# Revision and Translation of Existing Programs as a Tool for Teaching Computer Data Acquisition and Control Systems Design and Implementation

**Thomas Hannigan, Keith Koenig, Bryan Gassaway, Viva Austin**
**Department of Aerospace Engineering, Mississippi State University**

## Abstract

Keeping data acquisition and control systems (DACS) used in a graduate and under-graduate laboratory current in a rapidly evolving technological environment is an expensive and time-consuming task. Computer architecture and software have evolved more rapidly than the curriculum repeats, and the interfaces commonly used for DACS now vary widely, including parallel, serial, and Ethernet based protocols. Experimental programming is thus under near-constant revision and adaptation. Since the aerospace industry is widely varied, entry-level engineers may end up working with legacy systems from long-established laboratories, or find themselves in a startup research lab associated with modern computational facilities. It is essential that students learn the basics of designing experimental DACS, as well as the adaptation and evolution of existing programs. Using the well-documented and complete programs of the past allows a complete illustration and understanding of the principles of DACS, and provides a familiarization with legacy programming limitations. The revision of DACS programs written in various forms of BASIC and Testpoint into a more commonly used environment such as LabVIEW insures that the undergraduate laboratory experience interests, prepares and enthuses the experimentalists of tomorrow. This paper discusses and documents the processes used to familiarize upper division aerospace engineering students with the black arts of DACS. Details concerning the programming tasks, legacy hardware and software issues, and the motivation for keeping laboratory studies current are discussed. Also detailed are measures of student success and outcomes assessment concerning laboratory studies.

## Motivation for Continuing Laboratory Education

Every engineering discipline has struggled to keep classrooms and laboratories abreast of the waves of technology sweeping them into the future. In aerospace engineering in particular, the rapidly evolving computer hardware and software have enabled great strides in computational field simulations. This evolution has benefited every major discipline and thrust area of this field, including analysis, simulation or optimization of structures, aerodynamics, propulsion, and control systems. The tools used in the educational laboratory have had to evolve to keep pace with this technological revolution, and in an economic climate of declining tax revenues, public-funded institutions in particular have struggled to remain abreast. Laboratory managers and educators have

been in a constant revisionist mode just to keep up with the steady flow of ever faster and more capable computers and related data acquisition and control systems.
A quick look at the revenues invested in such hardware from one of the prominent suppliers, National Instruments[1], revealed a tremendous growth in the use of new technology, with NI net corporate incomes increasing by an order of magnitude in the late 80's, and a similar increase through the 90's, to a level four times greater than that of Keithley,[2] one of the most prominent suppliers of traditional equipment for decades, while Keithley also experienced moderate growth. Especially in the last few years, clones of the data acquisition boards of both these companies are also in plentiful supply. As computer systems evolved, hardware peripherals such as data acquisition, signal conditioning, and controller modules evolved likewise. A host of different hardware buss architectures and port communication protocols came into being, with some of them vanishing entirely within a generation. Although the cost of individual computers continued to decline during the last decade, the requirement for recurrent upgrades or replacements to software and hardware accelerated, with a great increase in the cost of this new technology. Since the introduction of new technology into industry was proceeding at the same accelerated pace, it was essential to insure that the students studying to be the fuel for this ongoing overhaul remain abreast of the current technologies, yet also be cognizant of the capabilities of the old. Many small companies cropped up to provide equipment and programming for data acquisition and control, but those engineers working with larger government and industrial laboratory facilities have generally been expected to adapt and extend their own facilities into a new age.

As a result of this continued path of evolution, aerospace engineering laboratories and classrooms have had to insure that the general computer and programming skills that were being taught were also under near-constant revision and adaptation. The use of computer data acquisition and control systems depended on programming in languages such as Pascal and various versions of BASIC, and those were evolving very rapidly. Suppliers of data acquisition cards for PCs offered sample programs and drivers first for the most common versions of Pascal, and interpreted BASIC, and pre-compiled binary drivers to be loaded into memory for use by more simple control programs. Borland's Turbo-Basic was adapted to common use for making the compilation process simple. At the same time graphical and object-oriented programming environments were being developed. These were soon emphasized as the way of the future in a windowed environment, and soon made an individually programmed solution a thing of the not-so-distant past.

