform Pro		AAA	
Samsung-v1 (wrlinux-4 [not installed] Platfor	rm Projec			
Est (vxworks-cert-6.6.4.1 [not installed] Image	age Proje			
Test_Vx6ImageProject (vxworks-cert-6.6.4.1	[not inst	Getting Started		
👻 🐸 testApplication (Wind River VxWorks 6.9 Do	wploadak			
Build Targets (PENTIUMgnu - debug)	New Go Into	e		
Search Wind River Launches				
Binaries	Open in <u>N</u> ew Window	9		
Includes	Copy	Ctrl+C Ctrl+V te		
🕨 🐸 PENTIUMgnu	Paste Delete	Delete		
HelloWorld_publisher.cxx	Attributes	Share		
HelloWorld_subscriber.cxx	Mo <u>v</u> e			
le HelloWorld.cxx	Rena <u>m</u> e	F2 plach		
h HelloWorld.h	Import			
HelloWorldPlugin.cxx	Export			
h HelloWorldPlugin.h	Open Wind River VxWorks 6.9 Development Shell			
HelloWorldSupport.cxx	Build Project	Ctrl+B Ctrl+P		
HelloWorldSupport.h	Rebuild Project			
<u> </u>	Build Options Re <u>f</u> resh	F5		
📲 Remote Systems 🕱	Close Project	13		
🖉 🗢 📣 🔅 🐉 🐅 象	Close Unrelated Projects			
▶ 📑 Local	Project References	•		
🕨 🌆 LinuxHost_local (Host OS (Native Developme	er <u>R</u> un VxWorks Kernel Task			
Vxsim0 (Wind River VxWorks 6.9)	Debug VxWorks Kernel Task			
kinet with the second secon	Download VxWorks Kernel Task			
	Index	►		
	<u>B</u> un As	🕨 Search 🛛 😁 🕶	•	🕹 🕆 😫 🖉 🐻 🖬 🔍 🗖 🗖
	Debug As		on/HelloWorld_publisher.o	- -
	Download Profile As		ation/HelloWorld_subscriber.o"`"]; t on/HelloWorld_subscriber.o	hen mkdir -p " dirname "test
	Team	g/testApplication_par	rtiallmage.o"`"]; then mkdir -p "`di	rname "testApplication_parti
	Comp <u>a</u> re With	tion out">" 1 then mil	allmage.o kdir -p "`dirname "testApplication/D	ebug/testApplication out"`"
	Restore from Local History	put		
	P <u>r</u> operties	Alt+Enter 8-09-14 14:51:34	NTIUMgnu (Elapsed Time: 00:50)	
		2 33-14 14:51:54	(214)554 (11115) (0150)	v
				Þ
e* EstApplication (Wind River VxWorks)	5 6.9 Downloadable Kernel Module Project)		29M of 156M 📋	

17. Run the application as described starting in <u>Step 5 in the 'Using the Command Line' section</u>, except load **HelloWorld.out** instead of **HelloWorld_subscriber.so** when you get to <u>Step 8</u>.

3.3.3 Building and Running an Application as a Real-Time Process

There are two ways to build and run your Connext RTP application:

- 3.3.3.1 Using the Command Line below
- 3.3.3.2 Using Workbench on page 25

3.3.3.1 Using the Command Line

1. Generate the source files and the makefile with RTI Code Generator (rtiddsgen).

Note: The architecture names for Kernel Mode and RTP Mode are different.

Please refer to the *RTI Code Generator User's Manual* for more information on how to use *rtiddsgen*.

2. Set up your environment with the **wrenv.sh** script or the **wrenv.bat** batch file in the VxWorks base directory. Execute the script with the **-p** parameter. For example:

wrenv.sh -p vxworks

- 3. Set the NDDSHOME environment variable as described in Set Up Environment Variables (rtisetenv), in "Hands-On 1" of *Introduction to Publish/Subscribe*, in the <u>RTI Connext Getting Star-</u> ted Guide.
- 4. Build the Publisher and Subscriber modules using the generated makefile. You may need to modify the HOST_TYPE, compiler and linker paths to match your development setup.

Notes:

• Steps 5-12 can be replaced by establishing a telnet connection to the VxWorks target. In that case, Workbench does not need to be used and both the Host Shell and Target Console will be redirected to the telnet connection. Once in the C interpreter (you will see a prompt '->' in the shell) you can type **cmd** and then **help** for more information on how to load and run applications on your target.)

