ENHANCING STUDENT LEARNING IN AN INTRODUCTORY EMBEDDED SYSTEMS LABORATORY

Aaron Striegel[1] and Diane T. Rover [2]

Abstract *— As technology advances, curriculum and laboratories are challenged to keep pace. This is especially true in computer engineering, where the range of technologies is constantly broadening and diversifying, as computer-based systems take on many forms and functions in everyday life. The question is, how should a contemporary curriculum train computer-engineering students for the vast scope of embedded system solutions? In this paper, we specifically consider where to begin, and ask, can powerful tools empower students to learn? We describe steps taken at Iowa State University to upgrade a sophomore level laboratory in embedded systems. We migrated from the popular 8-bit Motorola 68HC11 microcontroller as the core for a hardware-software development platform to the emerging 32-bit PowerPC 555 microcontroller. The 68HC11 is used in labs at numerous universities across the country and is supported with textbooks and educational packages. Conversely, the PowerPC is new to the academic environment and comes with a rich set of features and development tools. In this paper, we examine the similarities and differences between the laboratory platforms and their impact on student learning. We also identify strengths and weaknesses of the new laboratory environment, based on our own perspective and student feedback.*

Index Terms — Computer Engineering laboratory, embedded systems instruction, Motorola 68HC11, PowerPC 555.

INTRODUCTION

As technology advances, curriculum and laboratories are challenged to keep pace. This is especially true in computer engineering, where the range of technologies is constantly broadening and diversifying, as computer-based systems take on many forms and functions in everyday life. The question is, how should a contemporary curriculum train computer-engineering students for the vast scope of embedded system solutions? In this paper, we specifically consider where to begin, and ask, can powerful tools empower students to learn?

At Iowa State University, this was the question facing the faculty regarding an undergraduate course in embedded systems. Currently, the Electrical & Computer Engineering Department at Iowa State has two courses in computer engineering focusing on embedded systems, Cpr E 211 and Cpr E 588. Whereas Cpr E 588 is primarily suited for graduate students and seniors, Cpr E 211 is a required class for all computer engineering (CprE) and electrical engineering (EE) majors. Thus, Cpr E 211 became the starting point for focusing on this fundamental question.

COURSE OVERVIEW

The course description for CprE 211, Introduction to Microcontrollers, reads as follows:

> *Logic families. Documentation standards. Implementation and testing of combinatorial and sequential systems and subsystems. Introduction to microcontrollers. Microprocessor registers, memory, and programmable input/output devices. Interrupts. Single chip controllers. Design and testing of software for microcontrollers. Hardware/software design tradeoffs and issues. Individual design projects.*

The class meets three hours every week. The laboratory meets once weekly for two hours. By the end of the CprE 211 course, students should be able to:

- Demonstrate knowledge of a microcontroller and how it operates
- Program in C and assembly code
- Understand how C is converted to assembly code
- Understand basic concepts of microcontrollers
- Understand basic computing concepts such as interrupts, interrupt service routines, and input/output (I/O) subsystems
- Perform basic hardware and software debugging
- Work with and design basic embedded systems

The general learning objectives for the course include:

1. Enable the student to design software to interact with real-world systems
2. Familiarize the student with interfaces between the real-world and a digital system
3. Provide the student with information to work effectively in the laboratory
4. Familiarize the student with typical engineering activities

[1] Aaron Striegel, Department of Electrical & Computer Engineering, Iowa State University, Ames, IA 50010 adstrieg@iastate.edu
[2] Diane T. Rover, Department of Electrical & Computer Engineering, Iowa State University, Ames, IA 50010 drover@iastate.edu

0-7803-6669-7/01/$10.00 © 2002 IEEE

5. Provide students with the interactive skills needed to work in groups

THE MOTOROLA 68HC11 PLATFORM

The Cpr E 211 course [1] used the 68HC11, a popular 8-bit microcontroller produced by Motorola. The 68HC11 is used in labs at many universities across the country and is supported with numerous textbooks and educational packages. For our laboratory, we used a platform known as the F1 Board [2], as shown in Figure 1, using the 68HC11F1 variant coupled with basic I/O functionality that included:

- 32 KB of RAM (24 KB usable), 32 KB of ROM
- BUFFALO Monitor software (ROM)
- Serial input/output for downloading programs
- 2 x 20 LCD screen
- 8 x 5 keypad (A-Z, 0-9, etc.)
- 8-bit digital input (DIP switch)
- 8-bit digital output
- 8 A/D inputs (0-5V)
- Timing (Periodic Interrupt, Pulse Width Modulation, Time Measurement) with removal of I/O board

Programming for the microcontroller was provided by the ICC11 v5.0 software package [3] with support for C and assembly. The ICC11 v5.0 software package provided an integrated development environment (IDE) with support for compiling and serial downloading.

The F1 board was first introduced in fall of 1995 to the ISU Computer Engineering curriculum with an initial purchase of 20 F1 boards. Since the fall of 1995, we upgraded the ICC11 software from version 2.0 to version 5.0 and enhanced the durability of the F1 board by attaching a potentiometer, socketing the I/O board, and tilting the LCD screen (see Figure 1). Although such changes helped improve the board, the F1 board yielded the following problems:

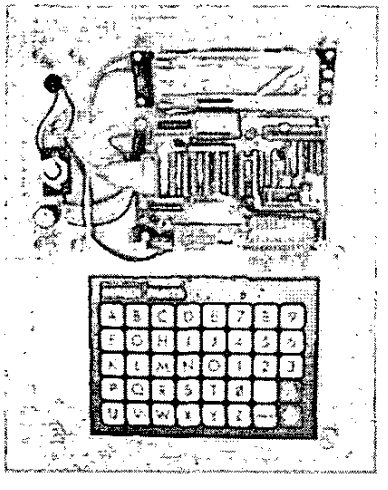

FIGURE 1
F1 BOARD (68HC11)

- *High failure rate* – Chips on the I/O board frequently failed due to either natural causes or incorrect wiring of external student circuits.
- *Limited I/O* – In order to support larger scale projects (semester-long projects), the I/O was extremely limiting (e.g., only 8-bit input/output).
- *Lack of debugging capability* – Debugging under ICC11 v5.0 was limited to only output via the serial port or LCD screen in the program (printf, etc.).

As a result of these problems, we decided to investigate alternatives for the laboratory platform. In the spring of 2001, a Senior Design team [4] was commissioned with the task of investigating and developing an alternative solution for the laboratory platform.

NEW PLATFORM ALTERNATIVES

Rather than focusing on an incremental improvement in the lab facilities, we set a goal to not only increase the utility of the laboratory (functionality, robustness), but also to increase the educational value of the laboratory (ease of use, quality of tools, future value). The main objective was to select and develop a platform that would allow students to creatively explore embedded system fundamentals while minimizing some of the usage and maintenance frustrations often associated with such laboratories. Thus, we outlined the following criteria for the Senior Design team:

- *Low failure rate* – The new platform must have fewer failures than the F1 board. The platform should safe-guard against student-built circuits and should be designed for easy repair.
- *Expanded I/O* – The platform must enhance the I/O facilities versus the existing F1 board.
- *Enhanced debugging* – The platform must be supported by an upgraded IDE with step-by-step debugging for both C and assembly programming.
- *Ease of use* – The platform must be usable by sophomore-level students in Computer Engineering and should not require extensive additional training.
- *Longevity* – The new platform should last a minimum of 3-5 years without a significant investment in new hardware/software.

After several months of research, the team narrowed down the platform alternatives to three options that are discussed in the next subsections and summarized in Table 1.

Motorola 68332 Platform

In the fall of 2000, the junior-level course in embedded systems using the Motorola 68332 was phased out and transitioned into other classes. As a result, the department had 12 unused prototype boards for the Motorola 68332. The Motorola 68332 is a 16-bit microcontroller with support for timing via a TPU (Timing Processing Unit) included on the chip. Over the lifetime of the course, various instructors had developed several basic prototype boards. The programming environment was based in a Linux system environment using gcc and an on-line monitor for serial downloading/execution on the 68332. This alternative would require the Senior Design team to design a custom I/O board for the 68332 prototype board, develop software libraries for the I/O board, and acquire GUI tools for the board.

