

Bitter, Rick et al "Frontmatter"  
*LabVIEW Advanced Programming Techinques*  
Boca Raton: CRC Press LLC,2001

# **LABVIEW ADVANCED PROGRAMMING TECHNIQUES**

**Rick Bitter**

**Taqi Mohiuddin**

**Matt Nawrocki**



**CRC Press**

**Boca Raton New York London Tokyo**

---

# Preface and Acknowledgments

As the power of the standard personal computer has steadily evolved, so have the capabilities of LabVIEW. LabVIEW has simplified the working lives of thousands of scientists, engineers, and technicians, and has increased their productivity. Automation has reduced the costs and increased the manufacturing outputs of factories around the world. Cycle times for product development have been shortened and the quality of many products has steadily improved. LabVIEW does not get credit for all of these improvements, but has without question played a valuable role in accomplishing these goals in many organizations.

In our earlier experiences with LabVIEW, we found that adequate coverage of key topics was lacking. Subjects that are useful to users without a formal background in computer science, such as approaches to software development, exception handling, and state machines, were very difficult to find. In addition, newer areas such as multithreading and ActiveX are even harder to locate, and sometimes documentation is nonexistent. Part of our intent in this book is to cover these topics that are difficult to find in other books on LabVIEW.

The chapters in this book are written in a manner that will allow readers to study the topic of interest without having to read the contents in sequential order. Users of LabVIEW with varying levels of expertise will find this book beneficial.

Proficiency with a programming language requires an understanding of the language constructs and the tools needed to produce and debug code. The first two chapters provide an overview of LabVIEW's Integrated Development Environment, programming constructs, and main features. These chapters are meant to supplement LabVIEW's documentation, and provide good background information for programmers new to the language.

Effective programmers have an understanding of programming techniques that are applicable to a large number of programming problems. Programming tools such as state machines that simplify logic of handling various occurrences, and the use of instrument drivers are two such programming tools. Exception handling is left out of more applications than we want to discuss (including some of our own), but we have included a chapter specifically on exception handling in LabVIEW.

Advanced programmers understand the operation of the language they are working with and how it interacts with the system. We present a chapter on multithreading's impact on LabVIEW. Version 5.0 was LabVIEW's debut into the world of multithreaded-capable programming languages. A number of the issues that occur with multithreading programming were abstracted from the programmer, but a working knowledge of multithreaded interactions is needed.

Object Oriented Programming (OOP) is commonly employed in languages such as C++ and Java. LabVIEW programmers can realize some of the benefits to such

an approach as well. We define key terms often used in OOP, give an explanation of object analysis, and introduce the application of these concepts within a LabVIEW environment.

Finally, we present two chapters on ActiveX. An explanation of related technologies such as Component Object Model (COM) and Object Linking and Embedding (OLE) is provided, along with the significance of ActiveX. A description on the use of ActiveX in LabVIEW applications is then provided. We follow this up with several useful examples of ActiveX, such as embedding a browser on the front panel, use of the tree view control, and automating tasks with Microsoft Word, Excel, and Access.

This book would not have been possible without the efforts of many individuals. First, we want to thank our friends at National Instruments. Ravi Marawar was invaluable in his support for the completion of this book. We would also like to thank Norma Dorst and Steve Rogers for their assistance.

Our publisher at CRC Press, Dawn Mesa, has provided us with guidance from the first day we began working on this book until its completion. This was truly helpful considering this is our first book. Thanks also go out to Felicia Shapiro, our editor, who was very understanding and flexible.

A special thanks to Tim Sussman, our colleague and friend. He came through for us at times when we needed him. Also thanks to Greg Stehling, John Gervasio, Jeff Hunt, Ron Wegner, Joe Luptak, Mike Crowley, the Tellabs Automation team (Paul Mueller, Kevin Ross, Bruce Miller, Mark Yedinak, and Purvi Shah), and Waj Hussain (if it weren't for Waj, we would have never written the papers which got us to writing this book).