Since the aerospace industry is widely varied, entry-level engineers may end up working with legacy systems from long-established laboratories, or find themselves in a startup research lab associated with modern computational facilities. It is highly unusual for even a well-established laboratory to have a static programming environment. Experimental research facilities such as wind tunnels, constructed decades ago, are still operable today, though little similarities exist between the hardware packed racks of yesteryear and the compact computer measurement and control equipment that are likely to be installed to control those facilities today. In some instances, however, those old control systems are just now being replaced, often by entry-level engineers who come to

the workplace with some understanding of and experience in the new programming environments such as Testpoint and LabVIEW. On the other hand, there are new and smaller facilities for specialized research that are being put in place by individual engineers and smaller companies, that can ill afford to duplicate the research equipment used before. These new companies often rely on the relatively new-skilled recent graduates who are still accustomed to learning hardware and software, and provided that their education was up-to-date with current technology, are likely to be familiar with state-of-the-art computers and data acquisition and control hardware.

As examples of these trends several recent graduates in aerospace engineering at Mississippi State University have secured jobs working with long-established companies precisely because of their knowledge of DACS programming. These included various groups from Boeing, Lockheed Martin, and contractors to NASA, where students were hired because of their exposure to ASYST, Testpoint, or LabVIEW. Furthermore, continuing surveys of graduates and employers have indicated their educational experiences with DACS programming were both necessary, and appropriate. Also, in recent class-related visits to such facilities as the propulsion labs at NASA Marshall, students have seen first hand how practicing engineers use the same sort of equipment and LabVIEW programming in their work as they use in their classes. Reinforcing this, some of the engineers specifically discussed how their student interns and new hires were most useful in updating the programs used for these experiments.

It is essential that students learn the basics of designing experimental DACS, as well as the adaptation and evolution of existing programs. While not every student will eventually work in a laboratory setting, it is likely that the results of their computational or design work will end up being tested in such a facility. Their understanding of the processes and limitations of experimental endeavors is essential if there is to be a successful feedback from the lab to the designer and manufacturer to complete the design process. If every student participates in the process of experiment design, programming for data acquisition and control, and conduct of laboratory tests, they will at least gain the necessary appreciation and knowledge of how that process relates to their computational analyses of the topics at hand. Since not every experiment is developed from scratch, and multiple and varied software solutions often exist for laboratory DACS tasks, a familiarization with those generations of solutions can be effective in giving the student a better perspective on the benefits of the latest software solutions. At the same time, using earlier programs as models for the development of the next generation solutions prepares the students to do precisely the same thing if they do end up working in an experimental laboratory.

Using the well-documented and complete programs of the past allows a detailed illustration and explanation of the principles of DACS, and provides a familiarization with legacy programming limitations. A key to effectively providing this education relies upon presenting appropriate coded solutions, and proper advice and counsel that allows a student to make modifications to existing programs, or to realize when it is more cost or time effective to build a new and perhaps more robust programmed solution. In some cases there are hardware issues that mandate a complete retrofit of data acquisition

equipment due to obsolescence of devices, or an upgrade in the speed or change in interface to devices that require a complete overhaul of the programming.

Past students have had a curricular requirement for a Fortran class, and most after the late 1980s came with rudimentary experience in BASIC programming from high school. The requirement for a separate class in computer programming has been eliminated from the current curriculum of aerospace engineering at Mississippi State University, but programmed solutions are still presented in many of the courses of this discipline. Since many students did not take an additional programming language course, instructors in those courses have commonly had to take the time to introduce syntax and structure of programming required for their classes. The issue of such overhead that took instructional time from their primary topics has been addressed in this and many other institutions by the addition of introductory courses[3] that include such topics as introductions to MathCAD[4] and Matlab[5], and specific familiarization with programs used in the laboratory for data acquisition and control. In the past fifteen years, the languages and derivatives shown in Table 1 have been used for programming in aerospace engineering laboratory classes.

| | | |
|---|---|---|
| Pascal | Basic (PC DOS) | CPM BASIC |
| Turbo Basic | Power Basic | BASICA |
| Mbasic | GWBasic | HPBasic |
| QuickBasic | VisualBasic | C/Perl |

Table 1: Languages used for data acquisition, control and analysis 1988-2003

Hewlett Packard (HP) BASIC was used with specific HP wind tunnel DACS equipment, and evolved through several upgrades of software and hardware. The general purpose experiments were revised as BASIC included in operating systems and their derivatives evolved from interpreted to compiled versions, and equipment drivers and programming examples were routinely provided for use in those environments. In addition, programming was accomplished in the environments of ASYST, Labtech Notebook, Testpoint, and LabVIEW[6]. The revision of DACS programs written in various languages into current commonly used environment such as LabVIEW insures that their experience in the undergraduate laboratory interests, prepares and enthuses the experimentalists of tomorrow.

