THE EDUCATIONAL BENEFITS OF CREATING A DEVELOPMENT BOARD BASED IN-SYSTEM PROGRAMMER FOR 8-BIT EMBEDDED MICROCONTROLLERS

Jeffrey J. Richardson
Department of Electrical and Computer Engineering Technology
Purdue University

Abstract

Embedded microcontrollers are one of the fundamental building blocks of today's electronic devices. Universities and technical colleges throughout the world utilize development boards to teach the core fundamentals associated with embedded microcontrollers through a hands-on, practical approach. Purdue University's Electrical and Computer Engineering Technology Department requires the students enrolled in the introductory microcontroller course to purchase their own microcontroller development board. To keep initial cost low, the microcontroller board of choice relies on a boot strap loader to eliminate the need for a chip programmer. Having a personal development board, coupled with free development software allows the students unlimited access to the tools required to experiment with the fundamentals taught in the classroom. Utilizing a development board has many benefits for the end user, but can limit the user to one particular microcontroller for all applications. However, utilizing the development board to create a chip programmer allows the students to implement and program additional microcontrollers that are not pin-for-pin compatible with the development board. Utilizing the development board as a chip programmer allows the cost associated with the embedded microcontroller sequence to be minimized while gaining maximum returns. The development board based programmer also serves as an educational platform to teach a series of microcontroller learning modules.

Introduction

Many universities and technical colleges utilize development boards as part of their engineering and technology courses. The development boards are a critical component in the educational process. These boards allow the students to explore the technology and concepts explained during lecture in a practical, hands-on approach. At Purdue University's main campus in West Lafayette, all of the students enrolled in the Electrical and Computer Engineering Technology Department are required to purchase their own microcontroller development board for use in the sophomore level introduction to microcontroller course. The requirement to purchase a development board has been an evolutionary process and has many benefits. One of the biggest benefits is that the students can work from home. In addition, the students can reuse the development board in future courses and projects.

In an effort to keep initial cost low, the development board relies on the third generation AVR family of flash based microcontrollers from Atmel. These microcontrollers offer the ability of self-programming the flash program memory through a boot strap loader [1]. When a communication channel is present in the design, updating the flash program memory can be accomplished without the need for an actual programmer [2]. The current development board of choice, shown in Figure 1, is supplied by PRIIO LLC and is specifically designed for prototyping and laboratory use [3]. The

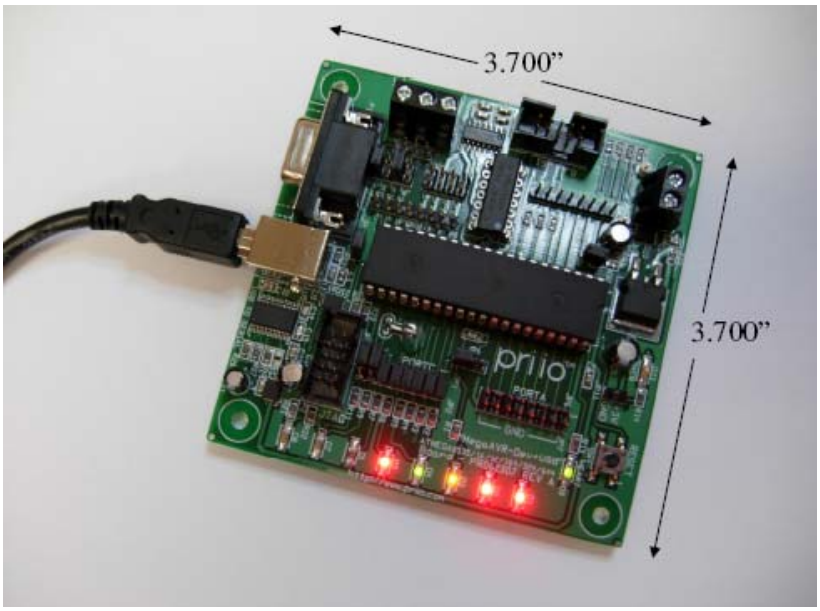

Figure 1 – MegaAVR-DEVelopment Board from PRIIO
(image from www.priio.com).

development board was chosen because it utilizes an Atmel 8-bit AVR microcontroller and is provided with the AVR Boot Loader from PRIIO. In order to program the microcontroller, the students download a PC application from PRIIO and make a simple serial connection between the development board and the host PC [3]. The boot loader eliminates the need for the students to purchase a chip programmer as part of the introductory microcontroller course.