Finally, we owe many thanks for the love and support of our families. They had to put up with us during the many hours spent on this book. Thank you, moms and dads: Auradker and Mariam Mohiuddin, Rich and Madalyn Bitter, Barney and Veronica Nawrocki. For moral support we thank Jahanara, Mazhar, Tanweer, Faheem, and Firdaus; Matt Bitter, Andrea and Jerry Lehmacher; Sheila Boyle, Andy, Corinne, Mark, and Colleen Nawrocki, Sue and Steve Fechtner.

---

# The Authors

Rick Bitter graduated from the University of Illinois at Chicago in 1994. He has presented papers at Motorola and National Instruments-sponsored symposiums. Rick currently develops performance testing applications in Motorola's Wireless Data Solutions Engineering department as a Senior Software Engineer.

Taqi Mohiuddin graduated in Electrical Engineering from the University of Illinois at Chicago in 1995. He obtained his MBA from DePaul University. He has worked with LabVIEW since 1995, beginning with version 3.1, ranging in various telecommunications applications. He is presently a Senior Engineer working with the Product Integration Test department at Motorola. He has presented papers on LabVIEW at Motorola and National Instruments conferences.

Matt Nawrocki graduated from Northern Illinois University in 1995. He has written papers and has done presentations on LabVIEW topics at Motorola, National Instruments, and Tellabs. Matt currently works in the Broadband Media Group at Tellabs, where he writes test automation software for the Cablesan test organization.

---

# Contents

## **1. INTRODUCTION TO LABVIEW**

- 1.1 Virtual Instruments
  - 1.1.1 The Front Panel
  - 1.1.2 Block Diagram
  - 1.1.3 Executing VIs
  - 1.1.4 LabVIEW File Extensions
- 1.2 Help
  - 1.2.1 Built-in Help
  - 1.2.2 Web Sites
- 1.3 Data Flow Programming
- 1.4 Menus and Palettes
- 1.5 Front Panel Controls
  - 1.5.1 Numeric
  - 1.5.2 Boolean
  - 1.5.3 String & Table
  - 1.5.4 List & Ring
  - 1.5.5 Array & Cluster
  - 1.5.6 Graphs and Charts
  - 1.5.7 Path & Refnum
- 1.6 Block Diagram Functions
  - 1.6.1 Structures
  - 1.6.2 Numeric, Boolean, String, and Comparison
  - 1.6.3 Array and Cluster
  - 1.6.4 Time & Dialog
  - 1.6.5 File I/O
  - 1.6.6 Instrument I/O, Data Acquisition, and Communication
  - 1.6.7 Creating Connectors
  - 1.6.8 Editing Icons
  - 1.6.9 Using SubVIs
  - 1.6.10 VI Setup
  - 1.6.11 Hierarchical Nature
- 1.7 Setting Preferences
  - 1.7.1 Paths
  - 1.7.2 Block Diagram
  - 1.7.3 History
  - 1.7.4 VI Server and Web Server
  - 1.7.5 Palettes

Bibliography

## **2. LABVIEW FEATURES**

- 2.1 Global and Local Variables
  - 2.2 Customizing Controls
    - 2.2.1 Custom Controls
    - 2.2.2 Type Definitions
    - 2.2.3 Strict Type Definitions
  - 2.3 Attribute Nodes
  - 2.4 Reentrant VIs
  - 2.5 Libraries (.LLB)
  - 2.6 File Manager
  - 2.7 Web Server
  - 2.8 Web Document Tool
  - 2.9 Instrument Wizard
  - 2.10 Profile Window
  - 2.11 Auto SubVI Creation
  - 2.12 Graphical Comparison Tools
    - 2.12.1 Compare VIs
    - 2.12.2 Compare VI Hierarchies
    - 2.12.3 SCC Compare Files
  - 2.13 Report Generation Palette
  - 2.14 Application Builder
  - 2.15 Sound VIs
  - 2.16 Application Control
    - 2.16.1 VI Server VIs
    - 2.16.2 Menu VIs
    - 2.16.3 Help VIs
    - 2.16.4 Other Application Control VIs
  - 2.17 Advanced Palette
    - 2.17.1 Data Manipulation
    - 2.17.2 Calling External Code
    - 2.17.3 Synchronization
  - 2.18 Source Code Control
    - 2.18.1 Configuration
    - 2.18.2 Adding and Modifying Files
    - 2.18.3 Advanced Features
  - 2.19 Graphs
    - 2.19.1 Standard Graphs
    - 2.19.2 3-D Graphs
    - 2.19.3 Picture Graphs
  - 2.20 Data Logging
  - 2.21 Find Feature
  - 2.22 Print Documentation
  - 2.23 VI History
  - 2.24 Key Navigation
- Bibliography