This solution presented several problems. First, the 68332 chip was being discontinued by Motorola and being consolidated into the 68334 line. Second, the prototype boards themselves were out-dated, thus making it difficult to acquire replacement parts should one of the boards fail. Finally, the tools for the Linux platform were extremely limited, operating at roughly the same level as the current setup.

PowerPC – Linux Platform

As a second alternative, a local software vendor offered an embedded platform using a variant of Linux (Hard Hat Linux [5]) running on a PowerPC microprocessor. The board included support for serial and Ethernet communications, custom Linux kernel development, and a 64K-gate FPGA. Program development would take place in the Linux environment on a host workstation through an IDE with debugging supported using gdb (GNU Project Debugger [6]).

The platform included several benefits not present in the current platform. First, the embedded Linux solution allowed freeware GUI programs to provide step-by-step debugging and program editing. Second, the FPGA would allow for a natural extension from the digital logic course (Cpr E 210), a prerequisite for Cpr E 211. Finally, the local software contact would be able to provide extensive technical support and assistance in installing the software.

MPC 555 – CodeWarrior Educational Package

While evaluating the second alternative, the design team discovered an educational package being offered by Metrowerks (a subdivision of Motorola) for the PowerPC microprocessor. The package includes the CodeWarrior IDE [7] for Windows and a prototype board for the PowerPC/MPC 555 microcontroller from Axiom [8]. For the MPC 555, code is written and debugged using the CodeWarrior IDE. The CodeWarrior IDE interacts with the MPC board using either a serial connection or a BDM (Background Debugging Mode) connection via the parallel port.

This alternative consolidated many of the features being

TABLE I
SUMMARY OF ALTERNATIVE PLATFORMS

Description	Processor	Development Environment	Strengths	Weaknesses
Current laboratory platform	Motorola 68HC11 (8 bit)	ICC11 v.50	- Large number of textbooks - Easy to learn - Already operational	- Robustness - Limited I/O - Limited debugging
Upgrade of boards in related laboratory	Motorola 68332 (16 bit)	gcc/EMACS	- Already operational - More powerful microcontroller	- No custom I/O board - Maintainability due to discontinuation of processor
Linux-based laboratory platform	Power PC 602 (32 bit)	IBM IDE gdb	- Local software vendor - Integrated FPGA - GUI for debugging	- Not a microcontroller - No custom I/O board - Complexity of Linux for students
Platform based on Metrowerks Educational Package	Power PC 555 (32 bit)	CodeWarrior 6.0 IDE (Windows)	- Low cost - Similar to 68HC11 feature set (e.g., I/O) - More powerful microcontroller - Excellent IDE	- No custom I/O board

sought into a single platform. First, the programming and debugging are contained entirely within the CodeWarrior IDE. Thus, with a setup similar to Visual C++, the students are exposed to a state-of-the-art programming environment similar to one that they might see in industry. In addition, such an environment is very easy to use and includes debugging features that support the course learning objectives. Second, the MPC 555 is a fully featured microcontroller, offering all of the features of the 68HC11 with a PowerPC core. Finally, the prototype nature of the MPC board facilitates development of an expansion board.

LABORATORY PLATFORM SELECTION

The four alternative platforms are listed in Table I: current 68HC11 laboratory platform, upgrade of 68332 laboratory platform, PowerPC Linux-based platform, or MPC/CodeWarrior platform. The platform selection was further constrained in that any lab development must be completed and operational for classes in the fall semester of 2001, leaving only part of the spring and the summer for development and testing.

The 68332 platform alternative was the least costly, however, we decided that such an upgrade would not significantly increase the educational value of the class.

For both of the PowerPC platforms, several risks were identified. First, there are few textbooks available for instruction on the PowerPC. Although there is a wide variety of textbooks on assembly programming for RISC processors, the majority of the textbooks are targeted towards a MIPS-like processor. Thus, for the near future, the students would be relying on in-class notes, manuals, and marginally-related textbooks. Second, the switch to a RISC-based processor with a rich resource set (general purpose processor, many registers) would represent a fundamental shift in the instruction. The 68HC11 has provided students with a clear perspective in a CISC-style architecture found in many embedded devices. The PowerPC represents a departure from this approach.

