# **RTI Connext DDS Core Libraries**

# Getting Started Guide Addendum for Embedded Systems Version 6.1.2



© 2023 Real-Time Innovations, Inc. All rights reserved. Printed in U.S.A. First printing. September 2023.

### Trademarks

RTI, Real-Time Innovations, Connext, 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 <a href="https://www.rti.com/terms">https://www.rti.com/terms</a> 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.

This is an independent publication and is neither affiliated with, nor authorized, sponsored, or approved by, Microsoft Corporation.

The security features of this product include software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<u>http://www.openssl.org/</u>). This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

### Notice

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: https://support.rti.com/

# Contents

Chapter 1 Addendum for Embedded Platforms
Chapter 2 Getting Started on QNX Embedded Systems
2.1 Building and Running a Hello World Example
2.2 Configuring Automatic Discovery
Chapter 3 Getting Started on INTEGRITY Systems
3.1 Building the Kernel
3.2 Building and Running a Hello World Example
3.2.1 Generate Example Code and Project File with rtiddsgen
3.2.2 Build the Publish and Subscribe Applications
3.2.3 Connect to the INTEGRITY Target from MULTI
3.2.4 Load the Application on the Target
3.2.5 Run the Application and View the Output
Chapter 4 Getting Started on VxWorks Systems
4.1 Building the VSB
4.2 Building the Kernel 13
4.3 Building and Running a Hello World Example
4.3.1 Generate Example Code and Makefile with rtiddsgen
4.3.2 Building and Running an Application as a Kernel Task
4.3.3 Building and Running an Application as a Real-Time Process
4.4 Using DDS Ping and Spy

# Chapter 1 Addendum for Embedded Platforms

In addition to enterprise-class platforms like Microsoft Windows and Linux, *RTI*® *Connext*® *DDS* 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 DDS* 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 DDS*.

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 DDS* 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 DDS Getting Started</u> <u>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 DDS 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\_cpp8.3.0'.
- 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.

For C++:

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

For Java:

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

Edit the example code to add the line **sprintf(instance->msg, "Hello World! (%d)", count);** as follows:

```
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);
}
```

 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 DDS</u> <u>Getting Started Guide</u>.

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

5. Connect to the QNX target (using ssh, for example) and start the subscriber application, **Hel-loWorld\_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)"
```

# 2.2 Configuring Automatic Discovery

In most cases, multiple applications—whether on the same host or different hosts—will discover each other and begin communicating automatically. However, in some cases you must configure the discovery service manually. For example, on LynxOS systems, multicast is not used for discovery by default; you will need to configure the addresses it will use. See information about setting discovery peers in the "Troubleshooting" section of *Introduction to Publish/Subscribe*, in the <u>RTI Connext DDS Getting Started Guide</u>.

# Chapter 3 Getting Started on INTEGRITY Systems

This section provides simple instructions on configuring a kernel and running *Connext DDS* applications on an INTEGRITY system. These instructions assume that the application module will be dynamically downloaded. Please refer to the documentation provided by Green Hills Systems for more information about this operating system.

For more information on using *Connext DDS* on an INTEGRITY system, please see the INTEGRITY section of the RTI Connext DDS Core Libraries Platform Notes.

The first section describes 3.1 Building the Kernel below.

The next section guides you through the steps to build and run an rtiddsgen-generated example application on an INTEGRITY target: 3.2 Building and Running a Hello World Example on the next page.

Before you start, make sure that you know how to:

- 1. Boot/reboot your INTEGRITY target.
- 2. Get the serial port output of your target (using telnet, minicom or hyperterminal).

# 3.1 Building the Kernel

Before you start, you should be familiar with running a kernel on your target.

- 1. Launch MULTI.
- 2. Select File, Create new project.
- 3. Choose the INTEGRITY Operating System and make sure the path to your INTEGRITY distribution is correct.
- 4. Choose a processor family and board name.

- 5. Click Next.
- 6. Choose Language: C/C++.
- 7. Project type: INTEGRITY Kernel.
- 8. Choose a project directory and name.
- 9. Click Next.
- 10. In Kernel Options, choose at least: 'TCP/IP stack'. Everything else can be left to default.
- 11. In the Project Builder, you should see the following file:

<name of your project>\_default.ld (under src/resource.gpj).

12. Right-click the file and edit it; the parameters of interest are the following:

```
CONSTANTS
{
    INTEGRITY_DebugBufferSize = 0x10000
    INTEGRITY_HeapSize = 0x100000
    INTEGRITY_StackSize = 0x4000
    INTEGRITY_DownloadSize = 0x400000
    INTEGRITY_MaxCoreSize = 0x200000
}
```

Note that most *Connext DDS* applications will require the StackSize and HeapSize parameters to be increased from their default value. The values shown above are adequate to run the examples presented in this document.

- 13. Once you have changed the desired values, right-click the top-level project and select Build.
- 14. Run the new kernel on your target.

# 3.2 Building and Running a Hello World Example

This section describes the basic steps for building and running an *rtiddsgen*-generated example on an INTEGRITY target:

- 3.2.1 Generate Example Code and Project File with rtiddsgen on the next page
- 3.2.2 Build the Publish and Subscribe Applications on page 8
- 3.2.3 Connect to the INTEGRITY Target from MULTI on page 8
- 3.2.4 Load the Application on the Target on page 8
- 3.2.5 Run the Application and View the Output on page 9

### 3.2.1 Generate Example Code and Project File with rtiddsgen

#### To create the example applications:

- 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 project file. Choose either C or C++.

#### For C:

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

#### For C++:

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

In your **myhello** directory, you will see that *rtiddsgen* has created a number of source code files (described in *Generated Files*, in the <u>RTI Code Generator User's Manual</u>), additional support files (not listed here), and a project file: **HelloWorld\_default.gpj**.

For C only, edit the example code (to add the line sprintf(instance->msg, "Hello World! (%d)", count);) as follows:

```
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);
}
```

You do not need to edit the C++ code, *rtiddsgen* has added **sprintf(instance->msg, "Hello World!** (%d)", count); for C++ already.

### 3.2.2 Build the Publish and Subscribe Applications

1. In a plain text editor, edit the top-level project file that was generated by *rtiddsgen*, **HelloWorld\_ default.gpj**, so that it points to the path to your INTEGRITY distribution:

Set macro \_\_OS\_DIR=path to your INTEGRITY distribution>

- 2. Save your changes.
- 3. Launch MULTI.
- 4. Open the top-level project file, HelloWorld\_default.gpj, in MULTI:

Select Components, Open Project Manager, then open the project file from there.

5. Right-click on the top-level project and build the project.

# 3.2.3 Connect to the INTEGRITY Target from MULTI

- 1. From the MULTI Launcher, click the Connection button and open the Connect option. Your mode should be Download (Download and debug application).
- 2. Create a custom connection with the following line:

For targets that only support the older INDRT connection mechanism:

rtserv -port udp@<ip address of your INTEGRITY target>

For targets that support the newer INDRT2 connection mechanism:

rtserv2 -port udp@<ip address of your INTEGRITY target>

(You might be able to see the IP address of your target on the output of its boot sequence.)

You only have to create your connection once, MULTI will remember it.

3. Make sure your target has booted; *then* select **Connect**. You should see a new window with the Kernel Tasks running on your target.

# 3.2.4 Load the Application on the Target

- 1. In the task window, select Target, Load module.
- 2. Browse for your executables; there should be 3 of them in your project directory:
  - HelloWorld\_publisherdd
  - HelloWorld\_subscriberdd
  - posix\_shm\_manager
- 3. Load the **posix\_shm\_manager** first, it will appear in the **Tasks** window as a separate address space and start running by itself once loaded. It will allow you to use the shared memory transport on your target.

Note: The default *rtiddsgen*-generated code tries to use shared memory, so unless you have manually disabled it, your application will crash if you do not load the shared memory manager before running the application.

4. Load the publisher, subscriber, or both. They should appear in separate address spaces in the Tasks window.

### 3.2.5 Run the Application and View the Output

1. Select the task called "Initial" in your application's address space in the Tasks window; you can either click the play button to run it, or click the debug button to debug it.

Note that with some versions of INTEGRITY, it is difficult to pass arguments to applications. Arguments can always be hard-coded in your application before compiling it. To quickly experiment with multiple runs of the application with different arguments, one option is to run your application within the debugger. Then you can set a breakpoint before the arguments are used and change them at that point.

2. From the Tasks window, select **Target**, **Show Target Windows**. This will show you the standard output of your target.

Some errors messages may still go through the serial port, so you should leave your serial port connection open and monitor it as well.

### To reboot the target:

Go to your serial port connection monitor and type 'rset'.

# Chapter 4 Getting Started on VxWorks Systems

This section provides simple instructions to configure a kernel and run *Connext DDS* applications on systems based on VxWorks 7, including VxWorks 21.11. 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 VxWorks 7 systems by expanding on the <u>VxWorks section of the RTI Connext DDS Core Libraries Platform Notes</u>; please read the following alongside that section.

This chapter uses VxWorks 7 as an example. The steps for VxWorks 21.11 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:

- 4.1 Building the VSB on the next page
- 4.2 Building the Kernel on page 13

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

• 4.3 Building and Running a Hello World Example on page 18

