# Advanced Architectures in LabVIEW™ Exercises

**Course Software Version 2011**
**February 2012 Edition**
**Part Number 325183B-01**

National Instruments
Not For Distribution

**Copyright**
© 2009–2012 National Instruments Corporation. All rights reserved.
Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

For components used in USI (Xerces C++, ICU, HDF5, b64, Stingray, and STLport), the following copyright stipulations apply. For a listing of the conditions and disclaimers, refer to either the `USICopyrights.chm` or the *Copyrights* topic in your software.

**Xerces C++**. This product includes software that was developed by the Apache Software Foundation (`http://www.apache.org/`). Copyright 1999 The Apache Software Foundation. All rights reserved.

**ICU**. Copyright 1995–2009 International Business Machines Corporation and others. All rights reserved.

**HDF5**. NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities
Copyright 1998, 1999, 2000, 2001, 2003 by the Board of Trustees of the University of Illinois. All rights reserved.

**b64**. Copyright © 2004–2006, Matthew Wilson and Synesis Software. All Rights Reserved.

**Stingray**. This software includes Stingray software developed by the Rogue Wave Software division of Quovadx, Inc.
Copyright 1995–2006, Quovadx, Inc. All Rights Reserved.

**STLport**. Copyright 1999–2003 Boris Fomitchev

**Trademarks**
LabVIEW, National Instruments, NI, ni.com, the National Instruments corporate logo, and the Eagle logo are trademarks of National Instruments Corporation. Refer to the *Trademark Information* at `ni.com/trademarks` for other National Instruments trademarks.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

**Patents**
For patents covering National Instruments products/technology, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your media, or the *National Instruments Patent Notice* at `ni.com/patents`.

National Instruments
Not For Distribution

**Worldwide Technical Support and Product Information**

`ni.com`

**Worldwide Offices**

Visit `ni.com/niglobal` to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

**National Instruments Corporate Headquarters**

11500 North Mopac Expressway  Austin, Texas 78759-3504  USA  Tel: 512 683 0100

For further support information, refer to the *Additional Information and Resources* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `feedback`.

# Contents


## Student Guide

A. NI Certification ..... v
B. Course Description ..... vi
C. What You Need to Get Started ..... vii
D. Installing the Course Software ..... vii
E. Course Goals ..... viii
F. Course Conventions ..... viii

## Lesson 1
## Software Architecture – Introduction

Exercise 1-1 Design a Wind Farm ..... 1-1
Exercise 1-2 Analyzing a Design Pattern ..... 1-10
Exercise 1-3 Identifying Design Challenges ..... 1-13

## Lesson 2
## Designing an API

Exercise 2-1 Evaluating an API Design ..... 2-1

## Lesson 3
## Multiple Processes and Inter-Process Communication

Exercise 3-1 Queue-Driven Message Handler ..... 3-1
Exercise 3-2 JKI State Machine (Optional) ..... 3-3
Exercise 3-3 Spawning Multiple Instances of an Asynchronous Independent VI ... 3-5
Exercise 3-4 Fixing the Run VI Method ..... 3-9
Exercise 3-5 Using Single Element Queues (SEQ) ..... 3-10
Exercise 3-6 Using Data Value References ..... 3-19
Exercise 3-7 Functional Global Variables (Optional) ..... 3-21

## Lesson 4
## Advanced User Interface Techniques

Exercise 4-1 Creating an XControl (Optional) ..... 4-1
Exercise 4-2 Modifying X Listbox Abilities (Optional) ..... 4-3
Exercise 4-3 Creating X Listbox Properties and Methods (Optional) ..... 4-6
Exercise 4-4 Creating the X Listbox Facade VI (Optional) ..... 4-12

## Lesson 5
## Introduction to Object-Oriented Programming in LabVIEW

Exercise 5-1 LVOOP ..... 5-1


National Instruments Not For Distribution

## Lesson 6
## Plug-In Architectures

Exercise 6-1 Using Plug Ins with VI Server.....6-1
Exercise 6-2 Using LVOOP Plug Ins.....6-6

