Improving the Effectiveness of Microcontroller Education

Robert B. Reese and Bryan A. Jones
Department of Electrical and Computer Engineering
Mississippi State University
Starkville, MS USA
{reese, bjones}@ece.msstate.edu

***Abstract*—The increased power and functionality of modern microcontrollers provides both an opportunity and a challenge to educators. Increased complexity, the necessity of integrating a hands-on lab experience, and downward pressure on total curriculum hours demand a significant investment of time to modernize microcontroller education, which few educators can afford. However, a successful microcontrollers course provides a unique opportunity to equip students to produce large, complex systems in their capstone design course. This paper details the evolution of an approach which provides an integrated course and laboratory experience for students. Results of these changes produced quantitative differences in the resulting capstone design courses, validating the approach.**

I. Introduction

In modern embedded digital systems, a microcontroller is as fundamental a component as transistors and individual logic gates were 30 years ago. It is difficult to envision any type of digital system other than the most simple that does not incorporate a microcontroller (or multiple microcontrollers). Shrinking transistor geometries have allowed microcontrollers to incorporate tens of subsystems around the CPU core, making modern microcontrollers extremely flexible in their capabilities. It was thought in the 90s that Field Programmable Gate Arrays (FPGAs) would eventually eclipse microcontrollers as the platform of choice for digital system implementation, but this has not happened because of the increasing speed, increasing flexibility, lower power consumption, faster prototyping, wider application range, and cheaper cost of a microcontroller solution. However, the increasing complexity of these devices leads to challenges for the instructor.

Effective teaching of microcontroller programming and hardware interfacing is challenging within EE and CPE programs for several reasons. First, a modern microcontroller is a complex system, with datasheet, programming reference, and family reference materials totaling over one thousand pages. Becoming familiar with a microcontroller family at the detail necessary for course integration is a time-consuming task for an educator. Second, microcontroller education still requires a significant hands-on laboratory experience despite advances in system simulation. Developing this laboratory experience requires an even greater time investment by the educator. Finally, downward pressures on total curriculum hours has forced educators to think of creative ways to combine traditional two-course sequences such as computer organization/assembly language programming, followed by a microcontroller/hardware interfacing course, into a single course.

II. Background

These barriers hamper many educators in their efforts to improve the effectiveness of microcontroller education. Table I shows a sampling of microcontroller content at institutions that competed in the SECON 2008 student hardware competition. The schools listed are those had detailed course information accessible from the web; general catalog information was not used. In general, only the EE requirements were checked, since almost all programs had the microcontroller course required of computer engineering majors.

As detailed in the course sampling of Table I, 67% of the surveyed schools employ assembly language in their curriculum, hampering student understanding of the nuances of hardware subsystems. For example, simple statements in C such as “if (i8_a < i32_b)” mask important architectural features such as sign extension, varying data with (8 vs. 32 bits), signed vs. unsigned comparison, conditional branches, and register allocation and assignment. Only 56% require microcontroller interfacing, limiting student ability to design complex systems in a capstone design course. As a result, students find the intelligence of their designs constrained to a handful of comparators, as shown in Figure 3. In contrast, Figure 1 illustrates the design complexity students achieve based on a firm grasp on the theory and application of microcontrollers;

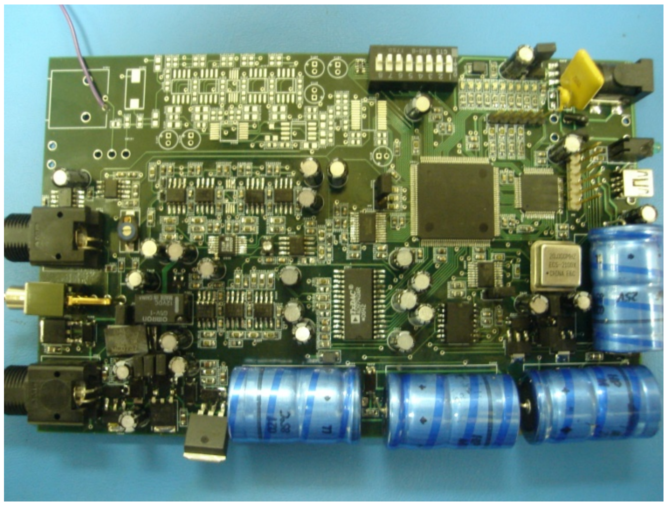
Figure 1. Photo of completed project, capstone course, Spring 2008.

