Converting a Microcontroller Lab From The Freescale S12 to the Atmel ATmega32 Processor

Christopher R. Carroll
University of Minnesota Duluth
ccarroll@d.umn.edu

Abstract

During the summer of 2013, the laboratory supporting the microcontroller course at the University of Minnesota Duluth was completely re-implemented. For the last several years, the processor that has been used was the Freescale S12, a popular 16-bit microcontroller with a long ancestral history[1]. The recent popularity of the Atmel AVR series of microcontrollers, as used in the Arduino microcomputers, for example, has prompted a change in the lab to use Atmel's ATmega32 microcontroller, an 8-bit member of the AVR family of microcontrollers[2,3]. The new processor has a fundamentally different architecture than that used in the past, but the input/output resources available are much the same. This paper addresses issues that will be faced in the conversion when the course is taught with the new lab hardware for the first time in the Fall.

At the very fundamental level, the S12 and ATmega32 differ in architecture. The S12 is a Princeton architecture computer (single memory for both program and data), while the ATmega32 is a Harvard architecture computer (separate program and data memories). The S12 is clearly a CISC machine (Complex Instruction Set Computer) while the ATmega32 is clearly a RISC machine (Reduced Instruction Set Computer). These differences will affect how the microcontroller course is taught when it is offered in the Fall using this new lab. Fortunately, however, the collection of input/output devices in the AVR microcontrollers mimics closely what is found in the S12, so that many of the existing lab exercises will be used again with only minor tweaking.

This paper will discuss what has been done and what is planned for the updated microcontroller course. The course will be offered in the Fall, 2013, semester, using this new lab hardware for the first time.

Background

For more than five years, the microcontroller course in the Electrical Engineering department at the University of Minnesota Duluth has used the Freescale MC9S12DP256 microcontroller (S12, for short) as the foundation for lab exercises. This is a 2nd-year required course in the Electrical Engineering program. The prerequisite for this course is Digital Logic, where students learn fundamentals of digital circuit hardware design. This course is all about software, teaching students to program in assembly language, with the microcontroller as the vehicle.

Formerly, the microcontroller course has used Wytec's Dragon-12 board (*Figure 1*) to provide the user interface for the S12, including a speaker, 4-digit 7-segment multiplexed display, 2x16 character alphanumeric Liquid Crystal Display (LCD), serial interfaces, discrete LEDs and switches, and more. The S12 is a 16-bit microcontroller with a rich instruction set and a very complete collection of input/output devices built-in that together provide an excellent basis for a variety of lab assignments. The S12 is a current product, and has a long ancestry that includes the popular 68HC11 8-bit microcontroller, and even extends back to the original Motorola 6800 microprocessor. The S12 has served admirably as the foundation for the microcontroller course's lab experience for many years.

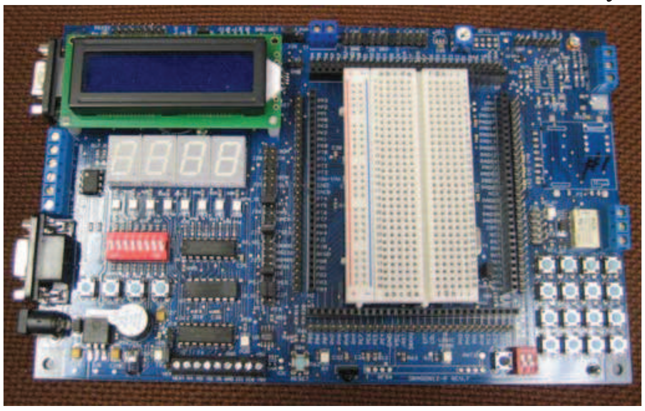

Figure 1. Wytec's Dragon-12 board

In recent years, a new family of microcontrollers, the AVR family from Atmel, has emerged, and has become popular due to its use in the convenient Arduino family of microcomputers. The Arduino microcomputers are easily incorporated into student senior projects, and are easily programmed in high-level languages such as C or Pascal. The AVR family includes a wide variety of processor capabilities, from simple components housed in 8-pin packages to powerful devices in 40-pin packages or larger. Because students have used Arduino microcomputers so often, there was considerable interest and pressure to update the microcontroller course, switching to an AVR processor to provide students with background knowledge for their project development.