## Lesson 7
## Tips, Tricks, & Other Techniques

Exercise 7-1 Using Drop Ins.....7-1

## Lesson 8
## Error Handling

Exercise 8-1 Designing an Error Handling System.....8-1

## Appendix A
## Course Slides

Topics.....A-1

## Appendix B
## Additional Information and Resources

National Instruments
Not For Distribution

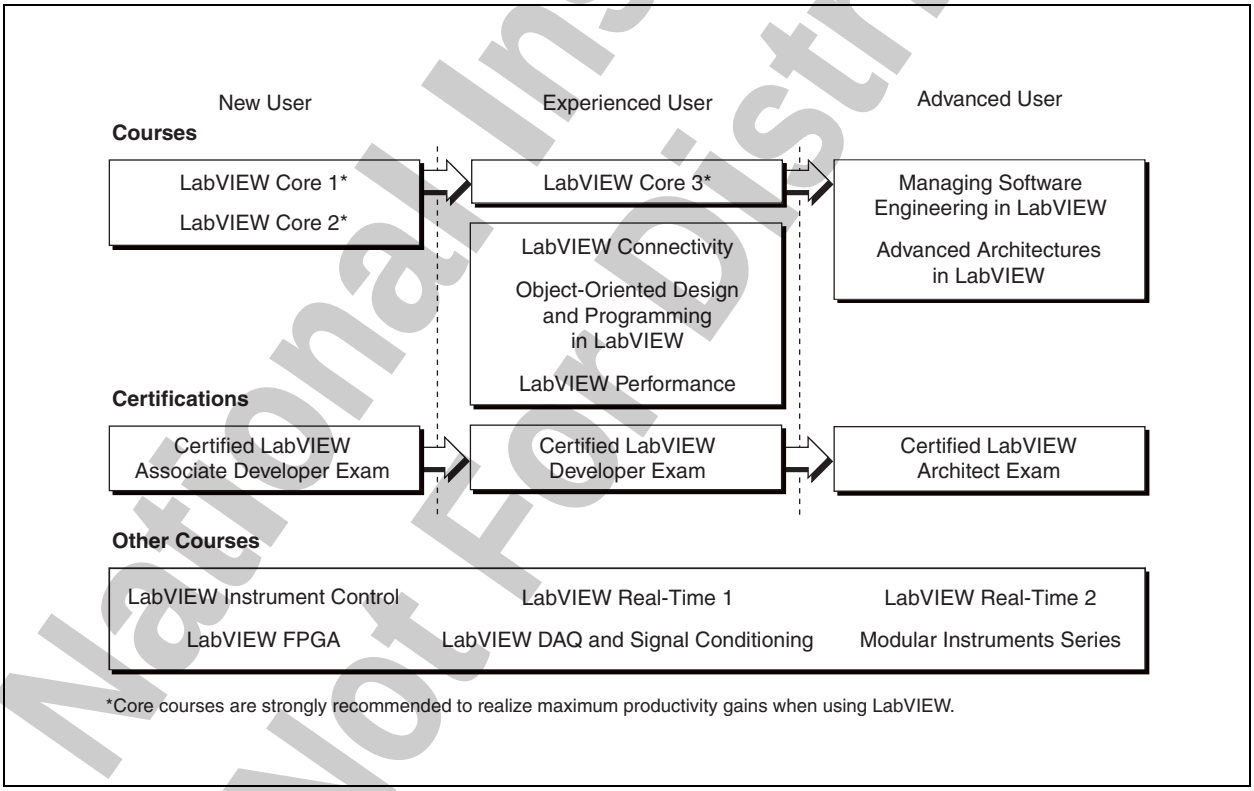
# Student Guide

Thank you for purchasing the *Advanced Architectures in LabVIEW* course kit. This course manual and the accompanying software are used in the three-day, hands-on *Advanced Architectures in LabVIEW* course.

You can apply the full purchase price of this course kit toward the corresponding course registration fee if you register within 90 days of purchasing the kit. Visit `ni.com/training` to register for a course and to access course schedules, syllabi, and training center location information.

## A. NI Certification

The *Advanced Architectures in LabVIEW* course is part of a series of courses designed to build your proficiency with LabVIEW and help you prepare for exams to become an NI Certified LabVIEW Developer and NI Certified LabVIEW Architect. The following illustration shows the courses that are part of the LabVIEW training series. Refer to `ni.com/training` for more information about NI Certification.




