Introduction to LabVIEW Programming

Stephen J. Chapman
BAE Systems Australia

nts PRESS

ISBN- 978-1-934891-20-9

Publisher: Tom Robbins
Development Manager: Gretchen Edelmon
Project Manager: Catherine Peacock
Compositor: PageTech Publishing Services Pvt. Ltd.

Library of Congress Control Number: 2015941970

About NTS Press

National Technology & Science Press or NTS Press is sponsored by engineers for engineering, science and mathematics students dedicated to the publication of scholarly material of lasting value. Our products are for educators who desire a high degree of integration among classroom text, hardware and software to permit hands-on, visual learning. Our success will be judged by the influence our publications have in inspiring you and others to pursue careers in engineering, science, and mathematics.

We believe the learning process is most rewarding if you build up an intuitive understanding of concepts by pausing to tinker, explore and reflect. Thus, our publishing philosophy follows the belief that learning takes place when you are actively involved and we attempt to encourage this process in several ways. We build our textbooks with a plethora of worked-out examples, with reinforcing and advanced problems, and with interactive computer visuals. We address the computational aspects of problem solving by integrating computer-based learning tools into these presentations to enhance the discussion and analysis of engineering and science applications. In several cases our educational materials are developed with hardware experimentation platforms in mind, such as Universal Serial Bus (USB) devices for acquiring, generating and analyzing information. Owning your own portable laboratory equipment permits you to perform experimentation and take measurements anywhere and at anytime, thereby reinforcing your understanding of theoretical concepts. Having a little fun is okay too.

Clearly, rising textbook costs have an enormous effect on the way you view educational material. And there is little doubt that the Internet has been the most influential agent of change in education in recent time. The prevalence of search engine technology combined with a variety of opensource, open-content, and Wiki sites designed to deliver content has created a proliferation of freely available information and has circulated it more widely. Authored, edited, printed material is being overtaken by community libraries of digitized content from which new content is constructed, remixed, reordered and reassembled, often absent of continuity, flow, and accountability. This free content seems to address concerns about rising textbook prices, but it still has a cost. By contrast we try carefully to design products that come together in one piece with the author's judgment, passion, and imagination intact. These books are available with a reasonable price, and may be counted on as "reputable islands of knowledge in the vast ocean of unscrutinized information." www.ntspress.com

Contents

Preface xi

About the Author xiii

1 Introduction to LabVIEW 1

1.1 The Two Faces of LabVIEW Programming 2

1.2 The LabVIEW File Structure 5

1.3 Getting Started with LabVIEW 6

1.4 Exploring the Front Panel and Block Diagram 9

1.5 Pull-Down and Shortcut Menus 15

1.6 Getting Help 17

1.7 Creating, Loading, and Saving VIs 19

1.7.1 Creating New VIs 19

1.7.2 Loading Existing VIs 24

1.7.3 Saving VIs 25

1.8 Summary 25

1.8.1 Key Terms and Symbols 26

1.9 Exercises 27

2 Virtual Instruments and the LabVIEW Programming Language 34

2.1 The Components of a VI 34

2.2 Programming a VI 36

2.3 Creating a VI from Scratch 38

2.3.1 Understand the Problem 38

2.3.2 Create the VI 38

2.3.3 Add the Required Controls and Indicators to the Front Panel 38

2.3.4 The Block Diagram 49

2.3.5 Adding Nodes to a Block Diagram 52

2.3.6 Wiring the Block Diagram 54

2.3.7 Testing the Program 58

2.4 Creating Output Plots in a VI 65

2.5 Additional Examples 71

2.6 Debugging LabVIEW Programs 83

2.6.1 Debugging LabVIEW Programs that Don't Run 83

2.6.2 Debugging LabVIEW Programs that Produce the Wrong Answer 85
2.7 Summary 87
2.7.1 Key Terms and Symbols 88
2.8 Exercises 89