**The undergraduate laboratory DACS experience**

The motivation for conducting the programming in a multi-faceted format has been established, and the focus will now be turned toward the specific implementation methods used in the undergraduate aerospace engineering laboratory at Mississippi State University. DACS programming for the laboratory tests conducted by undergraduates is often just a "black box" experience, with the focus on the results of the tests. Students can gain an added benefit from their experience, if the black arts used in such programming are illustrated and explained at every opportunity. Thus, in lower division undergraduate classes the languages and environments are presented, and in upper

division classes, a more complete understanding can be affected through developing DACS algorithms and flow charts from the study of existing programs. Required modifications are made to the existing programs, or those algorithms are implemented in a newer graphical environment, currently LabVIEW. Hence, little of the past is simply discarded unless it is merely redundant. For example, many program solutions exist in various BASIC versions, so TurboBasic is used here for illustration of typical legacy program solutions. Since many of the DACS programs used in introductory classes are compiled versions, the code itself is not examined, but rather flow charts or other explanations of the solution algorithms are presented.

If a single DACS programming environment, such as LabVIEW, is chosen, and all of the programming required for data acquisition and control of peripherals is presented only in that environment, deficiencies in student preparation may occur. Teaching in a single environment from scratch, assuming no previous programming experience and introducing no previous solutions might arguably allow a more complete familiarization with that particular environment. This would however, limit the number of topics introduced in the lab, and the programming itself may then become the focus of the laboratory experience, rather than the use of that programming as a tool. This does not prepare the students for their future in adapting and expanding existing solutions, and can lead to confusion and inactivity when new languages are introduced. The students focus on the difference in syntax instead of the problem at hand. This deficiency has been addressed by emphasizing good solution development including a five step problem solving method.[7] Students are asked to adhere to a rigorous application of these five steps: stating the problem clearly, describing all input and output, working an example problem by hand if appropriate, developing an algorithm or flow chart, and finally coding into a computer solution for testing. By using examples of acceptable solutions, and providing building block solutions, the students are exposed to legacy programming, and black box programming is de-emphasized. The intentions are to build student confidence in algorithm development, and provide a broader experience in the programming typically used for DACS.

Typical programming tasks and the methods used to accomplish these tasks are listed in Table 2 below. Students are generally free to choose their desired programming method on a particular task, but are required to use all methods over a series of similar analyses. Generally on a given task individual choices vary such that all methods are used on practically every problem. The similarities, advantages and disadvantages can then be detailed during common lecture periods. In the lab exercises associated with the introductory classes, the solutions are often provided, with little modification required other than variable manipulation to explore the effects of a given variable. In the upper division laboratory sequence, the students in various sections have a common lecture period, where the contrast of several methods can be detailed. PERL and C programming have only been used when a number of students indicated a familiarity with those languages as their primary means of programming. Similarly, Matlab is a secondary choice not commonly used for most analysis tasks. Testpoint was used as the primary graphical programming environment prior to 2000, when the college of engineering began coordinating a site license for LabVIEW. Data acquisition hardware pre-1999 was

primarily purchased from Keithley-Metrabyte, but after National Instrument Hardware and LabVIEW became more commonly used in the US, a transition was made to LabVIEW. LabVIEW programming is now required for individual student projects and is used in the development of new laboratory experiments. Transitions to newer versions of software are coordinated for the semester following the semester in which new versions are released.

The elimination of the programming language requirement in the aerospace engineering curriculum at Mississippi State University has left students with a general lack of programming familiarization. This is gradually being rectified, however, by the use of MathCAD, Matlab, Maple and Mathematica (the "M-codes") in three introductory aerospace engineering courses and in some mathematics courses. There are many instances where the analysis of an experiment includes a comparison to a theoretical prediction, with these predictions being generated via closed-form, iterative, or higher order numerical solutions. These solutions are most often compared in the form of plots of experimental data overlays of theoretical solutions. The elimination of a programming language from the curriculum does not negate the requirements for systematic solutions. Though solutions to some problems can be affected using spreadsheet programming, it is generally found that for other than simple data regression and analysis, programmed solutions are still the norm. Most students now choose to use one of the common M-codes, and those codes are being extended to include DACS compatibility with LabVIEW and direct access to National Instruments compatible hardware.

