An Embedded Systems Design Course Sequence

James A. Ochoa, George B. Wright
Texas A&M University

I. Introduction.

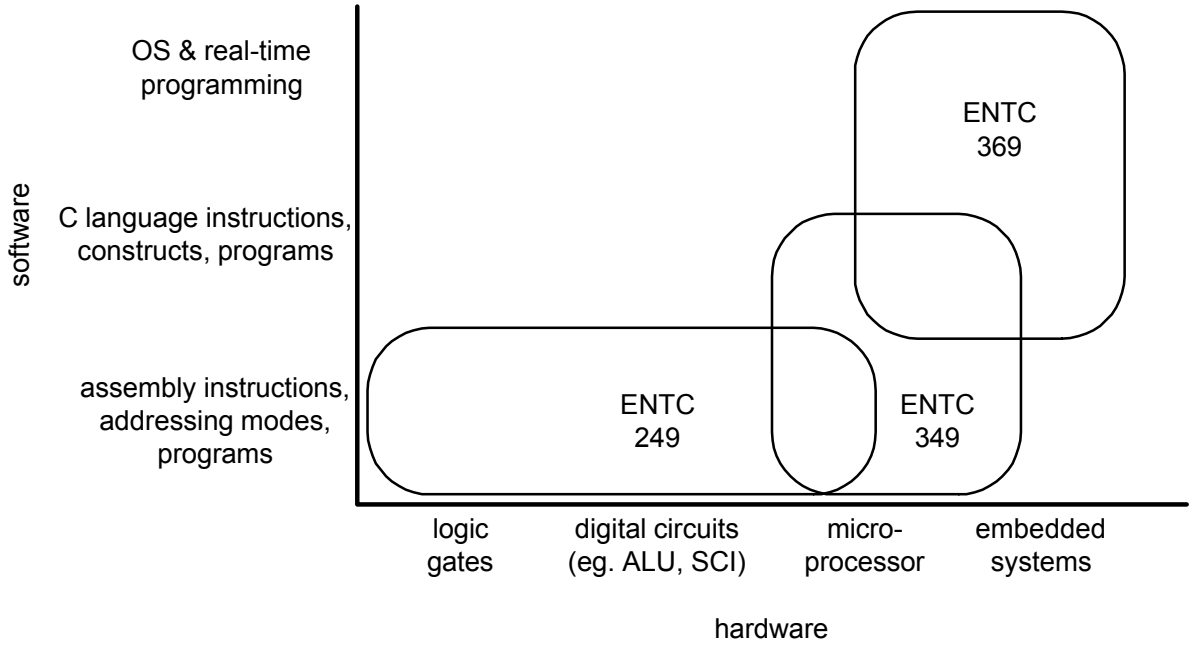
The number and variety of handheld computing devices is rapidly growing and it seems as if the world is becoming more dependent on microcontrollers every day. After all, these computers are part of virtually every electronic device, ranging from state-of-the-art instruments to home appliances. They are strongly integrated into our daily routines as well. It is not uncommon for professionals to carry a cell phone, personal data assistant and pager. Not surprisingly then, the impact of microcontrollers has also been seen in engineering education.

In response to the changing electronics world, the Electronics and Telecommunication Engineering Technology Programs at Texas A&M University have created an embedded systems design course sequence over the last decade that focuses on the area of microcontroller-based design. While traditional electronics engineering technology curricula typically include a single microprocessor course, the embedded systems design course sequence is made up of three courses: Advanced Digital Design, Microcontroller Systems, and Software Systems Technology. The course topics, which are aimed at developing a strong working knowledge of embedded system design fundamentals, range from design of low-level processor architectural elements up to system-level integration issues unique to the embedded world. The figure below illustrates the relationship of the three courses in terms of hardware and software emphasis.

The purpose of this paper is to provide a general overview of the three courses and to discuss changes that are currently being considered. The paper will present a brief history of the evolution of these courses as well as factors that have led to their change. As an example, in an applied design environment, a strong collection of support tools is crucial. The particular tool sets previously used in the three courses will be described and various considerations associated with their selection, acquisition, support and maintenance will be pointed out. As the embedded systems design courses continue to evolve, they will be largely influenced by existing development software and hardware. It is anticipated that a follow on paper will report the outcomes of recent industrial partnerships established to provide strategic direction for course enhancements.

II. Advanced Digital Design

The Advanced Digital Design course, ENTC 249, is a second semester sophomore course that builds significantly on the typical introduction to digital circuits (ENTC 219). The structure of the advanced course begins by providing students with simple logic design problems (e.g., counters) and steadily increasing the complexity of the problems. By the end of the semester, students design and implement an eight-bit microcontroller that is largely based on the Motorola 6805. This particular microcontroller architecture was chosen because of its direct relationship



to the next course in the sequence, Microcontroller Systems, which uses the Motorola 68332. These two microcontrollers have similar architectures, instruction sets, and addressing modes.