3 MathScript and the MathScript Node 94

3.1 The MathScript Window 95
3.2 The MathScript Language 99
3.3 Variables and Arrays 99
3.4 Assignment Statements 103
3.4.1 Shortcut Expressions 105
3.4.2 Reading Data from the Keyboard 106
3.5 Subarrays 108
3.5.1 Using Subarrays on the Left-Hand Side of an Assignment Statement 109
3.5.2 Assigning a Scalar to a Subarray 111
3.6 Special Values 111
3.7 Displaying Output Data 113
3.7.1 Setting the Default Format 113
3.7.2 The `disp` Function 113
3.7.3 Formatted Output with the `fprintf` Function 114
3.8 Data Files 115
3.8.1 Saving and Loading Data in the Default MathScript Format 115
3.8.2 Saving and Loading Data in Other Formats 116
3.9 Scalar and Array Operations 117
3.9.1 Scalar Operations 117
3.9.2 Array and Matrix Operations 117
3.9.3 Hierarchy of Operations 120
3.10 Built-In MathScript Functions 123
3.10.1 Optional Results 123
3.10.2 Using MathScript Functions with Array Inputs 124
3.10.3 Common MathScript Functions 124
3.11 Introduction to Plotting 125
3.11.1 Using Simple *xy* Plots 126
3.11.2 Printing a Plot 127
3.11.3 Exporting a Plot as a Graphical Image 127
3.11.4 Multiple Plots 127
3.11.5 Line Color, Line Style, Marker Style, and Legends 129
3.12 Examples 131
3.13 The MathScript Node 138
3.13.1 Using a MathScript Node in a LabVIEW Program Block Diagram 140
3.13.2 Error Inputs and Outputs 143
3.14 Debugging MathScript Programs 145
3.14.1 Syntax Errors 145
3.14.2 Run-Time and Logical Errors 146
3.14.3 Locating and Fixing Bugs in a MathScript Window 146

3.14.4 Using the Debugger in a MathScript Node 148
3.15 Performance Issues in MathScript 154
3.16 Using MATLAB M-files in MathScript 154
3.17 The MATLAB Node 155
3.18 Summary 155
3.18.1 Summary of Good Programming Practice 156
3.18.2 MathScript Summary 156
3.19 Exercises 159

4 Branching Structures and Loop Structures in LabVIEW Programs 165

4.1 Comparison and Boolean Functions 166
4.1.1 Comparison Functions 166
4.1.2 Boolean Functions 166
4.2 Branches in LabVIEW Programs: The Case Structure 169
4.2.1 Controlling Case Structures 173
4.2.2 Example Problem 176
4.3 Loops in LabVIEW Programs: The For and While Structures 181
4.3.1 The For Loop 181
4.3.2 The While Loop 185
4.3.3 More Details on For and While Loops 190
4.3.4 Example Problem 193
4.4 Shift Registers and Feedback Nodes 200
4.4.1 Shift Registers 200
4.4.2 Feedback Nodes 209
4.5 Useful Functions and VIs for Building LabVIEW Programs 210
4.5.1 The Programming >> Structures Subpalette 210
4.5.2 The Programming >> Numeric Subpalette 211
4.5.3 The Programming >> Boolean Subpalette 211
4.5.4 The Programming >> Comparison Subpalette 213
4.5.5 The Programming >> Timing Subpalette 214
4.5.6 The Programming >> Dialog and User Interface Subpalette 214
4.6 Loops and Vectors 215
4.7 Example Problems 222
4.8 Timing and Flat Sequence Structures 229
4.9 Summary 233
4.9.1 Key Terms and Symbols 234
4.10 Exercises 235

5 Branches and Loops in MathScript 239

5.1 Relational and Logic Operators 240
5.1.1 Relational Operators 240
5.1.2 A Caution About the == and ~= Operators 241
5.1.3 Logic Operators 242
5.1.4 Logical Functions 246

5.2 Branches 248
5.2.1 The `if` Structure 248
5.2.2 Examples Using `if` Structures 249
5.2.3 Notes Concerning the Use of `if` Structures 254
5.2.4 The `switch` Structure 256
5.3 Loops 264
5.3.1 The `while` Loop 264
5.3.2 The `for` Loop 268
5.3.3 The `break` and `continue` Statements 271
5.3.4 Nesting Loops 273
5.4 Summary 281
5.4.1 Summary of Good Programming Practice 281
5.4.2 MathScript Summary 282
5.5 Exercises 282

6 SubVIs 291

6.1 Creating a New VI and Converting It into a SubVI 293
6.2 Creating a SubVI from a Selection of Components in Another VI 303
6.3 Additional Details about Connector Panes 306
6.4 Example Problems 308
6.5 Adding User-Defined SubVIs to the Functions Palette 327
6.6 Understanding the Structure of Complicated Programs with Many SubVIs 329
6.7 Summary 331
6.7.1 Key Terms and Symbols 331
6.8 Exercises 331

7 MathScript Functions 338

7.1 Introduction to MathScript Functions 339
7.2 Optional Arguments 347
7.3 Sharing Data Using Global Memory 351
7.4 Summary 357
7.4.1 Summary of Helpful Hints 357
7.4.2 MathScript Summary 358
7.5 Exercises 358