Limitations of Boot Loading

Utilizing a boot strap loader is a very effective method of programming a microcontroller and has proven to be a cost effective solution for the introductory microcontroller course. However, the usage of a boot loader can produce several limitations in the educational environment.

The boot loader is actually a program that has been loaded into the boot loader section of the microcontroller's flash memory [4]. The boot loader is placed inside the microcontroller's flash memory by the supplier through the use of a traditional chip programmer and is tied to the ATmega16 microcontroller that comes with the development board. The non-portable nature of the boot loader prevents the students from programming alternate microcontrollers in the same manner. Only microcontrollers obtained from PRIIO with the boot loader preinstalled can be programmed in this fashion.

The Atmel AVR family of microcontrollers also contains fuse bits which allow the microcontroller to be configured or tailored depending on the specific application at hand. The fuse bits are used to select the system clock source and speed, whether or not the internal brownout detection circuitry is utilized and at what level, control the options associated with boot strap loader, the setup of the on-chip debugger, and the JTAG interface to mention the major functions. The fuse bits can only be modified through the use of a chip programmer and are not changeable with a boot loader. As a student progresses through the curriculum, the ATmega16 microcontroller may not be an appropriate choice when given different criteria or needs. In addition, the operating voltage and clock sources may also need to be changed to accommodate a particular situation. Therefore, an alternate solution is desired that will allow a multitude of different microcontrollers to be programmed and configured without adding the

expense of purchasing a commercial chip programmer.

Educational Need

As the student progresses through the curriculum, there are multiple opportunities to study microcontrollers and apply them in various projects. The Atmel ATmega16 is a solid choice for initially teaching microcontroller fundamentals but cannot serve as the only option for a microcontroller solution as the student proceeds through the curriculum, nor should it serve as the only choice. Students need to learn to evaluate other alternatives and make appropriate microcontroller selections depending on the design criteria at hand.

One such example is when the students are required to develop a project as part of their capstone junior and senior design sequence. During these courses, the students are expected to critically evaluate the various technical options to the given problem and make appropriate selections for the various components required to solve the problem. Selecting an appropriate microcontroller based on the design criteria is included in this activity. This can create a situation where the ATmega16 microcontroller may not be the best solution or create a need to change the fuse bits inside the microcontroller to adjust the system clock frequency or operating voltage. The students must be able to choose from a wide array of available microcontrollers and be able to implement them in the design.

Another such example can occur when the students enroll in a microcontroller based embedded networking course. The students study general topics associated with data exchange between multiple embedded systems and more specific topics like 2-wire interfacing (I2C), 3-wire interfacing (SPI), and complex UART data transmission and reception. In this course, it is common for a student to need more than one microcontroller to complete a laboratory assignment. In this case, simply having the students buy another development board is not an appropriate option. The boot loader also utilizes the standard UART and may create communication problems when trying to share this resource. In addition, the Atmega16 may not necessarily be the best choice for developing the required laboratory projects in this particular course as it is quite common for the experiments to require multiple UARTS. In this situation, the students need the ability to choose an appropriate microcontroller with multiple UARTS for the project and then successfully program it to accomplish the given task.