### **3. STATE MACHINES**

- 3.1 Introduction
  - 3.1.1 State Machines in LabVIEW
  - 3.1.2 When to Use a State Machine
  - 3.1.3 Types of State Machines
- 3.2 Enumerated Types and Type Definitions
  - 3.2.1 Type Definitions Used with State Machines
  - 3.2.2 Creating Enumerated Constants and Type Definitions
  - 3.2.3 Converting between Enumerated Types and Strings
  - 3.2.4 Drawbacks to Using Type Definitions and Enumerated Controls
- 3.3 Sequence-Style State Machine
  - 3.3.1 When to Use a Sequence-Style State Machine
  - 3.3.2 Example
- 3.4 Test Executive-Style State Machine
  - 3.4.1 When to Use a Test Executive-Style State Machine
  - 3.4.2 Recommended States for a Test Executive State Machine
  - 3.4.3 Determining States for Test Executive State Machines
  - 3.4.4 Example
- 3.5 Classical-Style State Machine
  - 3.5.1 When to Use a Classical-Style State Machine
  - 3.5.2 Example
- 3.6 Queued-Style State Machine
  - 3.6.1 When to Use the Queued-Style State Machine
  - 3.6.2 Example Using LabVIEW Queue Functions
  - 3.6.3 Example Using an Input Array
- 3.7 Drawbacks to Using State Machines
- 3.8 Recommendations and Suggestions
  - 3.8.1 Documentation
  - 3.8.2 Ensure Proper Setup
  - 3.8.3 Error and Close States
  - 3.8.4 Status of Shift Registers
  - 3.8.5 Typecasting an Index to an Enumerated Type
  - 3.8.6 Make Sure You Have a Way Out
- 3.9 Problems
  - 3.9.1 The Blackjack Example
  - 3.9.2 The Test Sequencer Example
  - 3.9.3 The PC Calculator Example

#### Bibliography

### **4. APPLICATION STRUCTURE**

- 4.1 Planning
- 4.2 Purpose of Structure
- 4.3 Software Models
  - 4.3.1 The Waterfall Model
  - 4.3.2 The Spiral Model



- 4.3.3 Block Diagrams
  - 4.3.4 Description of Logic
  - 4.4 Project Administration
  - 4.5 Documentation
    - 4.5.1 LabVIEW Documentation
    - 4.5.2 Printing LabVIEW Documentation
    - 4.5.3 VI History
  - 4.6 The Three-Tiered Structure
  - 4.7 Main Level
    - 4.7.1 User Interface
    - 4.7.2 Exception-Handling at the Main Level
  - 4.8 Second Level — Test Level
  - 4.9 Bottom Level — Drivers
  - 4.10 Style Tips
    - 4.10.1 Sequence Structures
    - 4.10.2 Nested Structures
    - 4.10.3 Drivers
    - 4.10.4 Polling Loops
    - 4.10.5 Array Handling
  - 4.11 Summary
- Bibliography