TABLE I. A SAMPLING OF MICROCONTROLLER REQUIREMENTS AT VARIOUS INSTITUTIONS

Institution	Course	Processor
University of Alabama (Tuscaloosa)	ECE 383 Microcomputers	Freescale MC9S12DP256B, C
University of Kentucky	EE 383 Embedded Sys.	Freescale 68HC12, asm
Auburn	ELEC 2220 Computer Sys.	Freescale 68HC12, asm
Univ. of Tennessee (Knoxville)	ECE 355 Computer Sys. Fund. ECE 455 Embedded Progr.	PowerPC family, asm + C, TI MSP 430, C
University of South Carolina (EE Dept)	CSCE 212 Intro. to Comp. Arch. CSCE 313 Embedded Sys. Lab	asm, no lab Freescale 68000
University of South Alabama	EE 264 Microprocessors and Interfacing	Freescale 68HC11, asm
Tennessee Tech. Univ.	ECE 3120 Microcomputer Sys\.	Freescale HCS12, asm + C
Embry-Riddle Aeronautical University	CEC 320 Microprocessor Sys.	Freescale 68000, asm
University of Louisville	ECE 412	Freescale 68HC11, asm
The Citadel	ELEC 330 Digital Sys. Eng.	Freescale 68HC11, asm
University of Central Florida	EEL 4767C Computer Sys. Design	Freescale MC68HC11, asm
Old Dominion University	ECE 446 Microcontrollers	Freescale MC68HC11, asm
Clemson	ECE 371 Microcomputer Interfacing	Freescale MC9S12DP256B, C
North Carolina State University	ECE 306 Intro. to Embedded Sys.	Renesas M16C/16P, C
Florida State Univ.	EEL 4746 Microprocessor Based Sys. Design	Freescale MC68HC11, asm
Virginia Commonwealth	EGRE 364 Microcomputer Sys.	Atmel ATMega, asm
Florida International	EEL 4746 Microprocessor Based Sys. Design	Intel 80386DX, C
University Of Florida	EEL 4744C Microprocessor Applications	Freescale 68HC12, asm

In addition, 78% of the schools employ Freescale processors which provide a very limited selection of DIP-packaged ICs, further complicating capstone design efforts.

The strong emphasis on an assembly-only approach, taken by 67% (12/18) of the schools, follows the standard approach taken by textbooks written in the 90s and early in this decade [1, 2]. The reason for this was that on-chip program space and data RAM was limited, so microcontroller programs had to be highly optimized. Also *C* compiler choices for microcontrollers were limited. However, modern microcontrollers now have large amounts of program memory, with even lower-end 8-bit microcontrollers commonly having 16 Kbytes of flash memory or more, and data RAM in low multiples of Kbytes.

Only 56% of the sampled institutions (10/18) require microcontroller interfacing for EE majors; the rest have it as a either a Junior or Senior elective. All of the curriculums that have it as an elective have a computer organization course as a prerequisite. Therefore, many students find themselves poorly prepared to execute complex designs for their capstone design course.

Freescale (formerly Motorola) processors are used in most of the embedded courses (14/18, or 78%), due to Motorola's domination of the embedded market in the late 80s and 90s. Many good assembly language textbooks were written for Freescale processors during that time frame, so they were a natural choice for embedded courses. A good university program by Freescale and inertia in the academic environment has enabled them to maintain that presence. The predominance of assembly language programming for Freescale processors is historical because C compilers for Freescale were initially only offered by third parties, with spotty support for student editions. This has changed in this decade, with the CodeWarrior suite now offered in student editions that have a limit on maximum program size and number of files. Support of DIP packages for prototyping on Freescale's various families remains poor.

From the data in Table I, it would appear that the approach proposed in this paper could be of value to educators who are currently using their microcontroller course for teaching assembly language programming, and for those who are also using assembly language for the hardware interfacing portion of the course. This paper outlines a method more fully covered in [3] which maintains an exposure to assembly programming for their students, but uses the C language for the hardware interfacing topics. References [4-5] provide additional examples of microcontroller texts published after 2000 whose principal programming language is *C*.