Fortunately, a new development board for AVR processors is available, the EasyAVR version 7 from MikroElektronika (*Figure 2*). The EasyAVR board features the ATmega32 microcontroller from the AVR family, and includes many of the same resources as the Dragon board (speaker, 4-digit 7-segment multiplexed display, 2x16 character alphanumeric LCD, serial interfaces, discrete LEDs and switches, etc.) and also includes a USB port that eases connection to modern personal computers, and a 128x64 Graphic Liquid Crystal Display (GLCD) with touch-screen capability that adds an additional resource to support new lab exercises. This EasyAVR board includes programming hardware that allows easy configuration of any of the AVR-family microcontrollers that are packaged in DIP packages, making the board a great resource for students in senior project development.

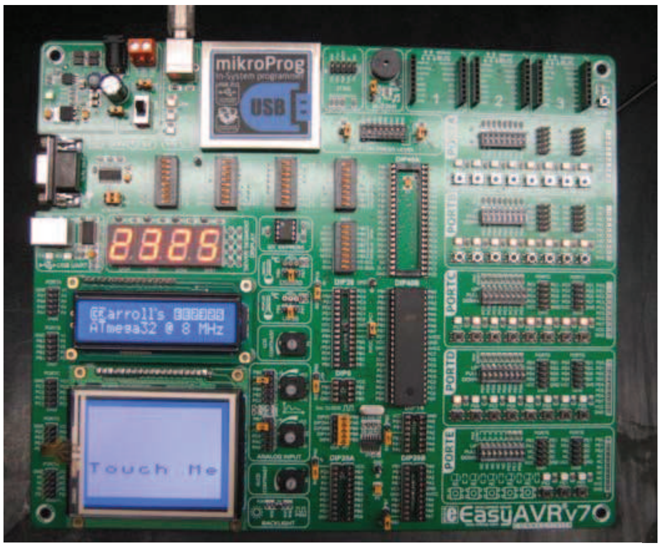

Figure 2. MikroElektronika's EasyAVR 7.0 board

Fundamental Differences

The S12 and ATmega32 microcontrollers are both current products from their respective manufacturers (Freescale and Atmel) and both support systems that can be embedded easily in projects. However, the two processors have several very fundamental differences that will affect how the microcontroller course is taught, and present some interesting instructional challenges.

First, the S12 is a 16-bit microcontroller, meaning that the fundamental size of operands that are manipulated in a single instruction is 16-bits. The ATmega32 is an 8-bit processor, which in some cases makes it less powerful from a computational viewpoint. However, the 8-bit nature of the ATmega32 actually will ease the most common difficulty faced by students, namely knowing whether registers in the processor are 8-bit or 16-bit registers, and knowing whether a particular instruction uses 8-bit or 16-bit operands. Confusion on this point has been the single most common difficulty faced by students when learning to use the S12 processor. The 8-bit nature of the ATmega32 should eliminate this source of confusion. Processing power is not an issue with the kinds of applications students face in the microcontroller course, so the 8-bit nature of the ATmega32 over the 16-bit S12 is actually a pedagogical advantage.

Second, the S12 is clearly a Complex Instruction Set Computer (CISC) device, whereas the ATmega32 is clearly a Reduced Instruction Set Computer (RISC) device. The debate among computer architects as to whether CISC or RISC designs lead to better performance has no clear winner. In some applications, CISC excels, whereas in other applications RISC wins. Although no processor is 100% CISC or RISC, the S12 and the ATmega32 display many of the characteristics of the two extremes. The S12's instruction set includes many exotic instructions meant for specific applications. Notable are the fuzzy logic instructions, and the table interpolation instructions, that perform complex calculations behind the scenes. By contrast, the ATmega32's instruction set includes only simple, straightforward instructions that each perform very limited tasks. CISC proponents argue that their philosophy is best because they can use their complex instruction set to accomplish some goal with only a few instructions, thus maximizing performance. RISC proponents admit that their philosophy will require more instructions to accomplish the same goal, but because RISC instructions are simple and streamlined, the clock rate can be boosted on a RISC processor so that the overall performance of a RISC solution, although longer, is still better. Who wins? It depends on the application. The CISC S12 processor includes just 4 general purpose registers (some 8-bit, some 16-bit) whereas the RISC ATmega32 processor includes 32 8-bit general purpose registers, another characteristic that often distinguishes CISC and RISC processors. From an instructional point of view, it will be easier to describe fully the instruction set of the ATmega32. However, the challenge is that student programs will be longer and harder to grade with the larger register set.

Third, the S12 is a Princeton architecture processor, whereas the ATmega32 is a Harvard architecture processor. The distinction lies in the structure of memory. Princeton architecture devices use a single memory structure to store both program and data. Harvard architecture devices use separate memory structures for program and for data. The Princeton architecture is simpler in hardware, but leads to what is known as the von Neumann bottleneck in performance as the processor has to share time accessing the same memory for both instruction fetch and data.