National Instruments
Not For Distribution

# B. Course Description

In the *Advanced Architectures in LabVIEW* course you participate in discussions and work independently and collaboratively to learn how to architect an application and then design the components to support the architecture.

This course assumes you have taken the *LabVIEW Core 3* course or have equivalent experience.

The course is divided into lessons, each covering a topic or a set of topics. Each lesson consists of the following parts:

- An introduction that describes what you will learn.
- A discussion of the topics.
- A set of exercises that reinforces the topics presented in the discussion.

  Some lessons include optional exercises or challenge steps to complete if time permits.
- A summary that outlines important concepts and skills taught in the lesson.

**Note** For course manual updates and corrections, refer to `ni.com/info` and enter the Info Code `aalerrata`.

National Instruments
Not For Distribution

## C. What You Need to Get Started

Before you use this course manual, make sure you have the following items:

- ❑ Computer running Windows 7/Vista/XP
- ❑ NI LabVIEW 2011
- ❑ Advanced Architectures in LabVIEW CD, which contains the following files:

| Directory | Description |
|---|---|
| `Exercises` | Folder for saving VIs created during the course and for completing certain course exercises |
| `Solutions` | Folder containing the solutions to all the course exercises |
| `Advanced Architectures in LabVIEW 2011 - Course Manual.pdf` | *Advanced Architectures in LabVIEW* Course manual. |

## D. Installing the Course Software

Complete the following steps to install the course software.

1. Insert the course CD in your computer.
2. Install the Exercises and Solutions files to the desired location.

**Tip** Folder names in angle brackets, such as `<Exercises>`, refer to folders on the root directory of your computer.

National Instruments
Not For Distribution

## E. Course Goals

After completing this course you will be able to:

- Design a software architecture to be implemented in LabVIEW
- Design a consistent, organized, and usable API
- Analyze and evaluate several solutions to a problem
- Use advanced design patterns and techniques to build the components or subsystems of an architecture
- Understand the design trade-offs when selecting an advanced design pattern or technique
- Analyze, critique, and improve the architecture of a LabVIEW application

## F. Course Conventions

The following conventions are used in this course manual:

**»** The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **Options»Settings»General** directs you to pull down the **Options** menu, select the **Settings** item, and select **General** from the last dialog box.

This icon denotes a tip, which alerts you to advisory information.

This icon denotes a note, which alerts you to important information.

**bold** Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

*italic* Italic text denotes variables, emphasis, a cross-reference, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.

`monospace` Text in this font denotes text or characters that you enter from the keyboard, sections of code, programming examples, and syntax examples. This font also is used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

National Instruments
Not For Distribution

# 1

# Software Architecture – Introduction

## Exercise 1-1 Design a Wind Farm

### Goal

To draft the initial software architecture for the Wind Farm course project.

### Scenario

LabVIEW is a programming language that facilitates the rapid prototyping and development of code. As such many users of LabVIEW have developed a “code and fix” model for constructing software. The proper process, for smaller as well as larger projects, is to first design a software architecture. The developer should specify the major components of the project and how those components communicate with one another. Data structures should also be defined.

This lesson provides you the opportunity to begin with a requirements document and sketch out a preliminary software architecture. You will have the opportunity to compare your architecture with your fellow students. You will also explore several implementations of the key components of the Wind Farm.

Additionally, this lesson allows you to experience a process that is similar to what you will encounter if you take the Certified LabVIEW Architect Exam.

### Implementation

Refer to the next section, *Wind Farm Course Project Requirements Document*, to begin designing the software architecture for the Wind Farm course project.

Your design may include diagrams and text. You should first consider the architecture, regardless of the programming language of the implementation. The architecture should be clear to C, Java, and LabVIEW developers. Use your remaining time to consider the LabVIEW implementation. Which design patterns will you use? Which communication components might be more difficult to implement?

National Instruments
Not For Distribution

# Wind Farm Course Project Requirements Document

V I Engineering's Chief Architect Christopher G. Relf contributed to the following requirements example. This example represents part of a deliverable from a V I Engineering led Requirements Engineering phase and shows how LabVIEW architects can assist clients in documenting requirements.