The following is a description of common laboratory experiments accomplished by all aerospace engineering students during their first laboratory course.

**An Introduction to Data Analysis** A set of calibration data is read into memory from a sequential data file, then output to a formatted file. In EXCEL, the data set is plotted, a linear regression is performed, statistical data is examined, and then a report is written. Programming for data reduction and analysis is performed with BASIC, Fortran, and MathCAD. All students initially use their choice of one of the three programming methods, then all three methods are reviewed in detail.

**An Introduction to Data Manipulation** A more lengthy and complex data set taken from a wind tunnel experiment with a pressure wing is manipulated to provide data in a format for analysis for pressure coefficient calculations, and force coefficient determinations. Programming languages are used to manipulate the data into proper format for use with EXCEL.

**An Introduction to a Laterally Vibrating Cantilevered Beam** Students examine analytical methods for determining vibration modes and nodal positions of a structure using Finite Element Analysis with Unigraphics, and a computational solution implementing Mykelstad's Method with BASIC is examined. An experimental evaluation is conducted using a shaker apparatus.

**An Introduction to a Vibrating Propeller** More complex vibrations are examined as torsional, bending, and mixed mode vibrations are examined in detail. Although this analysis is primarily experimental, the computational methods commonly used in classical analysis are explained and demonstrated. Programs written in BASIC and Fortran are presented.

**An Introduction to LabVIEW** Programming for data acquisition with LabVIEW is demonstrated, and students conduct an analysis of data acquisition results to determine minimum sampling frequencies and sampling durations required for accurate determination of frequency and magnitude apparent in transducer signals.

**A Study of Mechanics of Materials with a Strain Gage Mounted on an Aluminum Beam** This study includes the use of a LabVIEW program to collect experimental data relating load, deflection, and strain at points on a beam. This assignment begins a series in which DACS programming is only incidental to the primary task at hand.

**Calibrations and Measurements with Commonly Used Transducers** Strain gage, potentiometric and solid state transducers are examined, and DACS programs are written or revisions are made to existing programs to accomplish calibration and use of these transducers in typical fashion.

**An Introduction to Peripheral Control** The control of computer peripherals is illustrated, and control programs are written in BASIC and Labview for a fundamental project involving digital input and output.

Although a total of twelve such projects are completed in one course and eight in the second laboratory course, several additional experiments are conducted with other core curriculum classes to insure an adequate experimental exposure. The following table lists tasks common to some of the labs, and the tools used to accomplish those tasks now and in the past. Demonstrated codes or solutions provided to the students typically require only minor modifications for their particular solutions, or in some cases merely using the appropriate compile/editing programs to enter and then run the particular code. Where more than one solution method is indicated, a group of students would be broken down into individuals or pairs to accomplish solutions with a particular method, then the students would exchange and explain all solutions for a given problem. Thus peer-to-peer learning allows a much broader grasp of the nature of open-ended, multiple-solution-path problems.

| Laboratory Task | Demonstrated Methods | Student Solution Methods |
|---|---|---|
| Plotting calibration data, linear regression, statistical analysis | BASIC, C, Fortran programs, Excel Spreadsheet | Excel, MathCAD, BASIC, C, Fortran programs, Matlab program |
| Plotting experimental airfoil pressure data, determination of sectional airfoil properties | LabVIEW DACS, BASIC, EXCEL, Matlab data reduction and analysis | Excel, MathCAD, Matlab, BASIC, C programs |

| | | |
|---|---|---|
| Analysis of primary wind tunnel data for stability derivatives | BASIC, C, MathCAD | BASIC, Fortran, C, Matlab, MathCAD |
| Data acquisition of pressure distributions in low speed pipe flow and in a super-sonic converging-diverging nozzle, theoretical analysis | BASIC, LabVIEW for data acquisition, analysis with BASIC, LabVIEW, MathCAD | BASIC, Excel, MathCAD analysis |
| Data acquisition, control of a portable wind tunnel, display of airfoil pressure distribution | BASIC, Testpoint, LabVIEW | Testpoint, LabVIEW |
| Data acquisition and control of arbitrary peripheral | Assembly language, BASIC, C, LabVIEW | Assembly language, BASIC, C, LabVIEW |
| Data acquisition of arbitrary waveform, plotting, frequency identification | BASIC, Testpoint, LabVIEW | BASIC, Testpoint, LabVIEW |