In-System Programming

In-system programming (ISP) is very common in the 8-bit embedded microcontroller world and has become the standard for new designs. Microcontrollers available from Atmel, as well as other manufactures, support this style of programming. Having the ability to program the memory of the microcontroller while it is located in the design provides increased system flexibility during the production cycle and makes field updates easier and less time consuming. ISP also reduces the development time and decreases time to market [1]. To accomplish this type of programming, an in-system programmer is required.

Commercial in-system programmers are available from several suppliers and distributors with a wide variety of features and an equally wide range of cost. The CableAVR is an in-system programmer available from PRIIO and retails for $89 [5]. The Atmel AVR ISP 2 is a commercial in-system programmer available from sources such as Digikey and Mouser for a retail price around $34 [6,7]. These solutions can overcome the limitations of the boot loader, but will add additional cost for the students. Requiring a student to purchase a commercial programmer is not a good use of student funds and is not necessary.

Other in-system programmers are available from places such as Sparkfun for as low as $12.95 but require a standard parallel printer

port or a RS-232 based serial port which are no longer standard equipment on nowadays laptop computers creating an entirely new set of challenges that would need to be overcame.

Free In-System Programmer

Atmel's application note AVR 910: In-System Programming describes the details of how to create an in-system programmer system [8]. An in-system programmer is designed around a core microcontroller that takes information from the development computer and converts it into the appropriate signals to transfer data to and from the target microcontroller through four conductors. An overall block diagram of an ISP is shown in Figure 2. The application note describes the creation of the in-system programmer based on the Atmel AT90S1200, a small 8-bit microcontroller that is no longer in production, and the accompanying software is written using Atmel's assembly language.

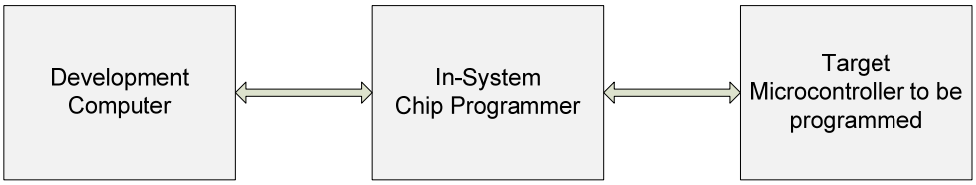

Figure 2 – Overall Block Diagram of an In-System Programmer Setup.