8 LabVIEW Arrays, Clusters, Plots, and Expression/ Formula Nodes 366

8.1 Arrays 367
8.1.1 Array Controls and Indicators 368
8.1.2 Multidimensional Arrays 372
8.1.3 Creating Arrays with Loops 374
8.2 Array Functions 374
8.2.1 Array Size 377
8.2.2 Initialize Array 379
8.2.3 Build Array 381
8.2.4 Array Subset 384
8.2.5 Index Array 386
8.2.6 Array Max & Min 386
8.3 Polymorphism 388
8.4 Clusters 389
8.4.1 Creating Cluster Controls and Indicators 390
8.4.2 Cluster Order 393
8.4.3 The Bundle and Unbundle Functions 394

8.4.4 The Bundle by Name and Unbundle by Name Functions 399
8.4.5 Cluster Constants 403
8.4.6 Polymorphism and Clusters 403
8.5 Charts and Graphs 403
8.5.1 Waveform Charts 404
8.5.2 Plotting Multiple Waveforms on a Waveform Chart 409
8.5.3 Waveform Graphs 412
8.5.4 Plotting Multiple Waveforms on a Waveform Graph 416
8.5.5 XY Graphs 418
8.5.6 Plotting Multiple Waveforms on an XY Graph 422
8.5.7 The Waveform Data Type 422
8.5.8 Customizing Lines on Plots 426
8.5.9 Customizing Axes on Plots 429
8.5.10 Plot Legends 431
8.5.11 Saving Images of Charts, Graphs, and other Indicators 434
8.6 Simplifying Calculations on a Block Diagram: Expression Nodes and Formula Nodes 445
8.7 Summary 450
8.7.1 Key Terms and Symbols 450
8.8 Exercises 452

9 LabVIEW Strings and File I/O 457

9.1 Strings 458
9.1.1 String Controls and Indicators 458
9.1.2 String Functions 462
9.2 LabVIEW Error Handling 471
9.2.1 Passing Error Clusters to and from SubVIs 482
9.3 File Input/Output 482
9.3.1 Reference Numbers and Standard Errors 484
9.3.2 Opening and Closing Files 485
9.3.3 Writing String Data to and Reading String Data from a Text File 487
9.3.4 Writing and Reading Other Sorts of Data to and from a Text File 494
9.3.5 Reading an Unknown Amount of Data from a Text File 501
9.3.6 Specifying File Names in I/O Operations 504
9.3.7 Reading and Writing Binary Files 505
9.4 LabVIEW Measurement Files 516
9.4.1 Writing Data to a Measurement File 516
9.4.2 Reading Data from a Measurement File 522
9.4.3 Binary Measurement Files 526
9.5 Summary 526
9.5.1 Key Terms and Symbols 527
9.6 Exercises 527

10 MathScript Strings and File I/O 529

10.1 String Functions 530
10.1.1 String Conversion Functions 530
10.1.2 Creating Two-Dimensional Character Arrays 530
10.1.3 Concatenating Strings 531
10.1.4 Comparing Strings 532
10.1.5 Searching and Replacing Characters within a String 534
10.1.6 Upper-Case and Lower-Case Conversion 536
10.1.7 Trimming Whitespace from Strings 536
10.1.8 Numeric-to-String Conversions 537
10.1.9 String-to-Numeric Conversions 538
10.1.10 Summary 539
10.2 The `textread` Function 545
10.3 MathScript File Processing 546
10.4 File Opening and Closing 548
10.4.1 The `fopen` Function 548
10.4.2 The `fclose` Function 550
10.5 Binary I/O Functions 550
10.5.1 The `fwrite` Function 550
10.5.2 The `fread` Function 551
10.6 Formatted I/O Functions 554
10.6.1 The `fprintf` Function 555
10.6.2 Understanding Format Conversion Specifiers 555
10.6.3 The `fscanf` Function 558
10.6.4 The `fgetl` Function 560
10.6.5 The `fgets` Function 560
10.7 Comparing Formatted and Binary I/O Functions 560
10.8 File Positioning and Status Functions 566
10.8.1 The `exist` Function 566
10.8.2 The `feof` Function 568
10.8.3 The `frewind` Function 568
10.8.4 The `fseek` Function 569
10.9 Summary 569
10.9.1 Summary of Helpful Hints 570
10.9.2 MathScript Summary 570
10.10 Exercises 571

11 Selected Additional Topics 573

11.1 VI Names and Name Conflicts 573
11.2 File Names and Moving VIs 581
11.3 LabVIEW Projects 583
11.4 Stand-Alone Applications 583
11.5 Where Do We Go from Here? 583

Index 585

Preface

LabVIEW is a development environment that engineers and scientists use to write programs that capture and process technical data. LabVIEW stands for **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench. The program was developed by National Instruments, a company that specializes in developing software to allow engineers to capture, analyze, process, and store data from laboratory instruments.

