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
Mississippi State University

Abstract
Keeping data acquisition and control systems (DACS) used in a graduate and undergraduate 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\textsuperscript{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,\textsuperscript{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 the students 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 that include such topics as introductions to MathCAD and Matlab, 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. 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.

<table>
<thead>
<tr>
<th>Pascal</th>
<th>Basic (PC DOS)</th>
<th>CPM BASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbo Basic</td>
<td>Power Basic</td>
<td>BASICA</td>
</tr>
<tr>
<td>Mbasic</td>
<td>GWBasic</td>
<td>HPBasic</td>
</tr>
<tr>
<td>QuickBasic</td>
<td>VisualBasic</td>
<td>C/Perl</td>
</tr>
</tbody>
</table>

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

In addition, programming was accomplished in the environments of ASYST, Labtech Notebook, Testpoint, and LabVIEW. 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 would limit the number of topics introduced in the lab, and might arguably allow a more complete familiarization with that environment. However, 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. 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-Metabyte, 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.

<table>
<thead>
<tr>
<th>Laboratory Task</th>
<th>Demonstrated Methods</th>
<th>Student Solution Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plotting calibration data, linear regression, statistical analysis</td>
<td>BASIC, C, ForTran programs, Excel Spreadsheet</td>
<td>Excel, MathCAD, BASIC, C, ForTran programs, Matlab program</td>
</tr>
<tr>
<td>Plotting experimental airfoil pressure data, determination of sectional airfoil properties</td>
<td>LabVIEW DACS, BASIC data reduction and analysis, EXCEL, Matlab data reduction and analysis</td>
<td>Excel, MathCAD, Matlab, BASIC, C programs</td>
</tr>
<tr>
<td>Analysis of primary wind tunnel data for stability derivatives</td>
<td>BASIC, C, MathCAD</td>
<td>BASIC, Fortran, C, Matlab, MathCAD</td>
</tr>
<tr>
<td>Data acquisition of pressure distributions in low subsonic pipe flow and in a blow-down supersonic converging-diverging nozzle, comparison to theoretical value</td>
<td>BASIC, LabVIEW for data acquisition, BASIC, LabVIEW, MathCAD analysis</td>
<td>BASIC, Excel, MathCAD analysis</td>
</tr>
<tr>
<td>Data acquisition, control of a portable wind tunnel, display of airfoil pressure distribution</td>
<td>BASIC, Testpoint, LabVIEW</td>
<td>Testpoint, LabVIEW</td>
</tr>
<tr>
<td>Data acquisition and control of arbitrary peripheral</td>
<td>Machine/Assembly language w/DEBUG, BASIC, C, LabVIEW</td>
<td>Assembly language, BASIC, C, LabVIEW</td>
</tr>
<tr>
<td>Data acquisition of arbitrary waveform, plotting</td>
<td>BASIC, Testpoint, LabVIEW</td>
<td>BASIC, Testpoint, LabVIEW</td>
</tr>
</tbody>
</table>
Table 2: Methods demonstrated and used for typical laboratory task completion

Assessment
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.

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
5. The Mathworks, Inc. MATLAB with SIMULINK, R13. 2002

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.