The 6805 microcontroller design is a semester long project. A modular design approach is followed. Students design each microcontroller subsystem separately and then, at the end of the semester, the individual components are integrated. The subsystem designs include: a bidirectional parallel port, a Universal Asynchronous Receiver Transmitter (UART), an addressing unit, a data unit, an arithmetic logic unit (ALU), and a central processing unit (CPU) controller. One at a time, each subsystem is designed, implemented and tested in a simulation environment, and then implemented in a programmable logic device (PLD). Once all the subsystems are completed, the students integrate them into a single design.

Students use the Altera Max+Plus II design environment for simulation and hardware implementation. The designs are developed both in a schematic entry as well as hardware description language. In particular, students begin using the Verilog HDL early in the semester because it allows their designs to be developed at a behavioral level. As a result, students are able to develop a working subsystem quickly (e.g., in one to two weeks per subsystem, except for the central processor). An additional benefit to gaining experience with Verilog HDL is that it provides, in most cases, a first structured programming experience, something that will be strongly leveraged in the following two courses. Students also learn assembly language instructions, addressing modes, and simple programming. These designs and assembly programs are then "burned" into PLDs and tested in hardware. As one element of the testing process, students generate programs that are run on their microcontroller.

The experience and skills gained in ENTC 249 provide a strong foundation for pursuing embedded system interfacing and design. In the follow on course, Microcontroller Systems (ENTC 349), students work with an off-the-shelf microcontroller to develop systems – requiring both software and hardware development and integration.

III. Microcontroller Systems

The Microcontroller Systems course uses the Motorola MC68332 microcontroller to introduce students to embedded programming and interfacing. While programming is an integral part of the course, the primary focus is on microcontroller interfacing. As such, only a simple assembler and compiler are used. The course is organized so that the first two-thirds of the semester is dedicated to studying the individual microcontroller modules, while the final third is focused on developing a complete embedded system.

The course begins by focusing on the MC68332 architecture. This particular microcontroller is comprised of four modules including a CPU, a System Integration Module (SIM), a Timed Processing Unit (TPU), and a Queued Serial Module (QSM). The course progresses throughout the semester by covering each module individually. In lecture, each module is presented by describing its general purpose, its underlying operation, and the configurations necessary to operate the module. Each lecture topic is accompanied by a laboratory assignment that utilizes the respective module to solve a stated problem. In each case, a program is written (either in assembly or C) and the module is interfaced to hardware (except for the CPU). In addition to studying the microcontroller architecture, the course covers a number of key topics fundamental to embedded systems. These include interrupts and interrupt service routines, serial communication (both synchronous and asynchronous), analog-to-digital signal conversion, user interfaces, etc. Software *design* is also addressed. The course emphasizes the design of software from a top down approach and students are required to develop a progression of software design representations based on a problem statement.

The CPU is taught from the programmers' model approach. In particular, the processor's addressing modes and assembly instructions are covered in detail. Students spend three weeks learning how to write medium complexity programs that utilize a broad set of instructions and addressing modes. Next, students gain experience with software interrupts and interrupt handling. In this process, students develop real-time programs that will be utilized later in the semester. The remainder of the semester focuses heavily on hardware interfacing. The first hardware area is asynchronous serial communications. Students study RS-232 communications, including communicating with a PC-based terminal. They develop an interrupt-driven communication driver, which will also be used later in the semester. The next hardware area involves using the microcontroller's TPU to implement both another serial communications port and a multiple-channel pulse width modulation (MCPWM) interface. The latter is used to establish an open-loop speed control system for a dc motor. Building on this, the microcontroller's serial peripheral interface (SPI), which is a subcomponent of the QSM, is used to establish a communication link to an analog-to-digital converter. Students use the converter to provide a feedback mechanism for measuring the dc motor's actual speed. In the final four weeks of the semester, students integrate their knowledge of the individual features of the microcontroller to create a closed-loop motor speed control system. The problem statement is given in terms of features and performance requirements. This system is comprised of multiple software threads including: three software interrupts and service routines, an application user interface, a debugging interface, and control algorithm. In addition, it includes hardware for controlling the dc motor through an H-bridge and measuring the actual motor speed through a series of signal conditioning circuits. Three hardware-interrupts and their respective services routines are also created. Working in small teams, students develop individualized approaches to solving the problem.

IV. Software Systems Technology

The Software Systems Technology (SST) course (ENTC 369) was first offered in the mid 1980's. At this time the faculty recognized that the impact of software on the overall design of electronics systems was growing. Microcontrollers were taking an ever-expanding role in electronics systems. Assembly language was commonly used to program microcontrollers as well larger computer systems. However, the advantages of high level programming, such as the C programming language, were becoming apparent. Over time, microcontroller capability grew. Eight-bit devices were replaced with sixteen-bit devices and sixteen-bit expanded to thirty-two bits. As the power of the microcontrollers expanded the development tools and the capability of the tools grew.

