Teaching Embedded Systems Using Multiple Microcontrollers

C. Richard G. Helps, David P. Phillips
Electronics Engineering Technology, Brigham Young University

Abstract

Embedded control systems and in particular microcontrollers are used in virtually every electronic system. It is essential that EET students be conversant with this technology. Students need to have a clear understanding of the diversity of embedded systems. They also need to be familiar with a range of development tools, operating systems and languages.

The characteristics of embedded systems add specific challenges to their development. They necessarily involve both hardware and software, and the software often has real-time constraints. Their development and debugging therefore require structured design techniques and good understanding of software design principles. Software is frequently developed in C or assembly code, often without the benefit of sophisticated or even standardized development tools. Data books for these devices are often obscure and good reference books are scarce.

A microcontroller development system has been created for EET students at BYU. The development system forms the "textbook" for a class in embedded real-time systems. Students combine the theory of real-time systems with the practice of microcontroller systems development. They each develop a data acquisition unit using most of the features available in microcontrollers. The unit interfaces to sensors, actuators, LCD displays and serial ports using different microcontroller architectures. The final data acquisition system can then be used for later classes or projects or reprogrammed for other applications.

1. Embedded Microcontrollers are an Important Topic in EET programs

Embedded controllers are found in many modern products ranging through consumer electronics, cars, industrial control, medical systems and communication. They serve as the primary intelligence for many products or as simple intelligent interfaces between systems. Microcontrollers are even replacing simple mechanical switches in many systems. A recent survey[1] indicated that for every 75 million desktop computers sold each year approximately 2.3 billion microcontrollers are sold. Another survey estimates that as many as 11 billion microcontrollers of all types were sold in 1997[2]. In the automobile market one estimate[3] indicates that as many as 40 million cars are produced each year that typically include four major microprocessors and dozens of low-end chips.

Mechanical engineering departments have also recognized the importance of this technology. In recent years many of them have started “Mechatronics” programs using microcontrollers as the intelligent element in the system.

2. Software Requirements for Embedded Systems

The software designer for an embedded system must consider all the usual constraints, such as structured design and algorithmic and logical correctness. In addition to these there are two other significant features of embedded system software. The first is the hardware interfacing inherent to every embedded system. The second is the real-time nature of embedded systems.

The peripherals and interface hardware of a microcontroller are tightly coupled to the CPU and form a logical extension to the software of the system. Hardware configuration and timing must be clearly understood for software to function successfully. Furthermore, when the hardware malfunctions it can sometimes appear to be a software error. Both of these problems are illustrated by a recent example in a student project. Students were using a microcontroller tied to a set of latches to multiplex and buffer outgoing data. Several students assumed that the latches were the standard D type latches and wrote software accordingly. The latches were in fact transparent: if they were turned on then all subsequent data was passed through until they were turned off. The problem manifested itself as data appearing on multiple outputs which were thought to be inactive. This was a software error based on a misunderstanding of the hardware. Tracing this problem was made more difficult since the latches were not performing within design specifications and were occasionally triggered by narrow noise spikes on the power lines. This was a hardware fault that also appeared to be a software logic problem.

Since hardware is sometimes misunderstood and faulty hardware problems such as those cited do occur, there is a tendency for software writers to blame many of their problems on hardware. This is the classic “Not Invented Here” syndrome. It is very important that students are involved in both the hardware and software design so they understand the hardware and can, if necessary, debug it. This problem is addressed by actively promoting a structured design approach for both hardware and software in the course.

The other important consideration in embedded system software is its real-time aspect. Interrupts and multitasking are the most common ways of implementing real-time software and are also some of the more difficult types of software to write and debug. This problem is aggravated by the scarcity of operating systems in microcontroller systems and by the prevalence of primitive development environments. Students need to understand both the principles and the practice of real-time systems with particular attention to minimalist implementations.

Embedded systems without a screen or keyboard are also much more difficult to test. Software design must include provisions for testing. Systems must either indicate their status by operating LEDs or some other external signal or they must be tested using a simulator or emulator. In many cases a combination of these techniques will be used.

3. Embedded System Development

Development must be based on a foundation of structured design. Embedded processors do not have keyboards and screens suitable for developing software. Software must be written on a different machine, the development workstation, which is usually a desktop computer. The

development workstation needs to perform the following functions: editor, assembler or compiler, simulator, programmer and emulator. Many chip manufacturers provide development packages which include most of these features. These development packages are often complete integrated development environments. A complete package with the features listed can greatly aid development but it does not enforce or even encourage structured design. That discipline must come from the programmer.