Table 2: Methods demonstrated and used for typical laboratory task completion

DACS codes written in BASIC or other programming languages are generally patterned after example programs provided by the manufacturer. These codes typically read 1-7 channels of analog data or they have up to four digital inputs or outputs, from a data acquisition card through the use of a series of imbedded codes provided by the manufacturers of the devices. The actual calls to the device are normally made through an assembled machine code that is device specific. Teaching students how the basic inputs and outputs are accomplished is generally done by having them write a machine code using the DEBUG editor native to all PC operating systems. Without an understanding of the nature of machine communications, particularly with peripheral devices, the students have a difficult time grasping the concept of using imbedded code in a program. However, with the current growth in graphical programming environments where the programmer is buffered from the actual coding process through the use of a graphical user interface, the concept of using pre-defined libraries of functions and tools is becoming easy for students to grasp.

**Outline of Elements of a Typical BASIC DACS Program**

Initialization Subroutines (loaded with $INCLUDE statements)
- Examples provided by the manufacturer, commented source code provided
- Typically consists of parameter statements, variable assignments and absolute calls to procedures within the array space containing the binary driver

Binary Driver File
- Loaded into memory at run-time, contains remote procedure calls and interfaces hardware (loaded with BLOAD statements)

DACS Program Logic
- Arranged to parse data returned by subroutine calls
- Includes detailed code to perform plots/regression analysis
- May simply be modifications of codes provided by manufacturer

**Outline of Elements of a Typical LabVIEW DACS Program**

Front Panel

The graphical user interface is constructed symbolically from defined functions

Wiring Diagram

The program structures appear as boxes that are defined sequences or loops

The actual program logic is constructed symbolically from defined functions

The flow of the program is as depicted by wiring

Figure 1: Outline of Elements of DACS Programs in BASIC and LabVIEW

The fundamental differences between programming with step-by-step sequential instructions ala BASIC, versus symbolic constructions in a graphical environment should be apparent from examining the outlines of elements of each method in Figure 1. The student may not be able to grasp the flow of a BASIC program that jumps into subroutines from a list of statements, then into binary instructions loaded somewhere in memory, and returning with streams of data into some other memory location. Complex but fully functional programs can be constructed in little time with minimal instruction using the newer programming environments such as LabVIEW. Such a program completed as a first assignment in LabVIEW is illustrated in Figure 2. The front panel includes controls, indicators and graphs that were chosen from a palette of functions that have pictorial descriptions on icons, with context sensitive Help a mouse click away!