For tips on using RTI DDS Ping and Spy, see 4.4 Using DDS Ping and Spy on page 34.

# 4.1 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

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



3. For the Build type, select Source Build.



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

New VxWorks Source Build Project 🔹 🗈 🗙						
Project Setup Base the new pro type.	oject either on an existing cor	figuration, a board support package, or a CPU				
Setup the project	ct					
Based on a Vx	Works 7 board support packa	ge	\$			
BSP:	itl_generic_1_0_7_0	\$ Show latest ver	sions only			
Active CPU:	NEHALEM		0			
Baseline:	✓ SR0540 2018-06-15					
Address mode:	ILP32 32-bit libraries		-			
Options:	Compiler support:	Primary and secondary compiler	÷			
	Floating point setting:	hard				
	Endianess setting:	little				
	VxWorks 6.9 compatibility:	Disabled for this BSP				
	Processor mode:	SMP support in Libraries	\$			
	Debug mode:	Off, and normal compiler optimizations enabled				
	IP version setting:	IPv6 and IPv4 enabled librarles	¢			
	VSB profile:	None	÷			
Link in sour	rces to project					
Setup information	00					
Base location:	itl_generic_1_0_7_0					
?	(	< Back Next > Cancel	Finish			

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

# 4.2 Building the Kernel

This section explains how to build a kernel capable of loading *Connext DDS* libraries. *Connext DDS* 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.
- 2. Select File, New, Wind River Workbench Project.

😕 🗐 🔲 Basic Device Development W	Vind 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
Reiresn	FD	File from Template
Convert Line Delimiters To	•	Example
Print	Ctrl+P	- <u>Aunipien</u>
		Other Ctrl+N

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

😣 🗉 New Wind River Workbench Project							
Target Operating System							
Select the target operatir	ng system for	the project.					
Target operating system:	Wind River \	/xWorks 6.9					
	Host OS (Na	tive Development	:)				
?	< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish			

4. When prompted to choose a **Build type**, select **System Image** (this may be **Kernel Image** or **VxWorks Image** depending on your version of VxWorks); click **Next**.

😣 🗊 New W	ind River W	/orkbench Proje	ct		
<b>Build Type</b> Select the b	uild type fo	or the project.			
Build type:	System In	nage			
Description:	Creates a with kerne	bootable VxWor I modules, and	ks operating syst can be configured	tem image, which d to include a ROM	can be linked IFS file system.
Setup inform	nation				
Resulting pr	oject type	Image Project			
Uses build t	ool:	None			
?		< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

5. Give your project a name; click Next.

😣 🗉 🛛 New VxWorks Im	age Project			
Project Create a new VxWork	s image project wi	th all available ke	rnel build specs.	
Project name: testKe	rnel		·	
Location				
O Create project in <u>v</u>	<u>rorkspace</u> <u>x</u> ternal location			
Directory: /home/loca	al/WindRiver/works	space/testKernel	-	Browse
?	< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

6. In Project Setup, for the **Based on** field, choose **a source build project**. For the **Project**, choose the VSB you created and built in the previous section. The BSP, SMP support and other options will be correctly populated from the VSB configuration.

For the Tool chain option, select GNU; click Next.

😣 🗉 New	v VxWorks Image	Project						
Project Se	Project Setup							
Base the and a too	new project eith I chain.	er on an existing project, or on a board support package						
Setup the	e project							
Based on	a board suppor	t package 🛛 🔻						
Project:		<b>■</b> Browse						
BSP:	wrSbc8641d	Browse						
	Address mode:	32-bit kernel ✓						
		Tool chain: gnu 🛛 🗸						
Target Ag	jent							
Enable	e WDB Target Ag	ent						
BSP valid	ation test suite-							
Add su	upport to project	Options						
Setup info	ormation							
Base dire	ectory: /local/VxV	Vorks/GPP-3.9/vxworks-6.9/target/config/wrSbc8641d						
?		< <u>B</u> ack <u>N</u> ext > Cancel <u>F</u> inish						