4. Need for Diversity in Microcontroller Courses

In a rapidly changing engineering environment the half-life of an engineering education is about five years[4]. Students need to be life-long learners. Students not only need a wide range of software, hardware and design skills[5] they also need to be specifically educated to recognize the range of capabilities available in different microcontroller architectures. Once students understand the principles behind microcontrollers and embedded systems and have some experience with implementing them, they can evaluate the suitability of new architectures and can quickly learn how to apply them. It is therefore essential that students be exposed to a range of different systems. They must not be narrowly trained to be able to operate a single system but must be exposed to the whole field of embedded systems so they can not only appreciate the effectiveness of a particular microcontroller for solving embedded system problems but be able to choose the best microcontroller for the job.

Embedded systems encompass not only microcontrollers but also embedded microprocessor systems such as the well-known PC104 family. There are also specialized systems such as the DSP microprocessors from TI, Motorola, Analog Devices and others that often end up in embedded systems. A dividing line can be drawn between microcontrollers and embedded microprocessors but need not be. Students need to be competent in working with desktop-based systems. Those skills transfer well to embedded systems such as the PC104 with a 386 CPU, where memory is plentiful and operating systems are similar to desktop ones. Students also need to be conversant with much smaller systems where RAM is measured in tens of bytes and EPROM space (off-line storage) is measured in just a few kilobytes. The techniques used for both systems are based on fundamental principles of multitasking, real-time systems and off-line programming, although scaled down when applied to the smaller systems. This is another reason why education must be principle-based and not only practice based.

Eight bit microcontrollers are the most commonly available. They have almost completely replaced four bit systems since there is now no significant price difference[2]. 16 and 32 bit systems are very similar to the 8 bit systems but are more expensive and are less well supported. We see very little educational benefit in concentrating on 16 and 32 bit systems although they are discussed in class.

Other issues with microcontroller development include problems of reliability, development time, and a lack of operating systems.

5. BYU EET Microcontroller Development System

In order to achieve the above objectives we have developed a microcontroller development system for students to explore several of the microcontroller systems available. Students have all studied topics in previous classes so that they are well grounded in several areas

- Basic digital systems
- Interfacing digital systems
- Structured Program design
- Coding and debugging (Assembly, C, Pascal or Java)

These classes require 14 semester credit hours spread over the first to third years of study. The challenge that the instructor faces when designing a course of this type is that one wishes to give the students the widest experience possible so that they have a thorough understanding of the field. They must also have sufficient depth of exposure so that they do understand the topics and are competent to design and solve problems' requiring embedded system control, and not merely survey the field. They also need to be aware of the tools that are available to facilitate development. Time-to-market is one of the most crucial factors in deciding the financial success of a new project and students need to be able to not only develop effective solutions to problems but also to develop them quickly. Added to this challenge is the constraint of a limited number of credit hours that can be dedicated to a single topic.

We selected two significantly different microcontrollers to cover in depth in a single course. Other processor technologies are covered in discussions in that course and are also covered in some depth in other courses. We selected the 8051 architecture to represent a fairly conventional CISC system. There are many other suitable chips which represent what we consider to be conventional architectures, some of the most notable being the Motorola 6811/6805 series, the Mitsubishi MELPS family and the NEC K series. We chose the 8051 largely because of the wide support available from many suppliers and the easy availability of EEPROM parts that simplified programming in the first part of the course. The second architecture we chose was the Microchip PIC architecture. This RISC chip is very different in its programming. The complete instruction set consists of only 35 instructions and the system has only one general-purpose register. We felt that students exposed to these two systems would be able to handle and adapt to a wide variety of other processors when the need arises.

Once we had chosen the systems to use, we found that manufacturers such as PIC, Phillips, Atmel and Motorola were very generous in supplying parts for us to test and for the students to use.

Structured programming skills are emphasized throughout the course. Microcontroller systems usually have very small memory sizes and very little or no memory management. While almost all desktop systems use one of the few major operating systems (UNIX, Windows, Mac), embedded microcontroller systems either have no operating system or use one of the many proprietary operating systems. In general memory is globally shared. Students need to be disciplined in program design, coding and debugging to ensure success. This discipline, taught in earlier programming classes, takes on new meaning in the embedded environment. Students in the class commented that they design with much smaller, more easily tested modules for microcontrollers than they do for normal C or assembly for desktops.