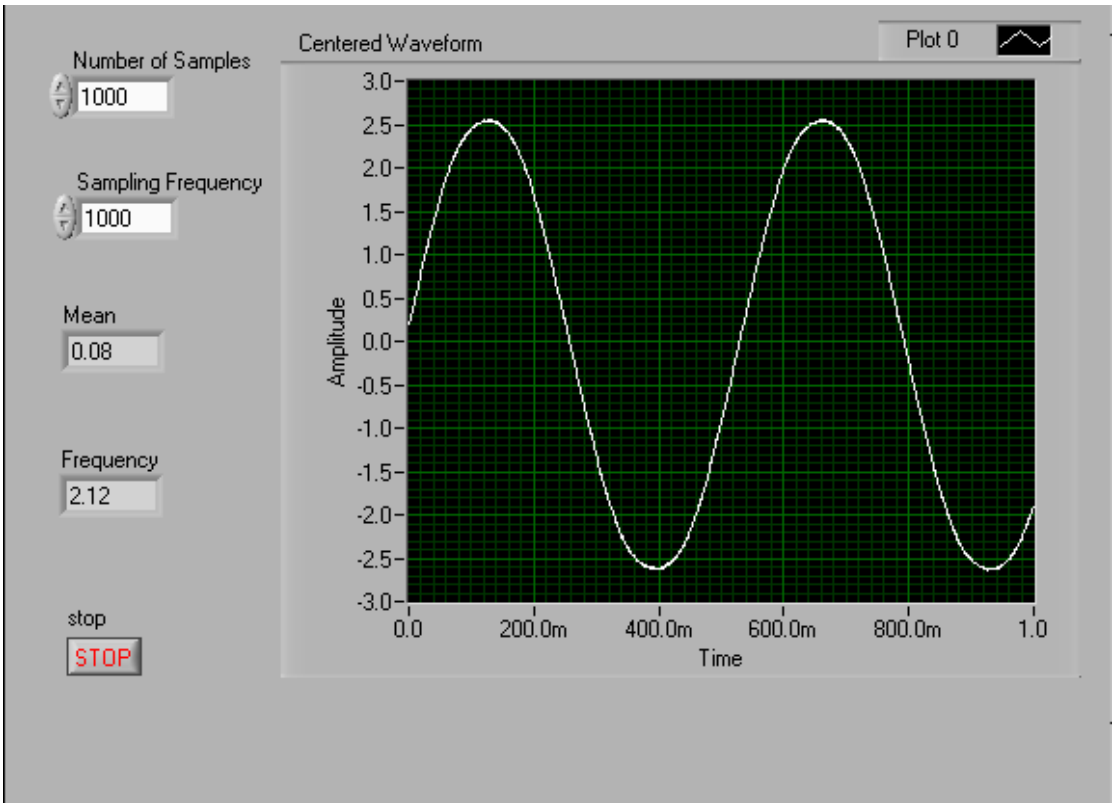


Figure 2: Front panel of a LabVIEW Data Acquisition Program

A quick look at the wiring diagram that depicts the programming logic is quite simple to understand, particularly if the context help function is active. When the function is

active, mouse-over of an icon on the diagram displays a very complete description of all optional connections. This wiring diagram shows the data flow from data acquisition card through functions such as averaging and frequency estimation, into a pre-defined, auto-scaling plot structure.

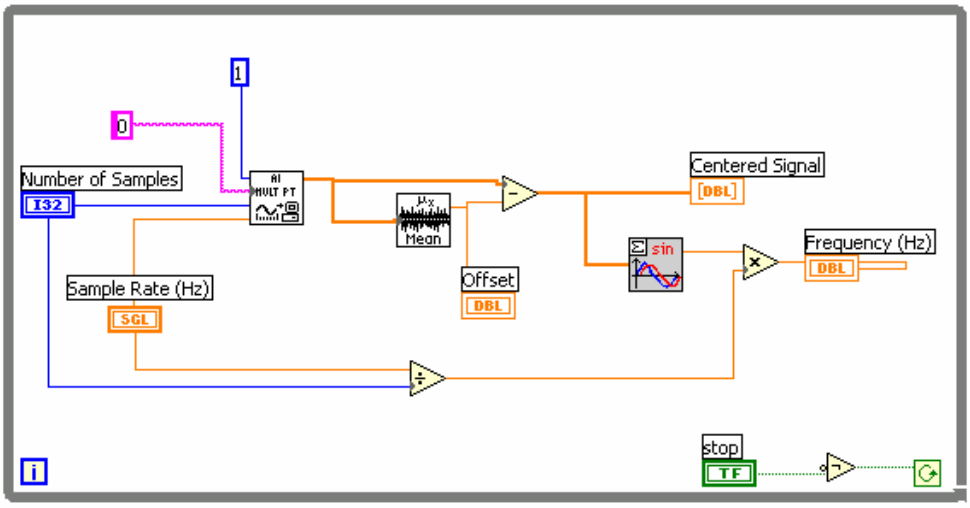


Figure 3: Wiring Diagram of the LabVIEW Program

Predefined data acquisition functions for waveform input from a single channel, statistical analysis function to calculate mean, and a Bunneman frequency estimator are just a few of the many data acquisition and signal processing functions available.

**Assessment**

The level of complexity that can be introduced and programmed by students who have no previous DACS knowledge now far surpasses the level that was attainable prior to the introduction of such tools. Even if complex subroutines and functions were provided, their interface was obscured in the nature of sequentially programmed instructions. Even with modular programming emphasis, the learning curve was often too steep for students unfamiliar with detailed programming to grasp. Use of rudimentary codes to illustrate the basic concepts is still invaluable to insure that students understand those concepts, but the level of competency to which the students can rise with one or two semesters is great. Details of example programs are reviewed, then small modifications are made, then complete re-writes of programs are performed. This building block approach, coupled with the ease of graphical programming environments has certainly caused student competency in this area to increase.