A. Trends in Microcontroller Development and its Educational Impact

Modern microcontrollers feature tens of subsystems clustered around a CPU core. A list of the subsystems for a PIC24HJ128GP502, which is available in a 28-pin DIP, are:

- 128 KiBytes of non-volatile program memory, 8 KiBytes of SRAM
- Five 16-bit timers
- Four PWM and output compare channels
- Two UARTs (asynchronous serial communication)
- Two Serial Peripheral Interface (SPI) ports (synchronous communication)
- One Controller Area Network (CAN) port (synchronous communication, used in automotive industry)
- One Inter-Integrated Circuit (I^2C) port (synchronous communication)
- One Cyclic Redundancy Check (CRC) generator
- One 10-bit/12-bit ADC with multiple channels and DMA
- One real-time clock module

- General purpose IO

From an educational viewpoint, this single device is easily capable of filling an entire semester's worth of hardware interfacing experiments; in fact, it contains more capability than can be covered in a single semester. This classes of devices can be considered a Super Microcontroller when compared to the devices of the 90s.

Coupled with Super Microcontrollers has been the emergence of Super Peripherals, which are peripherals with SPI or I^2C serial ports that encapsulate complex communication protocols and physical interfaces for standard communication links such as Zigbee wireless [6], Wireless Ethernet 802.11 [7], or USB [8]. These Super Peripherals actually house an internal microcontroller and its associated software stack to implement the peripheral function, simplifying system design by keeping the software complexity of these communication links out of the main microcontroller's code space. It also brings more parallelism to the system because the system contains multiple microcontrollers, with different microcontrollers dedicated to specific communication systems. The Super Microcontroller with multiple UART, SPI, I^2C, or CAN ports combined with Super Peripherals allow a building block approach to system design.

III. Instructional approach

Beginning Fall 2003, the Electrical & Computer Engineering Department at Mississippi State University shifted its introductory microprocessor course from a traditional assembly language orientation (x86-based) to one that emphasizes embedded system concepts and hardware/software prototyping skills [9]. The course is required for computer science, electrical engineering, computer engineering, and software engineering majors and is 4 credit hours (3 lecture, 1 lab).

Students are provided instruction in assembly language programming, but are not expected to become experts. All assembly language examples are given as code converted from C language snippets. Assembly language labs are specified as C programs, with the students acting as human compilers for implementation. This removes the mystery of the C to assembly language link, and prepares the students for the hardware labs that are implemented entirely in C. Using C for the hardware labs provides greater code clarity and allows for more complex examples. Grasping the hardware interfacing issues is difficult enough without adding the extra complexity of coding applications in assembly language.

The hardware labs cover the onboard peripherals of the target microcontroller such as the timer subsystem, I^2C interface, and analog-to-digital converter, as well as off-chip interfacing to devices such as a serial EEPROM, an I^2C digital-to-analog converter, and an H-bridge/DC motor. The hardware labs are built from a parts kit ($65) purchased by the student, with wiring done on a protoboard. Figure 2 shows a protoboard with the first hardware lab connections completed.

We use the parts kit approach because if a student is handed a development board with the processor and other devices on a PCB, then many tend to view it as a monolithic unit and want to use that same PCB when they need a microcontroller for the capstone course. Building a system from a

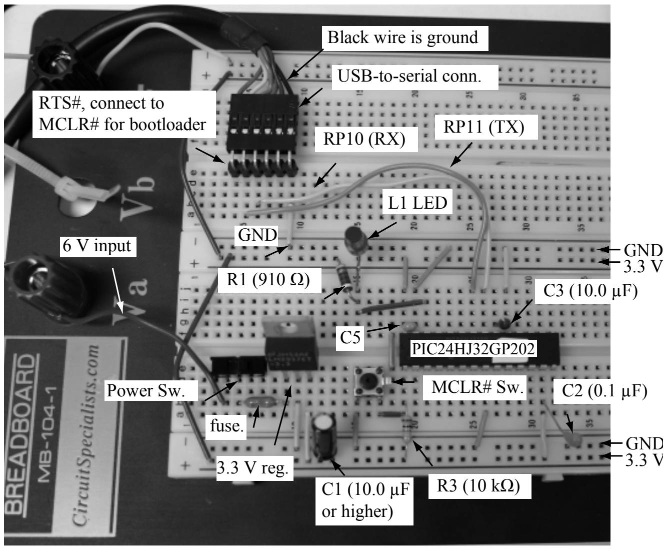

Figure 2. Wiring required for the first hardware lab.

bag of parts can be a painful but educational experience, but it only has to be done once. After that experience, we have found that students then understand the individual components of a microcontroller system, which instills the required confidence for building a custom PCB when they need it for the capstone experience.