There is no formal textbook for the class. Students are encouraged to use the same data books and Internet resources that they will have access to as practicing professionals. Class lectures and handout notes cover theoretical material. Students are also provided with sample code to help them learn new instruction sets, assembly syntax, and programming techniques faster. In many cases this sample code is also from publicly available sources such as manufacturers' application notes. The Internet provides data sheets, example code, application notes, development tools and other materials. These are freely available from nearly every manufacturer and from other interested parties.

We also felt that students would be more motivated if the class and development system had a clear and useful goal. Therefore the lab exercises were all steps in developing a serial port controlled, multitasking, real-time data acquisition system with onboard LCD display. In teaching some real-time and higher-current interfacing techniques a stepper motor controller was also developed.

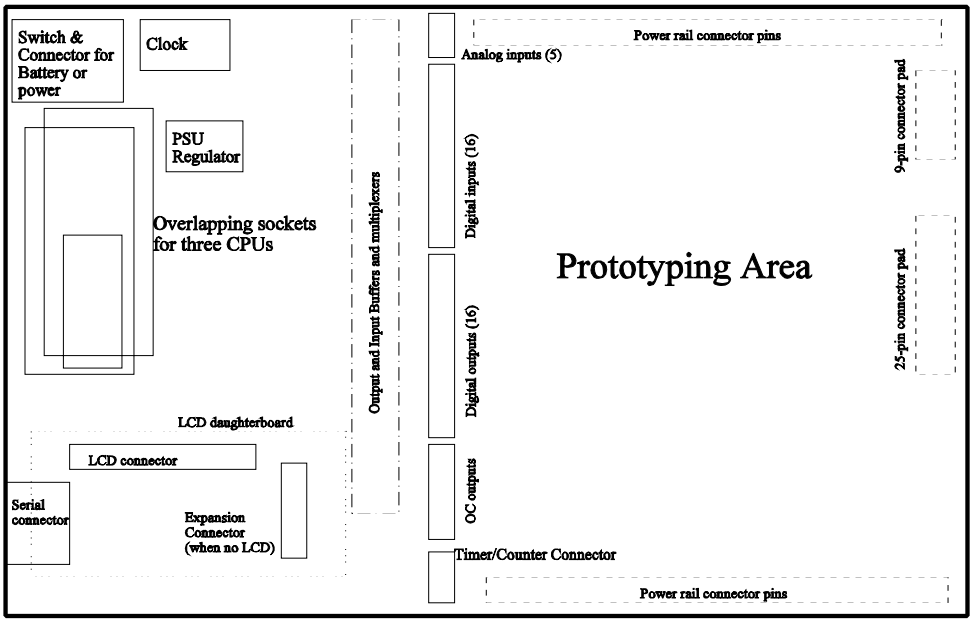

Figure 1 Development system PCB layout showing principal features

The development hardware consists of a double-sided PCB that is socketed to accept several microcontroller (MCU) chips. The MCU sockets are arranged so that only one MCU can be plugged in at a time to prevent conflict with multiple MCUs trying to control the buffer circuits. Some additional features are included such as buffer chips to expand the I/O ports, a serial connector, a power supply, a reset switch and prototyping area. The buffer chips are connected to the pins of one of the MCU sockets but wire-wrap pins make them available for general use. Figure 1 shows an outline of the development board.

The course begins by having the students develop a simple basic functioning processor. They take an 8051 MCU chip and apply power, a clock oscillator and a reset switch to it. They then develop a simple program to read a single switch as an input and turn on a single LED. This is the microcontroller equivalent of the traditional C program known as "Hello World." A very simple "Hello World" program consists of the instructions

```
ReadSwitchWriteLed:
        MOV   R0,P1             ; switch on P1
        MOV   P0,R0             ; LED on P0
        JMP   ReadSwitchWriteLED
```

This process teaches them to consider hardware issues such as impedances and current limits. It also teaches them all the necessary mechanics of software writing, cross assembling, downloading and running. It also introduces the problems of debugging embedded systems without the benefits of integrated development environments. Students are given PIC and 8051 sample programs. These sample programs demonstrate assembler syntax, program documentation, and contain functional assembly code.