However, upgrading to a PowerPC architecture also offers several significant benefits. First, the debugging environment is a substantial improvement over the current setup. Such state-of-the-art tools expose students to contemporary industry practices. There are more options to choose from and higher quality tools available. In addition to helping reduce student frustration in the laboratory, the tools also foster a creative environment, allowing students to explore and learn about embedded systems. Second, the switch to a RISC-style architecture exposes students to future trends in microcontrollers. As most RISC assembly languages are quite similar to one another, the general concepts can be applied across similar microcontrollers (StrongARM, MIPS, etc.). Third, the core concepts of the labs need not significantly change. Although the lab platform would significantly change, the fundamental concepts of C programming, assembly programming, and interrupts would still drive the laboratories. Thus, with the two PowerPC-based alternatives, we felt confident that we could offer a significant educational benefit that would enhance student learning in the course as well as the quality of the laboratory experience.

The next task was to evaluate the two PowerPC-based platforms and make a selection. The Linux platform suffered from several notable drawbacks compared to the MPC platform. First, although freeware programs were available to support the necessary debugging and editing features under Linux, the lack of integration between the tools made the interface awkward and difficult to use. Second, although the FPGA would offer unique features, the basic digital I/O functionality would still need to be mapped out via a customized I/O board as with the MPC platform. Third, the PowerPC 602 is a general-purpose microprocessor that does not include any on-board or on-chip I/O support, as compared to a microcontroller. Thus, for teaching concepts such as A/D conversion and timing, a customized solution would need to be developed using a combination of the FPGA and I/O board. Finally, although the Linux kernel support would offer an ideal platform for students to learn Linux, the complexity of the software system would unnecessarily complicate the course contents given the level of students and course learning objectives. The course itself is geared towards teaching the basic concepts of memory-mapped I/O, assembly programming, and interrupts, and thus additional complexity is better suited for other courses[3].

LAB DEVELOPMENT

Thus, in May 2001, we opted for the MPC/CodeWarrior platform by Motorola and began to develop the laboratory for deployment in the fall of 2001. The laboratory platform consists of a Windows 2000-based host workstation, the CodeWarrior IDE, and a customized PowerPC-based unit, as shown in Figure 2. The PowerPC-based unit consists of a frame, MPC 555 prototype board, BDM connection to the host, and a custom I/O board. The implementation was undertaken as the design team's project and completed by a graduate student and technician during the summer 2001.

Given the time constraints, the following features of the custom I/O board are implemented:

- 16 MB of on-board RAM
- 4 10-bit A/D channels via mounted potentiometers
- 2 8-bit DIP switch inputs
- 2 8-bit latched outputs (current-limited)
- 2 8-bit buffered inputs
- 1 4x4 hex keypad
- 2 8-bit LED bargraphs
- 2 7-segment displays with hex decoding

[3] For the junior-level course in operating systems (Cpr E 308), the Linux-based solution was adopted in the fall of 2001.

- 1 7-segment display with no decoding
- QSI terminal [9] with integrated serial communications
- 4x20 LCD screen
- Full alphanumeric keypad with F1-F5 keys
- BDM connection for debugging
- Serial connection for input/output
- 4 bits of TPU timing input/output
- 4 bits of general timing input/output (PWM, OC, IC)

For the I/O directly on the MPC 555 microcontroller (serial, BDM interface, A/D, TPU, timing), the pins are brought directly out to the I/O board. The other I/O operations are supported through memory mapping of the inputs and outputs. In addition, we developed software libraries to initialize the I/O devices and sample code for students to use. The final version was tested through two revisions and assembled for use in the fall 2001 semester offering of Cpr E 211. Figure 3 depicts several tools and views available in the CodeWarrior IDE. Figure 4 shows a photo of a completed PowerBox unit, as used in the lab.