# 1. Introduction

## 1.1. Purpose

The purpose of the Systems Requirements Specification (SyRS) is to enumerate the functions of the system and define in detail how the system interfaces to both surrounding systems and its users. This document is intended for both the customer and the developers implementing the system.

## 1.2. Definitions of Terms

This section provides the definitions of specific terms used to construct this SyRS.

- The word SHALL denotes a mandatory requirement. Departure from such a requirement is not permissible without formal agreement.
- The word SHOULD denotes a recommendation or advice on implementing such a requirement of the document.
- The word WILL denotes a provision or service, or an intention in connection with a requirement.
- The word MAY denotes a permissible practice or action. It does not express a requirement.

## 1.3. Product Description

The course project has been designed to give students the opportunity to design and evaluate several potential architectures for a larger LabVIEW application. Students will begin an initial design of the Wind Farm at the beginning of the course and then consider the implementation, utilizing programming techniques that they currently understand. Then several advanced design patterns will be taught and discussed. The merits of each will be analyzed as they apply to the overall course project.

A wind farm simulation was chosen as the course project because it is a relatively straight forward concept to understand. Key parameters that must be monitored are obvious. The number of wind turbines in operation is a variable that is only known at runtime. More over, multiple processes must run asynchronously and this can pose a challenge when designing a software architecture, even in LabVIEW.

National Instruments Not For Distribution

The Wind Farm Course Project is simply an application with which to illustrate concepts that are taught in the course. Each and every concept can easily be applied to your application that you are developing at your workplace. In every case, either the results of the exercise or the process utilized in the exercise is one which you can leverage outside of the course immediately.

## 1.4. Assumptions

### 1.4.1. Simplifying a Complex System

Each module or system that will be coded or used in the course project may appear to be quite simple. Students should assume that each of these systems actually represents a complex process that might take weeks or months to develop. For the purposes of this course, a very complex system was significantly simplified.

### 1.4.2. Vague User Interface Requirements

A clearly defined user interface is not included in the wind farm software requirements. Only general features have been specified. This was crafted intentionally to allow the students flexibility in assessing several different designs that have different ramifications for the user interface

### 1.4.3. Additional Requirements

A typical requirements specification would include much more information than is noted below. The wind farm requirements constitute a subset that might be classified as interface requirements, functional requirements and design constraints. A well researched and carefully crafted requirements specification would include many additional categories such as: hardware interfaces, communication interfaces, networking with other systems, safety issues, constraints, documentation, and much more. Additionally, each requirement should have a unique ID, a priority, and an identified user. For the purpose of this exercise, you may assume that all requirements must be implemented and the user is the operator. For more information on writing requirements refer to the following:

- IEEE Std 1233, 1998 and 1233-1996 aIEEE Guide for Developing System Requirements Specifications -Description
- *Managing Software Engineering in LabVIEW*
- *Large LabVIEW Application Development* Community Page on `ni.com`
- *Writing Better Requirements* by Alexander and Stevens

National Instruments Not For Distribution

## 1.5. Product Description

A wind farm is comprised of many wind turbines. Each wind turbine is a sophisticated system that measures wind speed, wind direction and many operational parameters related to rotating machinery. The optimum energy production occurs when the turbine rotates at a specified speed. The angle of attack and the direction of the turbine are adjusted by the control system to take advantage of the current wind conditions. This course project does not include an actual simulation or the complexities of a typical control system. Additionally, the course project runs on one target. An actual wind farm is a sophisticated, networked system.

# 2. Product Requirements

## 10-0000 Interface Requirements

### 11-0000 Operator Interfaces

#### 11-1000 General

11-1001 The user interface shall include a simple display of the operation of the wind turbines.

11-1002 The user interface will be intuitive and require minimal training.

11-1003 The user interface shall not contain tabs.

#### 11-2000 Wind Turbine Control and Data Visualization

11-2001 The user interface shall provide the capability for an operator to add a wind turbine to the display.

11-2002 The user interface shall provide the capability for an operator to display the turbine rotation speed from each wind turbine.

11-2003 The user interface may provide the capability for an operator to stop a wind turbine.

11-2004 The user interface may provide the capability for an operator to restart a wind turbine.

11-2005 The user interface may provide the capability for an operator to remove a wind turbine from the display.