Using rtpSp:

```
telnet raytheon-guy
cd " <PROJECT ROOT FOLDER>"
rtpSp "objs/<arch>/Foo subscriber.vxe -domainId XX"
```

Or using rtp exec:

```
telnet raytheon-guy
cd " <PROJECT ROOT FOLDER>"
rtp exec objs/<arch>/Foo subscriber.vxe - -domainId XX
```

• If you want to dynamically link your RTP to the RTI libraries, make the following modifications the generated makefile:

LIBS = -L\$(NDDSHOME)/lib/<architecture> -non-static -lnddscpp \-lnddsc lnddscore \$(syslibs <architecture>)

- 5. Add to the LD_LIBRARY_PATH environment variable the path to your RTI libraries as well as the path to libc.so.1 of your VxWorks installation to launch your RTP successfully.
- 6. Launch Workbench.
- 7. Make sure your target is running VxWorks.
- 8. Connect to the target with the target manager and open a host shell and a Target Console Tool to look at the output. Both are found by right-clicking the connected target in the **Target Tools** submenu.
- 9. Right-click on your target in the Target Manager window, then select Run, Run RTP on Target.

- 10. Set the **Exec Path on Target** to the **HelloWorld_subscriber.vxe** or the **HelloWorld_pub-lisher.vxe** file created by the build.
- 11. Set the arguments (domain ID and number of samples, using -d <domain ID> and -s <number of samples>).

A Stack size of 0x100000 should be sufficient. If your application doesn't run, try increasing this value.

12. Click Run.

3.3.3.2 Using Workbench

Note: The following steps might vary slightly depending on your chosen version of VxWorks.

- 1. Start Workbench.
- 2. Select File, New, Wind River Workbench Project.

😣 🖱 💷 Basic Device Development W	/ind River Workbench		
<u>File</u> <u>Edit</u> <u>Source</u> Refactor <u>N</u> avigate	Se <u>a</u> rch <u>P</u> roject <u>R</u> un	<u>W</u> indow <u>H</u> elp	
<u>N</u> ew	Shift+Alt+N 🕨	Wind River Workbench Project	
Open File <u>.</u>		P <u>r</u> oject	
Close	Ctrl+W	Build Target	
Save	Ctrl+S	Folder	
		File	
Re <u>f</u> resh	F5	File from Template	
Convert Line Delimiters To	•		
Print	Ctrl+P	E <u>x</u> ample	
<u></u>	CUITP	Other Ctrl	+N

3. Select the desired Target Operating System; click Next.

😣 🗉 New Wind River W	/orkbench Project	
Target Operating Syste	em	->
Select the target opera	ating system for the project.	
Target operating syster	n: Wind River VxWorks 6.9	
larger operating system	Host OS (Native Development)	
?	< <u>B</u> ack <u>N</u> ext > Cancel	<u>F</u> inish

4. When prompted to choose a Build Type, select Real Time Process Application; click Next.

😣 🗊 New W	ind River Workben	ch Project		
Build Type				
Select the b	uild type for the pr	oject.		
	Boot Loader / BSF	Project		
Build type:	Downloadable Ker	nel Module		
	Kernel Libraries			
Description:	ROMFS File Syster	n		
	Real Time Process	Application		
	Relinkable Kernel	Module		
	Shared User Libra	ry		
	Static Kernel Libra	ary		
	Static User Librar	у		
Setup inform	System Image			
Resulting pr	User-Defined			
	None			
Uses build t	ool: Linker			
?	< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

- 5. Give your project a name; click Next.
- 6. Leave everything else at its default setting; click Finish.

Your project will be created at this time.

- 7. Copy the source and header files generated by *rtiddsgen* in 3.3.1 Generate Example Code and Makefile with rtiddsgen on page 12 into the project directory. There can only be one main() in your project, so you must choose *either* a subscriber or a publisher. If you want to run both, you will need to create two separate projects.
- 8. View the added files by right-clicking on the project in Project Explorer, then selecting **Refresh** to see the files.
- 9. Open the project Properties by right-clicking on the project in Project Explorer and selecting **Properties**.
- 10. In the dialog box that appears, select **Build Properties** in the navigation pane on the left.
- 11. In the **Build Support and Specs** tab, select the desired build spec from the **Active build spec** drop-down menu; click **Apply** to save the changes.
- 12. In the **Build Macros** or **Defines** tab, add the following to DEFINES in the Build macro definitions:

-DRTI_VXWORKS -DRTI_STATIC -DRTI_RTP

😣 💷 Properties for testApplicationRTP				
type filter text	Build Properties	⇔ • ⇔• •		
Resource	Define and undefine directives	Edit		
Binary Parser	-DCPU=_VX_\$(CPU)	Add		
Build Properties Builders	-DTOOL_FAMILY=\$(TOOL_FAMILY)	Add to all		
✓ C/C++ General	-DTOOL=\$(TOOL)			
Code Style	-D_WRS_KERNEL	Remove		
Documentation	-D_VSB_CONFIG_FILE=\"\$(VSB_DIR)/h/config/vsbConfig.h\"	Remove from all		
File Types	-DIP_PORT_VXWORKS=69			
▶ Indexer	-DRTI_VXWORKS	Up		
Code Coverage Ana	-DRTI_STATIC	Down		
Project Info	-DRTI_RTP			
Project References				
Run/Debug Settings				
Task Tags				
Validation				
?		Cancel OK		

13. In the Variables tab, add to LIBPATH:

-L/(NDDSHOME)/lib/<architecture>

If you are using *static* linking, add to LIBS:

-Inddscppz -Inddscz -Inddscorez (in that order)

If you are using *dynamic* linking, add to LIBS:

-Inddscpp -Inddsc -Inddscore (in that order)

Click Apply to save the changes.