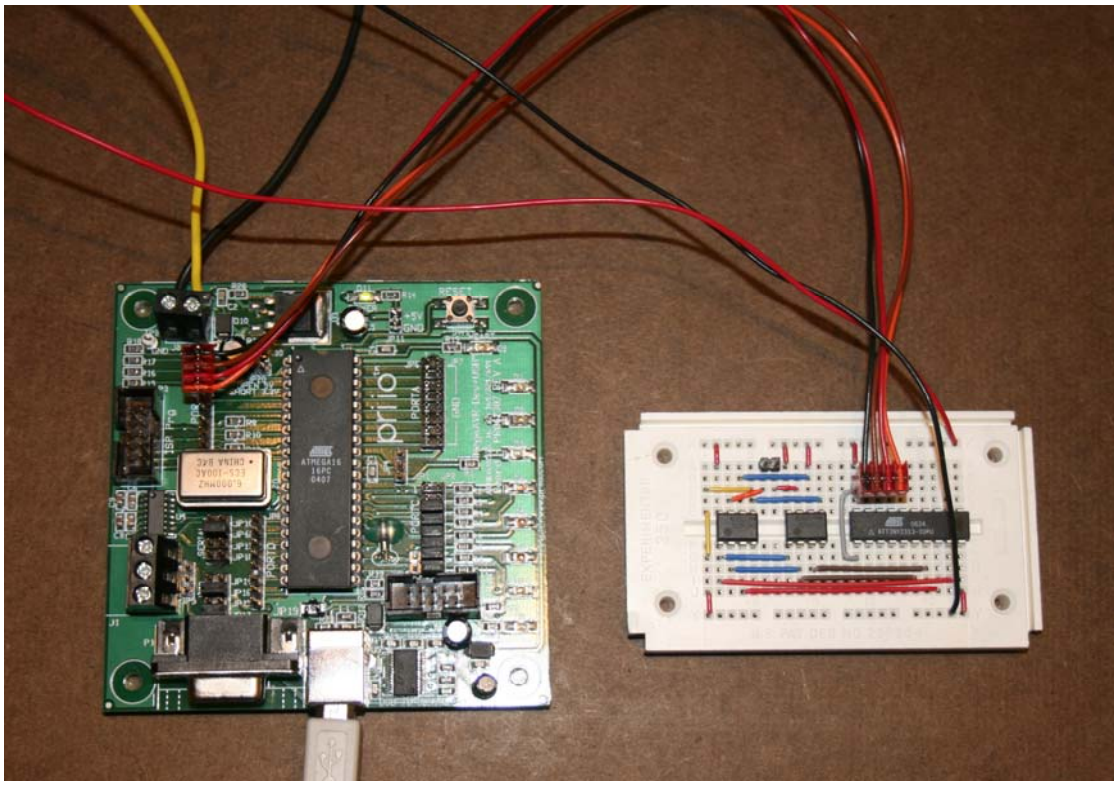

Figure 3 – Image of the Development Board Utilized as an In-System Programmer.

The obsolete microcontroller and associated assembly language program can create challenges in directly utilizing the application note in an educational setting. To overcome these draw backs, the in-system chip programmer's software was rewritten using the C programming language and is easily ported to be executed on the Atmel ATmega16 located on the student owned AVR development board. Rewriting the software allows the students to download the in-system programming application code into their development board through the boot loader. Once the code is downloaded, the process effectively turns the development board into a *free* in-system programmer. The image in Figure 3 shows the development board being utilized as an in-system programmer to program an Atmel ATtiny2313 8-bit microcontroller.

Software Overview

The overall software for the in-system programmer becomes fairly straight forward when converted into the C programming language. The in-system programmer's software monitors the serial data channel for incoming commands through the standard C input and output libraries and then makes an appropriate decision based on the command that was received.

The AVR910 ISP supports a mixture of single byte and multi-byte commands with most of the high-level details being handled on the development computer. Examples of single byte instructions include requesting the ISP identity, requesting the hardware and software versions, requesting the target microcontroller's device signature, and entering program mode. The data transfer instructions used to send information to and from the target micro-controller are based on a four byte command sequence. An overall flowchart of the main ISP software is shown in Figure 4 with excerpts from the C source code shown in Figure 5 & Figure 6. The complete source code and accompanying educational documentation can be downloaded from the following link: http://www.tech.purdue.edu/ecet/courses/reference material/atmel/ (under the chip programming option).