OBSERVATIONS

During Fall semester 2001, 120 students were introduced to the new laboratory in Cpr E 211. The students consisted of a mix of roughly 75% CprE's (sophomore level) and 25% EE's (sophomore through senior level). The textbook and course materials included Goodman's text on RISC assembly programming [10], the MPC 555 assembly programming manual [11], and notes available online from the course website. The course was organized similar to its previous offerings in terms of content on embedded C programming, assembly programming, interrupts, and I/O devices and interfacing.

Lab sections consisted of 16-20 students led by a single undergraduate or graduate TA. Both of the undergraduate TAs were members of the design team that developed the board. The labs were subdivided into three sets of concepts

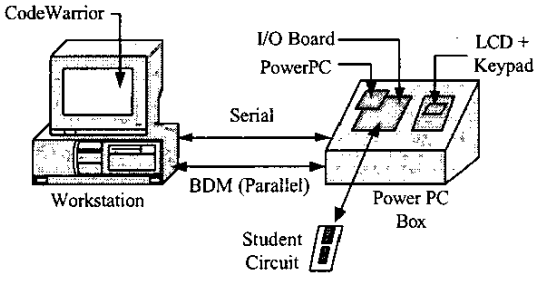

FIGURE 2
LABORATORY SETUP

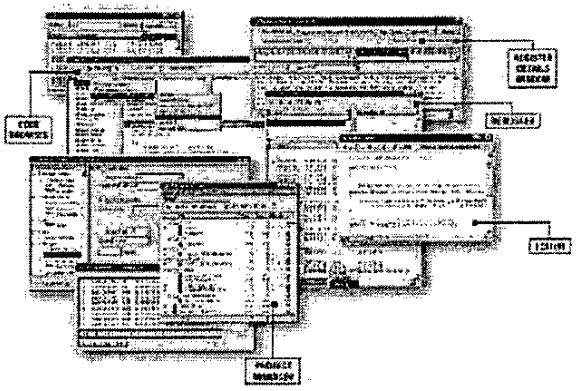

FIGURE 3
CODEWARRIOR: SAMPLE SCREENSHOTS

and skills, namely C programming (40%), assembly programming (30%), and I/O programming (30%). The labs followed a central theme to add real-world relevance to the topics under study, in this case, that of a police cruiser controller. The labs culminated in comprehensive final project (combining all labs) at the end of the semester. For each lab, the students worked in pairs, and for the project, in larger groups of 5-6 students. The weekly labs for the police cruiser controller were [14]:

Lab 1. Introduction to CodeWarrior/PowerPC
Lab 2. Digital Logic & I/O Interfaces
Lab 3. Car Odometer
Lab 4. Glove Compartment Lock
Lab 5. Police Cruiser Light Control
Lab 6. Introduction to Assembly on the CodeWarrior/PowerPC Platform
Lab 7. 7-Segment Display Decoder
Lab 8. Police Cruiser Communications: Serial Query/Response Terminal
Lab 9. Police Cruiser Speed Radar: A/D I/O
Lab 10. Real-Time Interrupts

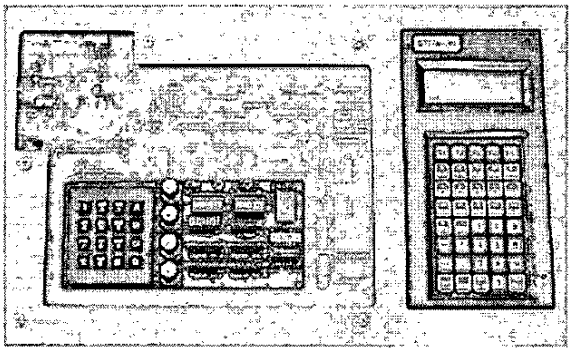

FIGURE 4
POWERBOX: POWER PC-BASED PLATFORM

Laboratory instruction and exercises were updated to take advantage of the platform's features. CodeWarrior supports mixed C-assembly programming and advanced debugging. Exercises and projects can tap into the array of I/O devices and interfaces on the PowerBox. The PowerPC architecture was leveraged to some extent, as described in the section on Laboratory Platform Selection.