## **5. DRIVERS**

- 5.1 Communication Standards
  - 5.1.1 GPIB
  - 5.1.2 Serial Communications
  - 5.1.3 VXI Discussion
  - 5.1.4 VISA Definition
  - 5.1.5 DDE
  - 5.1.6 OLE
  - 5.1.7 TCP/IP
  - 5.1.8 DataSocket
  - 5.1.9 DAQ
  - 5.1.10 File I/O
  - 5.1.11 Code Interface Node and Call Library Function
- 5.2 Driver Classifications
  - 5.2.1 Configuration Drivers
  - 5.2.2 Measurement Drivers
  - 5.2.3 Status Drivers
- 5.3 Inputs/Outputs
- 5.4 Error Handling
- 5.5 NI Spy
  - 5.5.1 NI Spy Introduction
  - 5.5.2 Configuring NI Spy
  - 5.5.3 Running NI Spy

- 5.6 Driver Guidelines
- 5.7 Reuse and Development Reduction
- 5.8 Driver Example
- 5.9 IVI Drivers
  - 5.9.1 Five Classes of IVI Drivers
  - 5.9.2 Interchangeability
  - 5.9.3 Simulation
  - 5.9.4 State Management
  - 5.9.5 IVI Driver Installation
  - 5.9.6 IVI Configuration
  - 5.9.7 How to Use IVI Drivers
  - 5.9.8 IVI Virtual Bench
  - 5.9.9 IVI Driver Example
- Bibliography

## **6. EXCEPTION HANDLING**

- 6.1 Exception Handling Defined
- 6.2 Types of Errors
  - 6.2.1 I/O Errors
  - 6.2.2 Logical Errors
- 6.3 Built-In Error Handling
  - 6.3.1 Error Cluster
  - 6.3.2 Error Codes
  - 6.3.3 VISA Error Handling
  - 6.3.4 Simple Error Handler
  - 6.3.5 General Error Handler
  - 6.3.6 Find First Error
- 6.4 Performing Exception Handling
  - 6.4.1 When?
  - 6.4.2 Exception Handling at Main Level
  - 6.4.3 Programmer-Defined Errors
  - 6.4.4 Managing Errors
  - 6.4.5 State Machine Exception Handling
  - 6.4.6 Logging Errors
  - 6.4.7 External Error Handler
  - 6.4.8 Proper Exit Procedure
  - 6.4.9 Exception Handling Example
- 6.5 Debugging Code
  - 6.5.1 Error List
  - 6.5.2 Execution Highlighting
  - 6.5.3 Single-Stepping
  - 6.5.4 Probe Tool
  - 6.5.5 Breakpoint Tool
  - 6.5.6 Suspending Execution
  - 6.5.7 Data Logging

- 6.5.8 NI Spy/GPIB Spy
- 6.5.9 Utilization of Debugging Tools
- 6.6 Summary
- Bibliography

## **7. ACTIVEX**

- 7.1 Introduction to OLE, COM, and ActiveX
  - 7.1.1 Definition of Related Terms
- 7.2 COM
  - 7.2.1 The Variant
  - 7.2.2 Problems that COM Addresses
  - 7.2.3 In-Process and Out-of-Process COM Objects
  - 7.2.4 COM Object Identification
  - 7.2.5 How COM Objects Are Called and Used
- 7.3 OLE
  - 7.3.1 Origins and Applications
- 7.4 ActiveX
  - 7.4.1 Description of ActiveX
  - 7.4.2 ActiveX Definitions
  - 7.4.3 Events
  - 7.4.4 Containers
  - 7.4.5 How ActiveX Controls Are Used
- 7.5 LabVIEW and ActiveX
  - 7.5.1 The LabVIEW ActiveX Container
  - 7.5.2 The ActiveX/OLE Palette
  - 7.5.3 Event Support in LabVIEW 5.1
  - 7.5.4 LabVIEW as ActiveX Server
- 7.6 The VI Server
- Bibliography