Advocates for the Harvard architecture tout better performance because accesses to the program and data memory are independent and can be simultaneous. Harvard architectures are somewhat clumsy in program development environments, because it is impossible or at least not easy to modify program memory during execution of a program. Self-modifying code (a no-no among computer scientists anyway!) is generally not possible in Harvard architectures, for example. Regardless, with appropriate development-system design, it is possible to implement a system that allows program development using Harvard architecture processors, and the EasyAVR development board accomplishes that well. The challenge here is a need to rely on industrial tools for program development, rather than using custom tools that Princeton architectures allow.

Many Similarities

The S12 and the ATmega32 microcontrollers share many similarities, despite their fundamental differences noted above. This should ease the transition in the course from one processor to the other, and should allow most of the lab exercises used formerly with the S12 to be adapted easily for the ATmega32. Generally, the ATmega32 contains nearly the same types of resources as the S12, although generally the S12 contains more of each feature.

In the memory address space, both processors include the same types of capabilities (*Figure 3*). As can be seen in the figure, the ATmega32 includes the same features as the S12, but in each case the S12 contains more of each feature.

Feature	MC9S12DP256	ATmega32
Static RAM	12K bytes	2K bytes
EEPROM	4K bytes	1K bytes
Flash memory	256K bytes	32K bytes
Internal input/output	1K bytes	64 bytes

Figure 3: Memory in the S12 and ATmega32

The input/output capabilities of the S12 and the ATmega32 are also very similar, although again the S12 generally contains a larger quantity of the features offered. Figure 4 shows the input/output features included in each of the processors.

Feature	MC9S12DP256	ATmega32
Parallel ports	ten: 4-, 7-, and 8-bit	four: all 8-bit
Timer	one	three
Input Capture	eight channels	one channel
Output Compare	eight channels	four channels
Pulse Width Modulation	eight channels	three channels
Asynchronous Serial I/O	two systems	one system
Serial Peripheral Interface	three systems	one system
Inter-Integrated Circuit I/O	one system	one system
Controller Area Network	five systems	--
Analog to Digital Conversion	sixteen analog inputs	eight analog inputs

Figure 4: Input/Output in the S12 and ATmega32

The comparisons for memory and input/output detailed in Figures 3 and 4 reflect the capabilities of the specific processors used on the Dragon-12 and EasyAVR development boards. In each case, many other members of the processor family exist, with varying amounts of memory and input/output capabilities, so either processor family is likely to include a family member that meets the needs of a particular application. In the case of the microcontroller lab application addressed here, the S12 capabilities far exceed what is required, and many of the I/O capabilities are wasted. The ATmega32 is a better match for the needs of the microcontroller lab.

Plans

The microcontroller lab is now equipped with nine stations based on EasyAVR development boards, each with associated personal computers. The old lab implementation contained stations based on the Dragon-12 boards which were networked via serial terminal lines to a single personal computer host running multi-user linux. The new environment makes each station self-contained. Station maintenance is harder now, since each station has its own personal computer, but the old implementation suffered from a dependence on a single computer. If that one computer failed, the whole lab shut down. The new implementation will be more reliable.

Development of new lab exercises is underway. Many of the exercises will be similar to old lab assignments because of the similarity of the input/output resources of the ATmega32 to the S12's features. Some new capabilities, such as the GLCD graphics display and touch screen, will warrant new lab experiments to make use of those devices.

Looking ahead, the experience students will gain using the ATmega32 microcontroller in this required microcontroller course will encourage them to consider AVR processors in future work, notably in their senior project designs. This new microcontroller lab using the EasyAVR development boards will serve as a valuable resource for students using AVR processors, as it provides the hardware and software needed to configure AVR devices for specific applications.

Summary

The laboratory supporting the microcontroller course at the University of Minnesota Duluth has been redesigned to use a processor from Atmel's AVR microcontroller family. MikroElektronika's EasyAVR development board, using the ATmega32 microcontroller, provides access to the device's resources and includes convenient input/output interfaces for use in lab experiments.

References

1. Pack, Daniel J. and Steven F. Barrett, Microcontroller Theory and Applications: HC12 & S12, Pearson/Prentice Hall, Upper Saddle River, NJ, 2008.

2. ATmega32 User Manual, Atmel Corporation document 2503Q-AVR-02/11, 2011.

3. Margush, Timothy S., Some Assembly Required: Assembly Language Programming with the AVR Microcontroller, CRC Press, New York, NY, 2012.