As the labs progressed over the semester, we conducted several surveys to obtain student feedback on the new platform. In addition, we compared the performance of students on similar labs from previous semesters. Two important results were observed.

First, student learning in the labs is improved both in terms of depth and breadth. For example, rather than focusing on tedious issues with debugging, the advanced tools allow the students to investigate interesting phenomena or to simply expedite the debugging of their code. The register viewing and step-by-step debugging of assembly code are especially useful to flatten the steep learning curve for assembly language programming. More time can then be spent on design and analysis issues. The BDM debugging features are particularly valuable when developing interrupt-driven programs, helping students work through a complicated but interesting subject. In fact, students were highly motivated to work with interrupts, due in part to the understanding and capabilities provided via using the tools.

Second, the advanced tools and overall scope of the labs allow students to explore issues beyond what is typically covered in an introductory microcontroller course, such as usability and performance issues. For example, students experimented with timing for keystrokes and effective use of the LCD screen. In addition, students not tethered to a restrictive environment begin to think beyond the mechanics of the procedures and the minutiae of the design and to grasp system-level design concepts.

Finally, the boards proved to be extremely reliable over the semester. The current-limiting design prevented student circuits from overloading the board and almost all of the components successfully survived the semester without need for repair. The only item requiring limited repair were several potentiometers (knobs stripped during a freshmen-level lab also held in the same room).

Future Work

Additional capabilities can be added to the platform with an expansion board. Although not planned for the near term, over the longer term, we are interested in adding the following features:

- *Analog/Digital interfacing with a PC* – This would allow for significantly more complex labs as the physical environment could be modeled using a program such as LabView [12].
- *Networking interface* – By adding a network interface such as Bluetooth or IEEE 802.11, the networking class could also use the laboratory equipment for testing protocols. In addition, Cpr E 211 students could be introduced to simple wireless device communications through lab projects.

FINAL COMMENTS

In conclusion, we successfully adapted an introductory embedded systems class to the PowerPC processor from the Motorola 68HC11, and in the process, enhanced the learning environment. Although the MPC 555 is relatively new to education, we believe such a laboratory environment can easily be adapted to other universities. The full scope of the laboratory evolution can be found at the Senior Design website [13].

ACKNOWLEDGMENT

The members of the Senior Design team responsible for platform research and development were Jon Froehlich, Brad Hottinger, Derek Miller, and Dan Murr. Professor Arun Somani initiated the laboratory development and mentored the design team. The teaching assistants in the CprE 211 laboratory during Fall 2001 were Sriram Nadathur, Jon Froehlich, and Derek Miller. ECE Department Electronics Services and electronics technician Jason Boyd supported development of the PowerBox.

REFERENCES

[1] CprE 211 website, http://class.ee.iastate.edu/cpre211/01fall/, Fall 2001

[2] F1 Board website, http://www.cncteknix.com/users/f1

[3] ImageCraft website, http://www.imagecraft.com/software/

[4] ISU EE/CpE Senior Design website, http://seniord.ee.iastate.edu/

[5] MontaVista Software website, http://www.mvista.com/

[6] GNU Project Debugger website, http://www.gnu.org/software/gdb/gdb.html

[7] CodeWarrior Embedded PowerPC website, http://www.metrowerks.com/embedded/powerpc/

[8] Axiom Manufacturing website, http://www.axman.com/

[9] QSI Corporation website, http://www.qsicorp.com/

[10] J. Goodman, K. Miller, A Programmer's View of Computer Architecture: With Assembly Language Examples from the Mips Risc Architecture, HBJ College & School Division, Jan 1993.

[11] RCPU Reference Manual, MPC500 Family, Motorola, 1994

[12] National Instruments website, http://www.ni.com/

[13] Cpr E 211 Senior Design website, http://seniord.ee.iastate.edu/dec0104/, Fall 2001

[14] A. Striegel and D. Rover, "Problem-Based Learning in an Introductory Computer-Engineering Course", *Proc. 32nd ASEE/IEEE Frontiers in Education Conference*, Boston, Nov. 2002.