7. In **Options**, select **SMP support in kernel** if your BSP supports it and you want to enable symmetric multi-processing capability in the kernel.

Select **IPv6 enabled kernel libraries** if your architecture supports IPv6 (See the <u>VxWorks section</u> of the <u>RTI Connext DDS Core Libraries Platform Notes</u> to check if your architecture supports IPv6); click **Next**.

😣 🗉 🛛 New VxWorks Imag	e Project			
Options				
Select the options to be	used.			
Options				
Select All Deselect	All			
SMP support in kern	el			
✓ IPv6 enabled kernel	libraries			
(?)	< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

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

😣 🗊 New Vx	Works Im	nage Project				
Configuration	n Profile					
Select kerne	l configur	ation profile.				
Profile: PRO	FILE_DEV	ELOPMENT	•			
Profile inform	nation —					
Description:	VxWork	s Kernel Developn	nent Configura	tion Profile		<u>^</u>
						_
Synopsis:	VxWork	s kernel including	development	and debugging	components	×
						<b>V</b>
2		< Back	Next >	Cance		nish
$\odot$			Mext >			IISTI

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

Your project will be created at this time.

10. From the Project Explorer, open Kernel Configuration.



- 11. Add Operating System Components, Kernel Components, \_thread variables support.
- 12. 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 DDS Core Libraries Platform Notes.

- If you plan to use any *Connext DDS* 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.
- 14. 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.
- 15. 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).

- 16. 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.
- 17. 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/.

# 4.3 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 6.x/7 target using kernel mode or RTP mode.

# 4.3.1 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 DDS 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 DDS Getting Started Guide</u>.

### 4.3.2 Building and Running an Application as a Kernel Task

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

- 4.3.2.1 Using the Command Line below
- 4.3.2.2 Using Workbench on the next page

### 4.3.2.1 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 set to the correct version of VxWorks. For example:

wrenv.sh -p vxworks-6.9

- 2. 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 DDS Getting</u> <u>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 DDS* 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 6.x 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 DDS* libraries being statically linked in both modules. To overcome that situation, see the "Notes for VxWorks 7 Platforms" section in the *RTI Connext DDS 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 Enabling Floating Point Coprocessor in Kernel Tasks, in the VxWorks chapter of the RTI Connext DDS Core Libraries Platform Notes.

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

### 4.3.2.2 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 Wind River Workbench							
<u>File</u> <u>E</u> dit <u>Source</u> Refactor <u>N</u> avigate Search <u>Pressure</u>	<u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp						
<u>N</u> ew Shift-	ft+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						
Con <u>v</u> ert Line Delimiters To	•						
Drint	E <u>x</u> ampie						
Euur	<u>O</u> ther Ctrl+N						

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

😣 🗊 New Wind River Workbench Project						
Target Operating Syst	em					
Select the target oper	ating system for	the project.				
Target operating syste	m: Wind River \	√xWorks 6.9				
	Host OS (Na	tive Development	)			
?	< <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 Wind River Workbench Project								
Build Type	Build Type 🔶							
Select the b	uild type for the pr	oject.						
	Boot Loader / BSF	P Project						
Build type:	Downloadable Kei	rnel Module						
	Kernel Libraries							
Description:	ROMFS File Syster	m						
	Real Time Process	s Application						
	Relinkable Kernel	Module						
	Shared User Libra	iry						
	Static Kernel Libra	ary						
	Sustem Image	У						
