

Preface

The personal computer now stands alongside such equipment as the multimeter and the oscilloscope as one of the standard tools of scientific research. This powerful instrument is included in most modern-day laboratories because it facilitates sophisticated measurements beyond the reach of manually operated set-ups and is invaluable in analyzing the resulting data. Thus, most instructors involved in the design of upper-division science and engineering curricula feel strongly that a portion of each student's advanced laboratory experience should be devoted to instruction in computer-based experimental skills.

Quite commonly, the topic of computer-based experimentation is presented in college courses through a sequence of experiments that explore the low-level details of the computer interface. Students investigate the mechanisms by which analog-to-digital and digital-to-analog operations are accomplished at the chip level, then go on to study the machine-level design of microprocessors. Finally, students learn to program the microprocessor chip using assembly language instructions and write a routine to carry out a primitive laboratory task. With this firm grounding in basic computer operation at the machine level, it is hoped that the student, when actually carrying out a computer-based laboratory experiment at a future date, will have the basis to understand pertinent issues and to troubleshoot problems.

However, in response to the fast-paced advances that have occurred in scientific instrumentation over the past 15 years, many educators have begun to revise their presentation of computer-based research skills. With the low cost, sophistication, and reliability of commercially available products, most active researchers no longer construct homemade data-taking hardware or write assembly language code. Rather, they wisely purchase robust data acquisition systems that can be controlled with symbolic ("high-level") commands. In contrast to the intimate knowledge of chip-level processes required in the past, today's researcher is routinely called upon to implement a wide range of high-level computer skills—for example, using a symbolic programming language to control a data acquisition process, storing the resultant data in a convenient form within a computer's filing system, communicating with instruments through the General Purpose Interface Bus (GPIB), or writing a program to analyze data in real time using sophisticated techniques such as the Fast Fourier Transform. This insight opens the door to a new realm of possibilities in the design of an advanced instructional laboratory. Rather than devoting the entire course to the time consuming task of teaching the details of data-taking technology, an instructor is freed to contemplate a much broader range of instruction in which students learn to use the computer for instrument control, data gathering, analysis, and interpretation. With the proper mix of

offerings, such a course has the potential to outfit students with a repository of relevant ideas, tools, and experiences that they can directly call upon in their first real research projects, and in the years beyond.

Inspired by such musings, while teaching at Occidental College eight years ago, I began to include a high-level presentation of computer-based experimentation in my upper-division laboratory course. Using National Instruments GPIB boards plugged into the expansion slots of several personal computers and the related language-specific driver software, I sought to teach students how to computer control the GPIB-interfaced instrumentation they were using to perform solid-state physics experiments. While this pedagogical experiment worked very well for some students, I judged it a failure for others, and it was not difficult to identify the source of this varying success. Those students with a solid background in computer programming could “hit the ground running” and almost immediately start to explore my intended topic—computer-based experimentation. Of course, these computer-literate students “spoke” a variety of programming languages, but I solved this minor inconvenience by developing a library of BASIC, FORTRAN, and C compilers and driver software for the course. The real problem was students with little or no computer programming background. These students were confronted with the hurdle of learning at least the rudiments of a particular computer language prior to tackling the real task at hand—computer interfacing. While most of the computer neophytes were fascinated by the prospect of controlling an instrument remotely, it proved impossible for them, given the 10-week time period of my course, to become competent in a language such as C at the level required to code a data acquisition program. I realized that I had to remove this software stumbling block so that each of my future students could be successful, but I never satisfactorily solved this teaching problem while at Occidental.

I first saw a LabVIEW system when I arrived at Reed College five years ago. After playing with this graphical programming language for a few hours, I became convinced that it was the perfect environment in which to teach computer-based research skills. Its intuitive use of wired-together icons to perform simple-to-understand tasks made adding two numbers, building an array, parsing a string, and digitizing an analog voltage practically equivalent in programming ease. Moreover, with a minimum of difficulty, one could implement all sorts of sophisticated analysis techniques such as curve fitting, digital filtering and fast Fourier transforming. On top of all that, LabVIEW programming was fun! It seemed to me that even a complete computer novice, given the proper systematic instruction, could learn this language in a short amount of time and then use it to perform a wide variety of laboratory tasks using the computer.