This course/lab combination has undergone several transitions and modifications since Fall 2003, moving from instruction focused on an 8-bit microcontroller [10] to more recent efforts centered on a modern 16-bit microcontroller [1].

We considered other approaches, especially the TekBots [11] approach at Oregon State since that has the hands-on, early microcontroller introduction that we wanted. However, the robot platform is fairly expensive as an initial purchase, and we were not going to integrate it across several courses as was done at Oregon State. We needed a single course solution, and one that was flexible. The parts kit approach allows us to easily make changes, both in parts for individual experiments, and also in the core processor choice. The parts kit also serves as a starting kit when students need a microcontroller for the capstone course. However, during the two-semester capstone course, many teams end up choosing a different processor, most commonly one of the same general family but with more I/O pins or a different set of on-chip peripherals.

When first designing the course, several processor families were examined. We finally chose Microchip because of their traditional support of the hobbyist market with availability of DIP packaging, C compiler support with student versions, inexpensive in-circuit programming options, and a strong university program. In 2007 [12], the rankings for microcontroller sales in the US were (1) Freescale (32%), (2) Renesas (14%), (3) Microchip (11%), (4) TI (8%), and (5) NEC (8%). Worldwide sales ranking for the same period were 1) Renesas (21%), (2) Freescale (12%), (3) NEC (10%), (4) Infineon (6%), and (5) Microchip (6%). These rankings are used to show that using Microchip processors are mainstream in terms of industrial acceptance, and have features that are valued by that community. From a pedagogical viewpoint, the microcon-

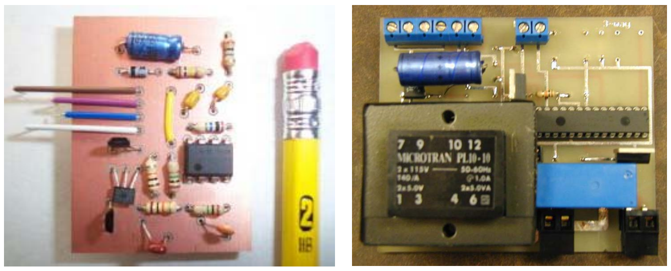

Figure 3. Two Photos of Complete Projects, Capstone Course, Fall 2002 (left), Fall 2001 (right)

troller choice is not that important as long as the microcontroller is representative of the general microcontroller population.

This type of course revision of rethinking the placement and content of microprocessor courses is hardly unique. Reference [13] describes a course change at the University of Alabama, Tuscaloosa that replaced an x86-approach in their introductory microprocessor course with an embedded approach using a Freescale MC9S12DP256B microcontroller. Given the increasing capabilities of modern microcontrollers and their ubiquitous role in digital systems, it is a natural evolution.

IV. Results

Since the curriculum change, we have seen a measurable improvement in the complexity and quality of projects in our capstone course. Figure 3 shows photographs of two completed capstone projects created before the curriculum changes percolated to the capstone course. The left photo, taken from a Fall 2002 capston design project, is an electromagnetic field detection circuit with audible buzzer ("SafetyPin") that is built from simple discretes and ICs. The photo on the right is from a 'smart light' controller for a second semester capstone design team advised by the PI of this proposal from Spring '01 to Fall '01. The team was bootstrapped to minimal competency with the PIC16F873 microcontroller during weekly advising sessions. The microcontroller was programmed in C, but the overall design and code complexity ended up being equivalent to approximately one lab exercise in the current microcontroller course. This frustrating experience (among many) was one of the drivers for the microprocessor course revision. Figure 1 shows a completed project from the Spring 2008 capstone course (USB scope with two 60 MB/s capture channels, and one function generator channel). Complexity differences between the designs in Figure 1 and Figure 3 are obvious.