14. In the **Build Paths** or **Paths** tab, add:

-I\$(NDDSHOME)/include

-I\$(NDDSHOME)/include/ndds

Click Apply to save the changes.

😣 💿 Properties for tes	tApplicationRTP		
type filter text	Build Properties		(⇒ + ⇒) + +
 Resource Binary Parser Build Properties Builders C/C++ General Code Coverage Ana Design Lafe 	Specify all build ✓ Build Support and Specs	es 🕸 Var	iables
Project Info Project References	Include directories	Ge	enerate
Run/Debug Settings Task Tags	-l\$(WIND_BASE)/target/h -l\$(WIND_BASE)/target/h/wrn/coreip		Edit
Validation	-l\${NDDSHOME}/include		Add
	-l\${NDDSHOME}/include/ndds	Ad	d to all
		R	emove
		Remo	ve from all
			Up
			Down
	Redirection directory: PENTIUMgnu Redirection root directory: Note: Leave this field blank to store build output together with t an absolute path (environment variables are permissible)		
	Restore <u>E</u>	<u>)</u> efaults	<u>A</u> pply
<u>?</u>	Can	ncel	ОК

- 15. Click **OK** to exit the Properties menu.
- 16. Build the project by right-clicking on the project in Project Explorer, then selecting **Build Project**.
- 17. Run the application as described starting in Step 5 in the Command Line section above.

3.4 Using DDS Ping and Spy

This section describes special usage notes when running the *RTI DDS Ping* and *Spy* command-line utilities on VxWorks systems. For complete details on using both utilities, see the API Reference HTML documentation (under Modules, Programming Tools).

RTI DDS Ping (rtiddsping) tests the connectivity of your system. It uses *RTI Connext* to send and receive "Ping" messages to other *rtiddsping* applications running on the same or different computers.

RTI DDS Spy (rtiddsspy) shows you what is being published and subscribed to.

When running these utilities on VxWorks systems in RTP mode (as Real-Time processes):

- The utilities must be executed in a command prompt (running the "cmd" command in the C-shell)
- The utilities are statically linked so they don't require any LD_LIBRARY_PATH setup.
- The name of the utilities are suffixed with a "z" to indicate that they are statically linked (i.e. *DDS Ping* is called **rtiddspingz.vxe**).
- Each executable can be run as in any Linux OS (e.g., rtiddspingz.vxe -help).

When running these utilities on VxWorks systems in kernel mode (as DKMs):

- The modules libnddscore.so, libnddsc.so, and libnddscpp.so must first be loaded.
- After loading the *Connext* modules, the utility module must be loaded in order to run it (i.e., **rtiddsping.so**).
- All the command-line options must be passed embedded in a single string (see examples below).
- The command must be typed in the VxWorks shell (either an rlogin shell, a target-server shell, or the serial line prompt).

The examples below illustrate how to run the utilities in Kernel mode. The string "vxworks prompt>" represents the prompt that the shell prints and is not part of the command that must be typed.

Ping:

```
vxworks prompt> rtiddsping "-domainId 3 -publisher -numSamples 100"
vxworks prompt> rtiddsping "-domainId 5 -subscriber -timeout 20"
vxworks prompt> rtiddsping "-help"
```

Spy:

```
vxworks prompt> rtiddsspy "-domainId 3 -topicRegex Alarm*"
vxworks prompt> rtiddsspy "-help"
```

Or if the stack of the shell is not large enough, use "taskSpawn" to avoid overflowing the stack (each utility requires ~25 kB of stack).

Ping:

```
vxworks prompt> taskSpawn "rtiddsping", 100, <floating_point_option>, 50000, rtiddsping, \
    "-domainId 3 -publisher -numSamples 100"
vxworks prompt> taskSpawn "rtiddsping", 100, <floating_point_option>, 50000, rtiddsping, \
    "-domainId 5 -subscriber -timeout 20"
vxworks prompt> taskSpawn "rtiddsping", 100, <floating_point_option>, 50000, rtiddsping, "-
help"
```

Spy:

```
vxworks prompt> taskSpawn "rtiddsspy", 100, <floating_point_option>, 50000, rtiddsspy, \
    "-domainId 3 -topicRegex Alarm*"
vxworks prompt> taskSpawn "rtiddsspy", 100, <floating_point_option>, 50000, rtiddsspy, "-
help"
```

Where <floating_point_option> is a numeric value that varies depending on the hardware. See <u>Enabling</u> <u>Floating Point Coprocessor in Kernel Tasks</u>, in the VxWorks chapter of the <u>RTI Connext Core Librar</u>ies Platform Notes.