Setup inform	User-Defined							
Resulting pr	None							
Uses build t	ool: Linker							
(V)	< <u>B</u> ack	<u>N</u> ext >	Cancel	Finish				

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 project	ct in <u>w</u> orkspace	2		
O Create proje	ct at e <u>x</u> ternal le	ocation	t ovtornal locati	
Directory: /hom	ne/local/WindRi	ver/workspace/t	testApplicat	B <u>r</u> owse
?	< <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 Workbench		
<u>File Edit Source Refactor Navigate Search Project Run Window H</u> elp		
188 12 - 🔤 🎭 崎 🚔 🎎   1 🍫 - Ο -   🖋 - ] 🎟 🔟   20 - 河 - 🏷 - ◇ -   🖻		😫 🏧 Basic De
🕒 Project Explorer 🛛 🦳 🤹 🐑 🗮 🖉 🗸 🧮 🖉 🖓 Getting Started 🕸 📄 creation.log 🗋 usrAppInit.c	reation.log 📄 creation.log 👘 🗖	🗄 Outline 🛛 🦳 🗖 🗖
🕨 🥵 NetworkImage (vxworks-cert-6.6.4.1 [not installed] Ima 👔 🖕 😅		An outline is not available.
▶ ﷺrti-slo-6.9.4.2-sbc8641d-v1 (Wind River VxWorks 6.9 Im		
Samsung-iso-v1 (wrlinux-4 [not installed] Platform Proje Wind River Workbench 3.3	🛨 What's New	
🕨 🖆 samsung-iso-v2 (wrlinux-4 [not installed] Platform Proje		
🕨 🖆 samsung-iso-v3 (wrlinux-4 [not installed] Platform Proje	AAA	
Samsung-v1 (wrlinux-4 [not installed] Platform Project)		
▶ 🗳 test (vxworks-cert-6.6.4.1 [not installed] Image Project		
▶ ≝ Test_Vx6ImageProject (vxworks-cert-6.6.4.1 [not install]       Getting Started		
💌 🖆 testApplication (Wind River VxWorks 6.9 Downloadable		
Build Targets (SIMLINUXdiab - debug) Create		
Wind River Launches		
▶ 🔊 Includes		
Construction of the second secon		
Calibur1700-micro-cert-v1 (vxworks-cert-6.6.4.1 [not in		
Zcalibur1700-Test_Image (vxworks-cert-6.6.4.1 [not inst     Set-up & Share		
1 Stow Wetcome Splash 🕨		
📲 Remote Systems 🕱 👘 🗖		
& 19 bi \$\$ \$\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$		
🗢 🖬 Local		
🕨 🎯 Wind River Registries 👘 Help 🕎 Build Console 🛛 🖉 Tasks 🖁 Problems 🔗 Search	\$\$ * B 📄 🔽	🕹 😚 🔽 🔊 🔍 🔽 🗖
۶ 🗞 Local Files		*
🛱 Local Shells		
👂 🍓 slo (Wind River VxWorks 6.9)		
العلم المعالي ا	108M of 157M	

- 7. Copy the source and header files generated by *rtiddsgen* in 4.3.1 Generate Example Code and Makefile with rtiddsgen on page 18 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.



- 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** dropdown menu; click **Apply** to save the changes.

type filter text	Build Properties	(⇒ + ⇒ + +
<ul> <li>Resource</li> <li>Binary Parser</li> <li>Build Properties</li> </ul>	Specify all build We Build Support and Specs Ø Tools ⊘Paths # Defines Libraries \$ V	ariables
Builders ▷ C/C++ General Code Coverage Ana	<ul> <li>Build support</li> <li>⊙ Managed build (makefiles generated by the IDE)</li> <li>○ Disabled</li> </ul>	
Project Info Project References Run/Debua Settinas	Build command: %makeprefix% makeno-print-directory	Variables
Task Tags	Available and enabled build specs:	
Validation	<ul> <li>ARMARCH4diab</li> <li>ARMARCH4diabbe</li> <li>ARMARCH4gnu</li> <li>ARMARCH4gnube</li> <li>ARMARCH5diab</li> <li>ARMARCH5diabbe</li> <li>ARMARCH5gnu</li> <li>ARMARCH5gnube</li> <li>ARMARCH5gnube</li> <li>ARMARCH6diab</li> <li>ARMARCH6diab</li> <li>ARMARCH6diabbe</li> <li>ARMARC</li></ul>	Enable All     Disable All     New     Import     Rename     Copy     Delete
	Default build spec:       SIMLINUXdiab       ▼         Active build spec:       PPC604gnu_SMP       ▼         ✓       Debug mode       ▼	
	Restore <u>D</u> efault	s <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.

type filter text	Build Properties		(+ + + + + + + + + + + + + + + + + + +
Resource	Specify all build		
Binary Parser Build Properties Builders ▷ C/C++ General Code Coverage Ana Project Info Project References Run/Debug Settings Task Tags Validation	Weild Support and Specific Sp	cs ∲Tools ≥Paths # Defines ALibraries \$ Va          Value         DKM         0	New       Edit       Rename       Copy       Delete
	Active build spec: PENT	TIUMgnu	
	Name (Common)	Value (Build spec specific)	New
	TOOL_FAMILY	gnu	Edit
	TOOL	gnu	8
	TOOL_PATH CC_ARCH_SPEC	-mtune=pentium -march=pentium -nostdl	Rename
	VSB_DIR	\$(WIND_BASE)/target/lib	
			Сору
	VSB_CONFIG_FILE	\$(VSB_DIR)/h/config/vsbConfig.h	Copy Delete