Five years later, I can say that LabVIEW has far exceeded my initial optimistic expectation as an effective pedagogical tool. Its programming features are clear, coherent, powerful, comprehensive and entertaining, enabling an instructional presentation of computer-based experimentation in which students create meaningful programs that illustrate useful concepts at each step of the learning curve. LabVIEW programs are modular, so that after each is created and understood, it becomes part of a library that can be used later as a building block of a more sophisticated program. With just a few weeks of training, students develop the ability to explore their own ideas by modifying programs or by writing appropriate new ones and retain these skills long after the course is over.

Advanced LabVIEW Labs contains the collection of LabVIEW-based exercises that I have developed and used successfully to present computer-based experimentation in my junior-level Advanced Laboratory course at Reed. The first seven chapters are designed to teach the LabVIEW programming language including its four control structures (While Loop, For Loop, Case Structure, and Sequence Structure), three graphing modes (Waveform Chart, Waveform Graph, and XY Graph) and File I/O. Chapters 8 and 9 explore two interesting and commonly used analysis techniques—least-squares curve fitting, and the Fast Fourier Transform. Chapters 10 and 11 investigate the analog-to-digital and digital-to-analog conversion functions available on a National Instruments data acquisition board. Finally, Chapter 12 provides instruction in LabVIEW-based GPIB control of laboratory instrumentation. Based on my experience in using these materials for instruction at Reed, each chapter takes approximately two to three hours for a student to complete.

Advanced LabVIEW Labs is very much a hands-on presentation of its subject. In a sense, the book is one long exercise through which students work their way. Because LabVIEW is a graphical (as opposed to text-based) programming language, its concepts and programs can be naturally rendered in pictures. Most pages of the book contain a screen shot (or two) of the LabVIEW program students are asked to reproduce. In earlier chapters, as the programming language is being taught, detailed instructions and multiple screen shots of intermediate programming steps are given to assist the student in successfully completing the exercises, while in later chapters, students are expected to complete the programs with much less coaching.

A modular approach to problem solving and programming is emphasized throughout the text. LabVIEW programs written in earlier chapters are used as sub-programs within the more advanced programs of later chapters. For example, the Sine Wave program written in Chapter 3 is used in the Fast Fourier Transform program developed in Chapter 9, which is then used as a sub-program within the Spectrum Analyzer built in Chapter 10.

Similarly, the construction of a temperature control system provides a unifying thread through several of the chapters. The calibration curve for a temperature-sensing thermistor is determined in Chapter 8, the temperature measurement set-up is assembled in Chapter 10, and the complete temperature control system is constructed as a somewhat open-ended project in Chapter 11. Attention-grabbing and useful electronic components such as a thermistor and thermoelectric device are used in this project to enhance student interest. Appendix I provides detailed construction plans for the low-cost (approximately \$80) temperature control apparatus needed.

All of the programs presented in this book are written using LabVIEW Version 5. The text will also work perfectly well for Version 4 users. However, LabVIEW 3 owners are gently advised that it might be time to consider an upgrade. While Version 3 of LabVIEW possesses all the capabilities necessary for this book, the editing steps are different (and primitive) compared to the later versions. But if you don't mind the extra transcription work, all of the *Advanced LabVIEW Labs* exercises can be executed in LabVIEW 3.

To aid readers in completing the exercises, I use the following conventions throughout the book: **Bold** text designates the features such as graphical icons, palettes, pull-down menus, and menu selections that are to be manipulated in the course of

constructing a LabVIEW program. The descriptive names that label controls, indicators, custom-made icons, programs, disk files, and directories (or folders) are given in the **straight** font. *Italic* text highlights character strings that the programmer must enter using the keyboard and also signals the first-time use of important terms and concepts.