11-2006 The user interface may provide the capability for an operator to stop all displayed wind turbines.

National Instruments
Not For Distribution

#### 11-3000 Software Shutdown

11-3001 The user interface shall provide the capability for an operator to shut the system down.

11-3002 The user interface may display a message on successful shutdown.

## 20-0000 Functional Requirements

### 21-0000 Main Software

#### 21-1000 Data Acquisition

21-1001 The system shall acquire each wind turbine’s rotational speed in RPM.

21-1002 The system may acquire each wind turbine's blade angle of attack in degrees relative to the support shaft.

21-1003 The system may acquire each wind turbine’s generated power in Watts.

21-1004 The system may acquire each wind turbine’s energy generation.

#### 21-2000 Data Logging

21-2001 The test software shall create a data log for each running wind turbine.

21-2002 The data logs shall contain the name of the wind turbine tested.

21-2003 The data logs shall contain the time and date when each turbine was started.

21-2004 The data logs shall contain the rotation speed of each wind turbine at 250 msec intervals.

21-2005 The data logs shall be saved to disk in one of the following user-specified formats:

- XML
- Tab-Delimited ASCII Text

21-2006 The data logs may also need to be saved to disk in additional formats, such as TDMS, in the future.

#### 21-3000 Wind Turbine Simulation

21-3001 A wind turbine shall receive the wind speed input.

21-3002 A wind turbine shall output the corresponding rotation speed.

National Instruments Not For Distribution

# 30-0000 Design Constraints

## 31-0000 Operator Interfaces

### 31-1000 General

31-1001 The test system software shall be limited to one physical screen.

31-1002 The user interface shall be of a maximum resolution of 1280 x 1024.

National Instruments
Not For Distribution

National Instruments
Not For Distribution

National Instruments
Not For Distribution

National Instruments
Not For Distribution

**End of Exercise 1-1**

# Exercise 1-2 Analyzing a Design Pattern

## Goal

To analyze the Producer/Consumer Design Pattern (Events).

## Scenario

The Producer/Consumer Design Pattern (Events) is a foundational starting point for many subcomponents for any software architecture. However, several changes should be made to the template before it becomes a robust starting point for component development. National Instruments provides a basic pattern that is generic to any application or use case. Further customization of the template to handle unique requirements for each programming team saves considerable time and effort during software development. The architect, with assistance from a larger team when available, must determine the needed changes.

## Discussion Questions

Open the Producer/Consumer Design Pattern (Events) from the template browser and answer the following questions:

- ❑ When might you use this design pattern?

National Instruments
Not For Distribution

- ❑ What is missing from this design pattern? What changes must be made every single time the pattern is used in any application?

- ❑ What common tasks used by most LabVIEW developers should be included in a template based on this design pattern?

- ❑ What unique tasks for your application should be included in a template based on this design pattern?

National Instruments
Not For Distribution

## Discussion Answers

- When might you use this design pattern?

  *You might use this design pattern when you need a producer loop to handle events and maintain a responsive user interface and a parallel consumer loop to asynchronously handle the processing of the event and its data.*

- What is missing from this design pattern? What changes must be made every single time the pattern is used in any application?

  *The producer loop will not know if the consumer loop stopped or generated an error and stopped dequeueing elements.*

  *The design pattern lacks a safe way to stop the consumer loop. Currently, the design pattern stops the consumer loop by releasing the queue, which causes the consumer loop to stop even if there are still elements left in the queue.*

  *The queue data should be made into a typedef to make the code more maintainable.*

- What common tasks used by most LabVIEW developers should be included in a template based on this design pattern?

  *You should include common tasks like Initialize, Shutdown, Panel Close event that stops the application in the development environment and exits the application in the run-time environment, Error Handling, Handle the dequeued data, and so on.*

### End of Exercise 1-2

National Instruments
Not For Distribution

# Exercise 1-3 Identifying Design Challenges

## Goal

To list three design challenges in an application that you are currently working on or that you have worked on in the past (not the Wind Farm Course Project).

## Scenario

In this course, you learn several advanced programming techniques as well as some tips and tricks. In order to facilitate integrating the ideas into your applications, you must first consider key challenges and periodically return to this list to determine how the technique could apply.