After this, students move through a progression of lab exercises exploring the various peripherals on the MCU chip and dealing with hardware interfacing and buffering. Interface chips are provided as part of the class development system but students are still required to research their characteristics and design the necessary software to take advantage of them.

About halfway through the semester the students switch over from the 8051 to the PIC16C74 for lab exercises and work, at an accelerated pace, through similar exercises with the PIC MCU. They then develop the data acquisition system. The data acquisition system accepts a simple set of ASCII commands over the serial port, monitors and controls ports appropriately, and sends results to the onboard LCD display and to the serial port. Although the data acquisition functions implemented are minimal, the structure is in place and since the students have developed the system themselves, they can easily extend or modify the system for any particular purpose.

In parallel with the lab exercises, class discussions explore topics such as multitasking, real-time definitions, practical real time strategies, other microprocessor architectures, and general embedded system software design. This is intended to give them the theoretical background and broadening which will enable them to evaluate the lab experiences with clear understanding.

6. Response to the Development System and Course

Student response has been very positive as measured by the standard anonymous course evaluation system used on campus. The course has been taught by each of the authors separately and was rated by the students at 6.1 on a 1 - 7 scale. This compares to an average for the college of about 5.3 and to an average for these instructors of about 5.5. Student comments indicate that they found the course challenging but that several thought it was one of the best courses taken. Microcontrollers are widely used by the students in later projects and students are

very willing to choose different types of microcontrollers for projects and even to combine different MCUs in a single project to take advantage of different features. This course has been evolving over the last few years and has only been taught a few times. Early informal responses from alumni indicate that the skills transfer well to the professional workplace.

7. Conclusions

The use of microcontrollers and embedded systems is growing rapidly in many different product categories. The microcontroller is characterized by diversity in type and rapid change. This is an excellent application area for engineering technology graduates. Diversity in background and in microcontrollers are both necessary to work in the field professionally.

Students with a principle-based multiple architecture background will be best equipped to take advantage of the many microcontrollers available and new ones being developed. Structured programming is essential in an environment characterized by hardware interfacing, real-time constraints, simple operating systems and minimal development interfaces.

BYU has developed a microcontroller development system and accompanying senior-level course that accomplishes most of the goals for teaching embedded system principles. The students develop working prototype embedded system devices using professional resources. They study and apply diverse microcontroller architectures. They gain understanding of the need for reliability, on-time delivery and real-time hardware/software design. This approach is very successful as measured by student responses to the course, and the ease with which they apply the system to other projects in the department.

Bibliography

1. Peatman, John B., *Design with PIC Microcontrollers*, Prentice Hall 1998
2. Cole, Bernard, *Embedded Systems - Boundaries Fade with Architecture Changes*, Electronic Engineering Times, March 16, 1998, p89.
3. Werner, Loren, *Embedded Operating Systems Face Greater Productivity Demands*, Electronic Design, Oct 1 1998, p51-60
4. Pfile, Richard E. & Conrad, William R., *Bring Realism Into the Classroom Through Your Consulting*, Proceedings, ASEE Annual Conference, Seattle,1998
5. Cheng, Betty H. C., Rover, Diane T. & Mutka, Matt W. *A Multi-Pronged Approach to Bringing Embedded Systems into Undergraduate Education*. Proceedings, ASEE Annual Conference, Seattle,1998

C. RICHARD G. HELPS

Richard Helps is the Program Chair of the Electronics Engineering Technology program at BYU. He spent ten years in industry as a control systems design engineer. He completed BS and MS degrees at the U of the Witwatersrand, South Africa and a further graduate degree at the U of Utah. His primary interests are in control systems, particularly embedded control systems, combined with artificial intelligence techniques such as neural networks and fuzzy logic.

DAVID P. PHILLIPS

David P. Phillips is a part-time faculty member of the Electronics Engineering Technology program at Brigham

Young University. He has twenty-one years of design experience (hardware and software) with microprocessor and microcontroller-based systems. He is now employed at Intelogis, Inc. where he is part of the design team working on Intelogis Powerline Networking products. He has worked at Los Alamos Scientific Labs, Brigham Young University, Purdue University, and Novell Inc. David received an M.S. degree from the University of New Mexico/Los Alamos and a B.S. degree from Brigham Young University.