Advanced LabVIEW Labs is compatible with both the Full Development System and the Student Edition of LabVIEW. To minimize the amount of valuable laboratory time devoted to learning the LabVIEW programming language, an instructor might consider having students purchase personal copies of the low-cost Student Edition software and then perform the software-only Chapters 1 through 9 as homework on their own computers.

The following table lists the items required to work the exercises in this book.

Chapter	Additional Software	Plug-In Board	Additional Hardware
1			
2			
3			
4	Word Processor ^a and/or Spreadsheet Analysis ^b Program		
5			
6			
7			
8			
9			
10		DAQ ^c	Thermistor ^e , Constant Current Circuit ^e , Function Generator
11		DAQ ^c	Thermistor + Constant Current Circuit, Thermoelectric Device + Current Driver Circuit ^f
12		GPIB ^d	GPIB Interfaced Instrument ^g

^aSuch as Microsoft Word

^bSuch as Microsoft Excel

^cSuch as E Series MIO, PCI-1200, Lab-PC or Lab-NB Board

^dSuch as AT-GPIB, PCI-GPIB, NB-GPIB

^eSee Chapter 10

^fSee Appendix I

^gSuch as HP34401A Multimeter

Some familiarity with op-amps is necessary to understand the circuits in Chapter 10 and 11. In my year-long Advanced Laboratory course, students acquire this requisite background from seven weeks of advanced electronics experiments (amplifying, filtering, timing and logic circuits) that precedes their instruction in computer-based data acquisition and analysis. The material in *Advanced LabVIEW Labs* is then covered over the course of seven weeks in the following manner: Chapters 1–3 (Week 1),

Chapters 4–7 (Week 2), Chapters 8–9 (Week 3), Chapter 10 (Week 4), Chapter 11 (Week 5), Chapter 12 (Week 6 and 7). A one-hour lecture each Monday provides students with an overview of that coming week's lab work. After completing the LabVIEW-based instruction, my Advanced Laboratory students then devote the remaining 11 weeks of the course to open-ended investigations of physical phenomena.

One indication of the effectiveness of *Advanced LabVIEW Labs* is that many Reed students have successfully utilized their LabVIEW programming expertise to execute sophisticated open-ended Advanced Laboratory experiments during the latter portion of the course. Many also have gone on to use LabVIEW in their Senior Thesis projects. LabVIEW programming, because of its wide use in industrial and research labs, has also proved to be a marketable skill to students applying for summer internships, full-time jobs, and graduate schools. I hope you, too, will find *Advanced LabVIEW Labs* to be an interesting and beneficial introduction to the power and flexibility of LabVIEW-based data acquisition and analysis.

Any suggestions or corrections are gladly welcomed and can be sent to John Essick, Reed College, 3203 SE Woodstock Blvd., Portland, OR 97202 USA or jessick@reed.edu.

For their advice and assistance while I prepared this book, I would like to thank my colleagues David Griffiths, Richard Crandall, Mary James, and Mark Beck, my students Zach Nobel and Ben Palmer as well as Ravi Marawar, Mahesh Chugani and Lisa Wells of National Instruments. Also I'd like to express my appreciation to Roger Bengston (University of Texas, Austin), Grant Hart (Brigham Young University), Robert B. Muir (University of North Carolina, Chapel Hill), Edward Nadgorny (Michigan Technological University), and Ronald Ransome (Rutgers University) for their helpful reviews of the manuscript. Special thanks to Alison Reeves and Gillian Kieff of Prentice Hall, and Jennifer Maughan and her crew at WestWords, for making this book possible. I gratefully acknowledge the support of the M.J. Murdock Charitable Trust and the National Science Foundation in purchasing LabVIEW systems for the Reed College Physics Department. Finally, to my family: Thank you for being lively and loving while I worked on this project.

John Essick
Portland, Oregon