In the mid 1980's the SST course was based on eight-bit Motorola microcontrollers. These devices only required the most basic development tools. Typically, assemblers and text editors were the required tools. Today the SST course is taught using the Motorola 68332 microcontroller. This thirty-two bit device has several on-chip peripherals that give the student significant resources to utilize in the laboratory. Today, the software tools include the Introl C Cross Compiler, integrated development environment and associated tools. In addition, the MicroC/II Real-time Kernel is studied and used as an example real-time kernel. It has proven to be an excellent tool for teaching real-time concepts as well as supporting of project development.

The SST lecture begins with a brief review of the C Programming Language and moves quickly through a review of the Motorola 68332. The primary focus of the course is on the development of C programs for the Motorola 68332 using Introl C. The course is project oriented. That is, the laboratory work focuses on the development of an embedded system. The lecture is designed to support that development. One example of a laboratory project is a calculator. Each student is required to develop an operational calculator that meets certain functional specifications. In general, the requirements include keypad input as well as display output.

The SST course has been taught for many years. It has experienced one major upgrade when eight bit microcontrollers were replaced with thirty-two bit microcontrollers. It may be time to consider another major upgrade. Of course this type of change is costly for the university for several reasons. First, there is the direct cost of purchasing new development systems and support tools for software development. In addition, there are additional costs associated with infrastructure. These cost are related to the training of faculty, graduate students and technicians that will use and maintain the new systems. Of course, the cost of moving to new technologies must be balanced against the cost of not moving.

V. Conclusion

The embedded systems design sequence has been very successful in providing a comprehensive education in the area. Students gain a working knowledge of embedded design practices that is immediately applicable upon graduation. Given today's increased utilization of embedded processors, the skills developed by the courses described in this paper will provide students with strong marketability.

Microcontroller technology continues to move forward. As these systems continue to grow in complexity, engineering as well as engineering technology faculty should consider the following factors:

1) What computer architectures should be taught/used in labs: CISC, RISC, both?
2) What programming languages should be taught: C, assembly, other?
3) What manufactures should be considered mainstream: Motorola, Intel, Texas Instruments, other?
4) What software tools are important: compilers, assemblers, simulators, other?
5) What interfacing and peripheral technologies will be significant in the near future: USB, IEEE 802.11, Bluetooth, RIO?

While much of this decision process is still underway, the Electronics and Telecommunications Programs have recently established strategic partnerships with Motorola, Texas Instruments, and Metrowerks. These key relationships form a basis of support to facilitate insight into the current and future directions of embedded system technologies. Of particular interest is the integration of wireless data technologies into the courses discussed in this paper. To this end, two additional industry partners have recently been identified and contacted. It is clear to the authors that the next generation of microcontrollers to be used in the courses will be based on an RISC architecture. However, much work is still needed to identify a particular device that provides the key capabilities now being defined. It is anticipated that more details will be available in the near future and will be presented in a follow paper.

VI. References

1. T. Caceras, Z. Combs, J. Ochoa, "An Advanced Microcontroller Systems Course for Upper-Level Undergraduate Curriculum," 2002 American Society for Engineering Education Annual Conference, Proceedings, Montreal, Canada, June 16-19, 2002.
2. J.R. Porter, R. Fink, J. Ochoa, "Enhancing Core Curriculum Concepts Through Industry Collaborations," 2001 American Society for Engineering Education Conference for Industry and Education Collaboration, Proceedings, San Diego, CA, Jan. 30, 2001.
3. R. Fink, J. Porter, J. Ochoa, R. Alexander, "Synergy of Applied Research and Education in Engineering Technology," 2001 American Society for Engineering Education Annual Conference, Proceedings, Albuquerque, NM, June 2001.
4. J. Ochoa, M. Landrum, "An Approach to Advanced Digital Design for Undergraduate Students," 1999 American Society for Engineering Education Gulf Southwest Region Annual Conference, Proceedings, Dallas, Texas, March 1999.

VII. Bibliography

JAMES A. OCHOA

James Ochoa received a BS degree in electrical engineering at Texas A&M University – Kingsville in 1990 and a Ph.D. in electrical engineering at Texas A&M University in 1998. After completing his Ph.D., he joined the faculty in the Department of Engineering Technology at Texas A&M University. His research activities include digital electronics and circuit testing, system-on-a-programmable-chip, and micro-controller based control systems. Dr. Ochoa can be contacted via email at j-ochoa@tamu.edu

GEORGE B. WRIGHT

George Wright received a BS degree in electronics engineering technology at the University of Houston in 1971 and a M.Ed. in 1975. He currently teaches in the Engineering Technology Program at Texas A&M University. His recent research work has focused on the development of in-vehicle networks for law enforcement vehicles. His work was sponsored by the U.S. Department of Transportation. He holds two patents for his work in this area.