For another comparison, of the eleven EE Capstone projects for Spring '02-Fall '02, three did not include a microcontroller component. Of the remaining projects, about half were programmed in assembly language, with a simplistic application. Some teams from the Spring' 02-Fall '02 did have team members that had taken the elective follow-on microcontroller course and the projects for those teams were more sophisticated. By contrast, of the eleven capstone projects for Fall '07-Spr '08, all had microcontrollers, with applications written in C, with applications complexity varying from medium to high.

Another measure of progress is the recent performance of the MSU IEEE Southeastern Conference (SECON) student hardware competition team (the hardware competition is this team's design project for the capstone course). The hardware competition typically involves autonomous robots performing some task on a small playing field. Since 2005, after the course changes from the curriculum change had time to percolate to the senior capstone course, the team has placed in the top three teams several times (1st: 2005 and 2007; 2nd: 2008, 3rd: 2006). Each time the competition has been against 20+ teams (and usually 30+). The SECON success cannot be attributed fully to the microprocessor course change, but we feel that the course change has made a substantial contribution. We had experienced some previous successes in the SECON hardware competition -- winning it in 2003, and usually finishing in the top 10, but not with the consistency and high achievement recently demonstrated.

V. Conclusions and Future Work

The ever-growing complexity of microcontrollers challenges the educator while also providing unique abilities to equip students with the ability to create large, complex systems. The evolution of the approach developed produced a textbook which includes a hands-on laboratory experience. Quantitative results show a significant improvement in the quality and complexity of the resulting capstone design projects, validating this approach.

References

[1] J. Peatman, *Embedded Design with the PIC18F452*, ISBN: 0-13046-213-6, Prentice Hall, August 2002, 420 pp.

[2] R. Tocci, F. Ambrosio, *Microprocessors and Microcomputers: Hardware and Software, 6/E*, ISBN: 0-13060-904-8, March 2002, 612 pp.

[3] R. Reese, J. W. Bruce, and B. A. Jones, "Microcontrollers: From Assembly Language to C Using the PIC24 Family," Cengage Learning, Dec. 2008, 838 pp.

[4] R. Barnettt, S. Cox, and L O'Cull, *Embedded C Programming And The Atmel AVR*, CENGAGE Delmar Learning, June 2006, 560 pp.

[5] Han-Way Huang, *PIC Microcontroller: An Introduction to Software & Hardware Interfacing*, CENGAGE Delmar Learning, July 2004, 816 pp.

[6] Texas Instruments, *CC2480: Z-Acell 2.4 GHz Zigbee® Processor*, CC280 Datasheet SWRS074A, 43 pp. Available online at http://focus.ti.com/lit/ds/symlink/cc2480a1.pdf.

[7] DPAC Technologies, *Airborne™ Wireless LAN Node Module Data Book*, 39L3702-01 Rev. D 10/25/2004, www.dpactech.com, 128 pp.

[8] Future Technology Devices International, *Vinculum VNC1L Embedded USB Host Controller IC*, Datasheet version 0.97, 2007, 17pp. Available online at http://www.vinculum.com/documents/datasheets/DS_VNC1L-1A.pdf.

[9] R. Reese, "Embedded System Emphasis in an Introductory Microprocessor Course", 2005 ASEE Annual Conference & Exposition, June 12-15, 2005, Portland, Oregon, poster session.

[10] R. Reese, *Microprocessors: From Assembly to C with the PIC18xx2*, ISBN: 1-58450-378-5, Thomson Delmar Publishing, July 2005, 664 pp.

[11] R. Traylor, D. Heer, and T. Fiez. "Using an Integrated Platform for Learning™ to Reinvent Engineering Education". IEEE Transactions on Education, v46, No 4, 2003, pp 409-419.

[12] M LaPedus, "Renesas seeks control of controller arena", EETimes, March 14, 2008. Available online at http://www.eetimes.com/showArticle.jhtml?articleID=207200324.

[13] W. A. Stapleton, "Microcomputer Fundamentals for Embedded Systems Education", 36th ASEE/IEEE Frontiers in Education Conference, Oct. 28-31, 2006, Session M2D.