The great strength of LabVIEW is that it can acquire data in real time from almost any type of instrument and save the data to disk. It can also process the data in many ways, either as it is received or afterwards from disk files.

LabVIEW was designed around the concept of a Virtual Instrument (VI), the software equivalent of one of the physical pieces of equipment sitting on the laboratory workbench. The Virtual Instrument accepts data from input controls or terminals, processes the data, and sends the resulting data to displays or output terminals. This processing is normally done by connecting a VI and processing nodes together with virtual wires, in a manner that is known as the LabVIEW graphical programming language.

In addition, LabVIEW supports an optional conventional procedural programming language called MathScript. The MathScript language is virtually identical to MATLAB, which is *very* widely used in engineering applications. MathScript functions and programs can be run in a stand-alone MathScript Window, or they can be run in a MathScript Node that can be integrated into LabVIEW graphical programs in a VI.

The LabVIEW graphical programming language is built-in, whereas MathScript is an extra-cost option. Nevertheless, MathScript can be attractive for the following reasons:

1. Some sorts of calculations are easier to express as equations, compared with wiring many low-level nodes together.
2. There are many thousands of *existing* MATLAB functions shared freely on the Internet. With MathScript, these functions can be integrated into a user's LabVIEW programs without the need to rewrite each one.

This text teaches the basics of both the LabVIEW graphical programming language and the MathScript procedural programming language. It starts from the very beginning and teaches the student how to solve problems in each language. We define a problem-solving methodology for each language and then apply it consistently to solve problems. The book works from the basics, building up skills in both languages. In many cases, we solve the *same* problem in both languages, allowing a student to compare the techniques applied in each language.

Note that the book has been designed so that the LabVIEW graphical programming and the MathScript procedural programming can be decoupled from each other. It can be used for courses with different emphases:

- A course on LabVIEW graphical programming would feature only Chapters 1, 2, 4, 6, 8, 9, and 11.
- A course emphasizing MathScript would feature only Chapters 1, 3, 5, 7, and 10.
- A course covering both topics would use all chapters.

There is a certain amount of repetition in the text to allow this decoupling. For example, the discussion of the benefits of LabVIEW subVIs at the beginning of Chapter 6 is very similar to the discussion of the benefits of MathScript functions at the beginning of Chapter 7. The concepts are the same for both, but the material is repeated in each chapter because some courses may skip one or the other.

The book largely alternates between chapters featuring LabVIEW graphical programming and chapters featuring MathScript, with each pair of chapters teaching the same basic concepts in the two different languages.

A Final Note to the User

No matter how hard I try to proofread a document like this book, it is inevitable that some typographical errors will slip through and appear in print. If you should spot any such errors, please drop me a note via the publisher, and I will do my best to get them eliminated from subsequent printings and editions. Thank you very much for your help in this matter.

I will maintain a complete list of errata and corrections at the book's website, which is http://www.ntspress.com/publications/introductiontolabviewprogramming/ Please check that site for any updates and/or corrections.

Stephen J. Chapman
Melbourne, Australia

About the Author

Stephen J. Chapman received a BS in electrical engineering from Louisiana State University (1975) and an MSE in electrical engineering from the University of Central Florida (1979). He then pursued further graduate studies at Rice University.

From 1975 to 1980 he served as an officer in the U.S. Navy, assigned to teach electrical engineering at the U.S. Naval Nuclear Power School in Orlando, Florida. From 1980 to 1982, he was affiliated with the University of Houston, where he ran the power systems program in the College of Technology.

From 1982 to 1988 and from 1991 to 1995, he served as a member of the technical staff of the Massachusetts Institute of Technology's Lincoln Laboratory, both at the main facility in Lexington, Massachusetts and at the field site on Kwajalein Atoll in the Republic of the Marshall Islands. While there, he did research in radar signal processing systems. He ultimately became the leader of four large operational range instrumentation radars at the Kwajalein field site (TRADEX, ALTAIR, ALCOR, and MMW).

From 1988 to 1991, Mr. Chapman was a research engineer at the Shell Development Company in Houston, Texas, where he did seismic signal processing research. He was also affiliated with the University of Houston, where he continued to teach on a part-time basis.

Mr. Chapman is currently Manager of Systems Modeling and Operational Analysis for BAE Systems Australia in Melbourne. He is the technical leader of a team that has developed a model of how naval ships defend themselves against anti-ship missile attacks.

Mr. Chapman is a Senior Member of the Institute of Electrical and Electronic Engineers (and several of its component societies). He is also a member of Engineers Australia.