## Implementation

Take a moment to think about your current application or a recent project. Consider data structures, modules, processes, and behavior. Determine which aspects pose the greatest challenge when designing an architecture. What were or are the greatest coding challenges? Identify and document the top three issues in that application.

**Note** Common challenges include deciding what will run synchronously and what will be asynchronous, designing the asynchronous communication mechanism, or crafting cohesive modules such that changes have limited affects on the rest of the code and minimal coding is needed to make those changes.

1.

2.

3.

**End of Exercise 1-3**

National Instruments Not For Distribution

# Notes

National Instruments
Not For Distribution

# 2

# Designing an API

## Exercise 2-1 Evaluating an API Design

### Goal

To review the design decisions behind the STM API.

### Implementation

The installer for the Simple Messaging Reference Library (STM) is included on your course CD. The installer places the VIs in a subpalette under `user.lib`.

1. Install the STM Reference Library.
   - ❑ Navigate to the `<Exercises>\Advanced Architectures in LabVIEW\Adv Design Patterns and Tools\Installer for STM` and run `setup.exe`.

**Note** The VIs used in this exercise are automatically installed to `<Program Files>\National Instruments\LabVIEW 2011\user.lib`.

2. If LabVIEW is open, you must close and relaunch it in order to access the STM Reference Library.
3. Use and examine the STM Reference Library.
   - ❑ Open a blank VI and view the block diagram.
   - ❑ From the palette, place the **User Library»STM»Examples» Basic Client-Server»STM Basic Client Example.vi** on the block diagram.
   - ❑ From the palette, place the **User Library»STM»Examples» Basic Client-Server»STM Basic Server Example.vi** on the block diagram.
   - ❑ View the front panel for both VIs.
   - ❑ Run the STM Basic Server Example VI. You must run the server first.

National Instruments Not For Distribution

- ❑ Run the STM Basic Client Example VI. Communication is on the local host. If you are in a classroom with networked computers, you can run this example on two computers.

**Note** To run this example on two computers, you must determine the IP address of the computer running `STM Basic Server Example.vi`. From the Windows Start menu, open a Windows Command Prompt and enter `ipconfig`. Press <Enter> to run the command which displays the IP address.

- ❑ Stop the VIs.
- ❑ Open the block diagram for both VIs and notice the design decisions that were made.

## Discussion Questions

1. In what ways are the icons clear and meaningful?

2. How has connector pane selection facilitated the use of this API?

3. How does the choice in label names make the VIs easy to use?

4. How has the use of polymorphic VIs made this library easy to use?

National Instruments
Not For Distribution

5. For any of the previous questions, what improvements could be made?

6. How might this library change in the future?

National Instruments
Not For Distribution

## Discussion Answers

1. In what ways are the icons clear and meaningful?

   **The VI icons have a common STM banner to show that they are part of the same API.**

   **The icons have meaningful graphics, such as arrows to indicate the direction of the communication. The text in the icons indicates whether the polymorphic VI is using the TCP, UDP, or Serial instance.**

2. How has connector pane selection facilitated the use of this API?

   **The connector panes consistently use the lower terminals for error wires and the upper terminals for the connection information**

3. How does the choice in label names make the VIs easy to use?

   **The label names all have a common STM or TCP prefix. The label names also very clearly describe the VI's functionality (i.e. STM Read Message, STM Read Meta Data).**

4. How has the use of polymorphic VIs made this library easy to use?

   **You can use the same polymorphic VIs to implement TCP, UDP, or serial communication. The polymorphic VIs automatically adapt based on whether you wire a TCP connection ID, UDP connection ID, VISA resource, or an existing Connection Info cluster to the top-left input terminal.**

5. For any of the previous questions, what improvements could be made?

   **You can add a block diagram comment to the example VIs explaining how to use the polymorphic VIs. It is not obvious that the STM API contains polymorphic VIs and UDP, TCP, and serial implementations. You can also add a separate example for a UDP, TCP, and serial implementation.**

6. How might this library change in the future?

   **You might add more protocols in addition to UDP, TCP, and serial.**

## End of Exercise 2-1

National Instruments
Not For Distribution

# Notes

National Instruments
Not For Distribution

# Notes

National Instruments
Not For Distribution