Assessment of course effectiveness and recommendations for course modifications are generally accomplished in an ongoing manner. Driven by accreditation requirements to insure that the course content is appropriate, current and effective, a textual summary of each course, a matrix of departmental objective accomplishment and recommendations to the curriculum committee are completed each semester. Over the past several years,

there has been a concentrated effort to include laboratory exercises and demonstrations into structures, fluid mechanics, aerodynamics and controls classes. Those classes generally accomplish "turn-key" labs, but those labs do provide additional familiarization with equipment and programs. Thus the students are receiving some additional motivation in learning the art behind the production of massive data streams. Typically students don't seem to have a complete grasp of data manipulation and reduction until they control all the details of data acquisition and analysis. Feedback from graduates who have been assigned to upgrade computer systems on research facilities used by NASA, Lockheed, and Boeing continues to indicate that the training received in upgrading hardware and translating programs across environments is time well spent.

One former student related that of the various individuals in his group working on taking a NASA wind tunnel out of mothballs, he was made lead because of his familiarity with multiple data acquisition systems, protocols, and programming environments. His experience at translating programs written in HP BASIC to TestPoint and LabVIEW was particularly useful. Another alumni who works for a aerospace research company, Bosch Aerospace, credited his aerospace lab experiences for his success in rapidly developing data acquisition and control hardware for a cycloidal propeller project. Learning to take an existing code and adapt it quickly for another application, or translating it into another programming environment was viewed as merely incidental to the task at hand instead of a project unto itself. A member of the advisory committee for the department related during his last visit that his experiences in these laboratory courses motivated and enabled his success in graduate studies involving experimental research. Another graduate who had worked on software development for the lab during his undergraduate and master's level studies also credited his laboratory experiences translating various programs with giving him all the necessary skills to succeed as a research engineer working on a project to develop a next generation blimp. In every case, these alumni rated their programming skills learned in the laboratory courses and experiments as being fundamental to becoming effective engineers in the workplace. More recent graduates have taken into the workplace not only those basic skills learned in these classes, but also a knowledge and experience gained working with the most common current-generation analytical tools, such as MathCAD and Matlab.

**Conclusions**

Teaching data acquisition and control system programming in multiple languages and environments is an effective way to emphasize the necessity for life-long learning. Skills obtained in modifying given programs and translating programs into graphical environments are effectively learned in the manner indicated. By not focusing on particular programming languages or styles, but rather, deliberately introducing multiple paths to the solutions to problems, the open-ended nature of engineering is effectively illustrated. The unique and different solutions shared between groups of students enhances learning and gives those students a perspective that cannot be found in a course that focuses on more narrowly defined programming solutions.

**Bibliographic Information**

1. National Instruments Web Site, 2002 Annual Reports Financial Highlights, http://www.ni.com/company
2. Keithley Instruments Web Site, 2002 Company Fact Sheet, http://www.keithley.com
3. Rais-Rohani, M., Koenig, K., Hannigan, T., "Keeping Students Engaged: An Overview of Three Introductory Courses in Aerospace Engineering", Proceedings of the 2003 ASEE Annual Conference & Exposition, Nashville, TN, June 22-25, 2003
4. Mathsoft Engineering & Education, Inc. Mathcad. 11. 2002
5. The Mathworks, Inc. MATLAB with SIMULINK. R13. 2002
6. National Instruments Corporation. LabVIEW. 6.0i. 2000

**Biographical Information**

THOMAS HANNIGAN
Thomas Hannigan is an Instructor of Aerospace Engineering and Engineering Mechanics. He received his BS and MS degrees from Mississippi State University. His interests include introductory engineering mechanics, airplane flight mechanics, and he coordinates laboratory activities for the department. He holds FAA Gold Seal Flight Instructor Certification for single, multi engine and instrument airplanes.

KEITH KOENIG
Keith Koenig is a Professor of Aerospace Engineering. He received his BS degree from Mississippi State University and his MS and PhD degrees from the California Institute of Technology. Prof. Koenig teaches courses in aerodynamics and propulsion. His research areas include rocket and scramjet propulsion and sports equipment engineering.

BRYAN GASSAWAY
Bryan Gassaway is a lecturer and PhD student in the Aerospace Engineering Department. He received his BS and MS degrees from Mississippi State University. He teaches astrodynamics and has taught courses introducing aerospace engineering, flight mechanics, stability and control, structures, propulsion and astrodynamics, as well as assisting with the laboratory classes.

VIVA AUSTIN
Viva Austin is a graduate teaching assistant in the senior aerospace engineering laboratories. She obtained her BS degree in aerospace engineering from Mississippi State University, and is currently enrolled as a candidate for a master of science degree. She assists in teaching upper division laboratory classes as well as assisting in the conduct of laboratory activities for three lower division introductory classes.