## **8. ACTIVEX EXAMPLES**

- 8.1 Common Dialog Control
- 8.2 Progress Bar Control
- 8.3 Microsoft Calendar Control
- 8.4 Web Browser Control
- 8.5 Microsoft Scripting Control
- 8.6 Microsoft Winsock Control
  - 8.6.1 Using Winsock Control with TCP
  - 8.6.2 Using Winsock Control with UDP
  - 8.6.3 Using Winsock in Client Applications
  - 8.6.4 Using Winsock in Server Applications
  - 8.6.5 Using Winsock for Multiple-Connection Servers
- 8.7 Microsoft System Information Control
- 8.8 Microsoft MAPI Services
- 8.9 MAPI Messages Control

- 8.10 Microsoft Status Bar Control
- 8.11 Microsoft Tree View Control
- 8.12 Microsoft Agent
  - 8.12.1 Request Objects — First Tier
  - 8.12.2 Other First-Tier Controls
  - 8.12.3 The Characters Object
  - 8.12.4 The Character Control
- 8.13 Registry Editing Control
- 8.14 Controlling Microsoft Word
- 8.15 Microsoft Access Control
- 8.16 Controlling LabVIEW from Other Applications
- 8.17 Understanding ActiveX Error Codes
- 8.18 Controls that Do Not Work Well with LabVIEW
- 8.19 Advanced ActiveX Details

## **9. MULTITHREADING IN LABVIEW**

- 9.1 Multithreading Terminology
  - 9.1.1 Win32
  - 9.1.2 UNIX
  - 9.1.3 Multitasking
  - 9.1.4 Kernel Objects
  - 9.1.5 Thread
  - 9.1.6 Process
  - 9.1.7 Application
  - 9.1.8 Priority
  - 9.1.9 Security
  - 9.1.10 Thread Safe
- 9.2 Thread Mechanics
  - 9.2.1 Thread States
  - 9.2.2 Scheduling Threads
  - 9.2.3 Context Switching
- 9.3 Win32 Multithreading
- 9.4 Pthreads
- 9.5 Multithreading Problems
  - 9.5.1 Race Conditions
  - 9.5.2 Priority Inversion
  - 9.5.3 Starvation
  - 9.5.4 Deadlocking
  - 9.5.5 Operating System Solutions
- 9.6 Multithreading Myths
  - 9.6.1 The More Threads, the Merrier
  - 9.6.2 Always Makes My Program Run Faster
  - 9.6.3 Makes Applications More Robust
  - 9.6.4 Conclusion on Myths
- 9.7 Multithreaded LabVIEW
  - 9.7.1 Execution Subsystems

- 9.7.2 The Run Queue
- 9.7.3 DLLs in Multithreaded LabVIEW
- 9.7.4 Customizing the Thread Configuration
- 9.8 Thread Count Estimations for LabVIEW
  - 9.8.1 Same as Caller or Single Subsystem Applications
  - 9.8.2 Multiple Subsystem Applications
  - 9.8.3 Optimizing VIs for Threading
  - 9.8.4 Using VI Priorities
- 9.9 Subroutines in LabVIEW
  - 9.9.1 LabVIEW Data Types
  - 9.9.2 When to Use Subroutines
  - 9.9.3 Chapter Summary
- Bibliography

## **10. OBJECT-ORIENTED PROGRAMMING IN LABVIEW**

- 10.1 What Is Object-Oriented?
  - 10.1.1 The Class
  - 10.1.2 Encapsulation
  - 10.1.3 Aggregation
  - 10.1.4 Inheritance
  - 10.1.5 Polymorphism
- 10.2 Objects and Classes
  - 10.2.1 Methods
  - 10.2.2 Properties
- 10.3 Object Analysis
- 10.4 Object Design
  - 10.4.1 Container Classes
  - 10.4.2 Abstract Classes
- 10.5 Object Programming
- 10.6 Developing Objects in LabVIEW
  - 10.6.1 Properties
  - 10.6.2 Constructors
  - 10.6.3 Destructors
  - 10.6.4 Methods
- 10.7 Example, Developing Instrument Drivers
  - 10.7.1 Complex Instrument Designs
- 10.8 Object Template
- 10.9 Exercises
- Bibliography