	VSB_CONFIG_FILE LIBPATH LIBS	\$(VSB_DIR)/h/config/vsbConfig.h -L\${NDDSHOME}/lib/pentiumVx6.9gcc4.3. -Inddscppz -Inddscz -Inddscorez	Copy Delete
	VSB_CONFIG_FILE LIBPATH LIBS	\$(VSB_DIR)/h/config/vsbConfig.h -L\${NDDSHOME}/lib/pentiumVx6.9gcc4.3. -Inddscppz -Inddscz -Inddscorez Restore Defaults	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	(⇒ + ⇒ + +
<ul> <li>Resource Binary Parser</li> <li>Build Properties</li> <li>Builders</li> <li>C/C++ General</li> <li>Code Coverage Ana</li> </ul>	Specify all build Build Support and Specs Paths # Defines Librarie Build spec specific settings Active build spec: PENTIUMgnu Include paths:	s 🕏 Variables
Project Info Project References	Include directories	Generate
Run/Debug Settings	-I\$(WIND_BASE)/target/h -I\$(WIND_BASE)/target/h/wrn/coreip	Edit
Validation	-I\${NDDSHOME}/include	Add
	-I\${NDDSHOME}/include/ndds	Add to all
		Remove
		Remove from all
		Up
		Down
	Redirection directory: PENTIUMgnu	Default
	Note: Leave this field blank to store build output together with th an absolute path (environment variables are permissible) t	browse ne sources, or enter o redirect the output.
	Restore <u>D</u> e	efaults <u>A</u> pply
?	Canc	el 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	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			
] M C⁺ 🖩 🕲 👜 🗍 🖬 🔩 😹 🔋   ≫* O*   🖋	• ] III ] \$ + \$ + \$ + \$ + \$			😭 📴 Basic De
🎦 Project Explorer 🛛 📄 😫 🐑 📰 🥃	🛛 👻 🗖 🕞 Getting Started 🛿 📄 creation.	log 📄 usrAppInit.c 📄 cre	eation.log	🗝 🖬 Outline 🛛 👘 🗖
NetworkImage (vxworks-cert-6.6.4.1 [not ins	stalled] II 📥 🏠 🗇 🔿			An outline is not available.
▶ ﷺ rti-slo-6.9.4.2-sbc8641d-v1 (Wind River VxWa	orks 6.9			
Samsung-iso-v1 (wrlinux-4 [not installed] Plate	Itform Pro Wind Riv	/er Workbench 3.3	🛨 What's New	
Samsung-iso-v2 (wrlinux-4 [not installed] Plate	itform Pre			
🕨 😂 samsung-iso-v3 (wrlinux-4 [not installed] Plat	tform Pre		AAA	
🕨 😂 samsung-v1 (wrlinux-4 [not installed] Platfon	m Projec			
test (vxworks-cert-6.6.4.1 [not installed] Ima	age Proje			
Test_Vx6ImageProject (vxworks-cert-6.6.4.1	[not inst	Getting Started		
🗢 🖑 testApplication (Wind River VxWorks 6.9 Dov	White data			
Build Targets (PENTIUMgnu - debug)	New Go Into	P		
🏇 Wind River Launches				
👂 🗱 Binaries	Open in New Window	g		
▶	Copy	Ctrl+C		
PENTIUMgnu	Paste	Delete		
HelloWorld_publisher.cxx	Attributes	Share		
HelloWorld_subscriber.cxx	Mo <u>v</u> e	Jindie	_	
le HelloWorld.cxx	Rena <u>m</u> e	F2 Planh		
🖪 HelloWorld.h	Import			
le HelloWorldPlugin.cxx	Export			
HelloWorldPlugin.h	Open Wind River VxWorks 6.9 Development St	nell		
HelloWorldSupport.cxx	Build Project	Ctrl+B Ctrl+P		
HelloWorldSupport.h	Rebuild Project			
<u>×</u>	Build Options	•		
📲 Remote Systems 🕱	Close Project	FD		
1 1 1 1 1 1 0 0 0 0 0	Close Unrelated Projects			
▶ 📑 Local	Project References	•		
LinuxHost_local (Host OS (Native Development)	ei Run VxWorks Kernel Task			
Wind River VxWorks 6.9)	Debug VxWorks Kernel Task			
Karrier Karrier (Wind River VxWorks 6.9)	Download VxWorks Kernel Task			
	Index	•		
	<u>B</u> un As	Search	😂 • 🗰 📄	- 🕹 😚 😫 🖉 🚮 🖳 🏹 🗖
	Debug As	jects/testA	Application/HelloWorld_publisher.o	
	Download Profile As	Ig/Objects/te iects/testA	estApplication/HelloWorld_subscriber.o Application/HelloWorld_subscriber.o	" " ]; then mkdir -p " dirname "test
	Team	g/testApplic	ation_partialImage.o"`" ]; then mkdir	-p "`dirname "testApplication_parti
	Comp <u>a</u> re With	tion out"`"	on_partialimage.o 1: then mkdir -n "`dirname "testApplica	tion/Debug/testApplication out"`"
	Restore from Local History	put	"······	·····,,,
	P <u>r</u> operties	Alt+Enter B-09-14 14	ication/PENTIUMgnu :51:34 (Flapsed Time: 00:50)	
			(Empsed Time: 00.50)	
	X			Þ
testApplication (Wind River VxWorks	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>.

# 4.3.3 Building and Running an Application as a Real-Time Process

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

- 4.3.3.1 Using the Command Line below
- 4.3.3.2 Using Workbench on page 31

### 4.3.3.1 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 *rtidds-gen*.

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 set to the correct version of VxWorks. For example:

```
wrenv.sh -p vxworks-6.9
```

- 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 DDS Getting</u> <u>Started Guide</u>.
- 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.
- 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\_publisher.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.

### 4.3.3.2 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 Wind I	River Workbench		
<u>File</u> <u>E</u> dit <u>Source</u> Refac <u>tor</u> <u>N</u> avigate Sea	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		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	•		
		E <u>x</u> ample	
Print	Ctrl+P	Other Chil	
· · · ·		Other Ctr	1+N

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

😣 💷 New Wind River W	orkbench Project			
Target Operating System	n			
Select the target operat	ing system for the proj	ect.		
Target operating system	Wind River VxWorks	69		
iaiget operating system	Host OS (Native Dev	elopment)	_	
?	< <u>B</u> ack <u>N</u> ex	t > Ca	ancel	<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	rnel Module		
	Kernel Libraries			
Description:	ROMFS File Syster	n		
	Real Time Process	Application		
	Relinkable Kernel	Module		
	Shared User Library			
	Static Kernel Libra	ary		
	Static User Librar	у		
-Setup inform	System Image			
Beculting pr	User-Defined			
Resulting pr	None			
Uses build t	ool: Linker			
		[]		1
(?)	< <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 4.3.1 Generate Example Code and Makefile with rtiddsgen on page 18 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 **Prop**erties.
- 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** dropdown 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 test	ApplicationRTP	
type filter text	Build Properties	<b>⇔</b> • ⇔• •
Resource	Define and undefine directives	Edit
Binary Parser	-DCPU=_VX_\$(CPU)	Add
Build Properties	-DTOOL_FAMILY=\$(TOOL_FAMILY)	
Builders	-DTOOL=\$(TOOL)	Add to all
✓ C/C++ General	-D_WRS_KERNEL	Remove
Code Style	-D_VSB_CONFIG_FILE=\"\$(VSB_DIR)/h/config/vsbConfig.h\"	Remove from all
Documentation	-DIP_PORT_VXWORKS=69	
File Types	-DRTI_VXWORKS	Up
P Indexer	-DRTI_STATIC	Down
Code Coverage Ana	-DRTI_RTP	
Project Into		
Project References		
Task Tags		
Validation		
Validation		
<		T
?		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	(→ + → + +
<ul> <li>Resource Binary Parser</li> <li>Build Properties</li> <li>Builders</li> <li>C/C++ General</li> <li>Code Coverage Ana</li> <li>Project Info</li> </ul>	Specify all build         ✓ Build Support and Specs	s \$ Variables
Project References Run/Debug Settings Task Tags Validation	-I\$(WIND_BASE)/target/h -I\$(WIND_BASE)/target/h/wrn/coreip -I\${NDDSHOME}/include -I\${NDDSHOME}/include/ndds	Edit Add Add to all Remove Remove from all Up Down
۲	Redirection directory:       PENTIUMgnu         Redirection root directory:       Image: Comparison of the store build output together with together with the store build output together with together wither with together with together with together wither with	Default Browse he sources, or enter to redirect the output. efaults <u>Apply</u>
?	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.
- 17. Run the application as described starting in Step 5 in the Command Line section above.

# 4.4 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 DDS* to send and receive "Ping" messages to other *rtiddsping* applications running on the same or different computers.

RTIDDS 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 DDS* 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 DDS Core</u> <u>Libraries Platform Notes</u>.