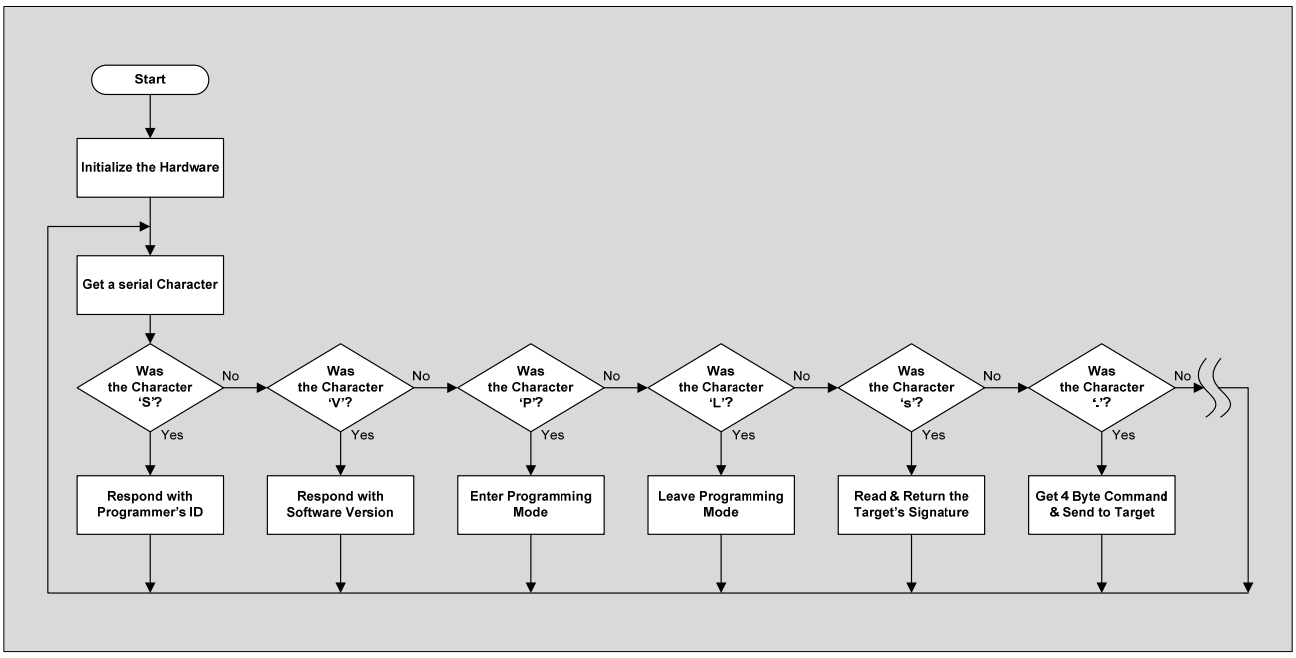

Figure 4 – Overall Software Flowchart for the In-System Chip Programmer.

```
// example code segment taken from the main function

init_hardware();    // setup the serial communication channel

while (1)
{

   ch = getchar();              // get a character from the development computer

   switch ( ch )
   {
         case '.':                    // was the command 'Universal Command'

         cmd1 = getchar();            // get a 4 command sequence...
         cmd2 = getchar();
         cmd3 = getchar();
         cmd4 = getchar();

         write_SPI(cmd1);             // send 4 command sequence to Target
         write_SPI(cmd2);
         write_SPI(cmd3);
         cmd = write_SPI(cmd4);       // get the respond from the Target

         putchar(cmd);                // send Target response to PC

         putchar(0x0D);               // respond with OK
         break;

         . . .  // additional case/switch code omitted for readability

   }
}
```

Figure 5 – Software Excerpt from the ISP Main Function.

```
// Software function to bit-bang the port pins to communicate with the target microcontroller.
// Manual bit-banging of the port pins allows any of the general purpose I/O pins to be utilized.
unsigned char write_SPI(unsigned char SPI_txdata)
{
         unsigned char SPI_rxvalue;
         unsigned char cntr;
         unsigned char mask;

         SPI_rxvalue = 0;
         mask = 0x80;

         for (cntr = 0; cntr<8; cntr++)                  // handle all 8-bits
         {
                  if ( (mask & SPI_txdata) != 0)         // Output data to the target micro
                  {
                           MOSI = 1;                     // set the output pin?
                  }
                  else
                  {
                           MOSI = 0;                     // clear the port bit?
                  }

                  SCK = 1;                               // start the clock cycle
                  mask = mask >> 1;                      // move to the next bit
                  delay_us(10);                          // stretch the clock cycle

                  SPI_rxvalue = SPI_rxvalue << 1;        // make room for next bit

                  if ( MISO != 0 )                       // is the incoming bit set??
                  {
                           SPI_rxvalue = SPI_rxvalue | 0x01;   // then set this bit
                  }

                  SCK = 0;                               // finish the clock cycle
         }

         return (SPI_rxvalue);                           // return the data from the slave
}
```

Figure 6 – Software Function to Transfer Data between the Target and the In-System Programmer.

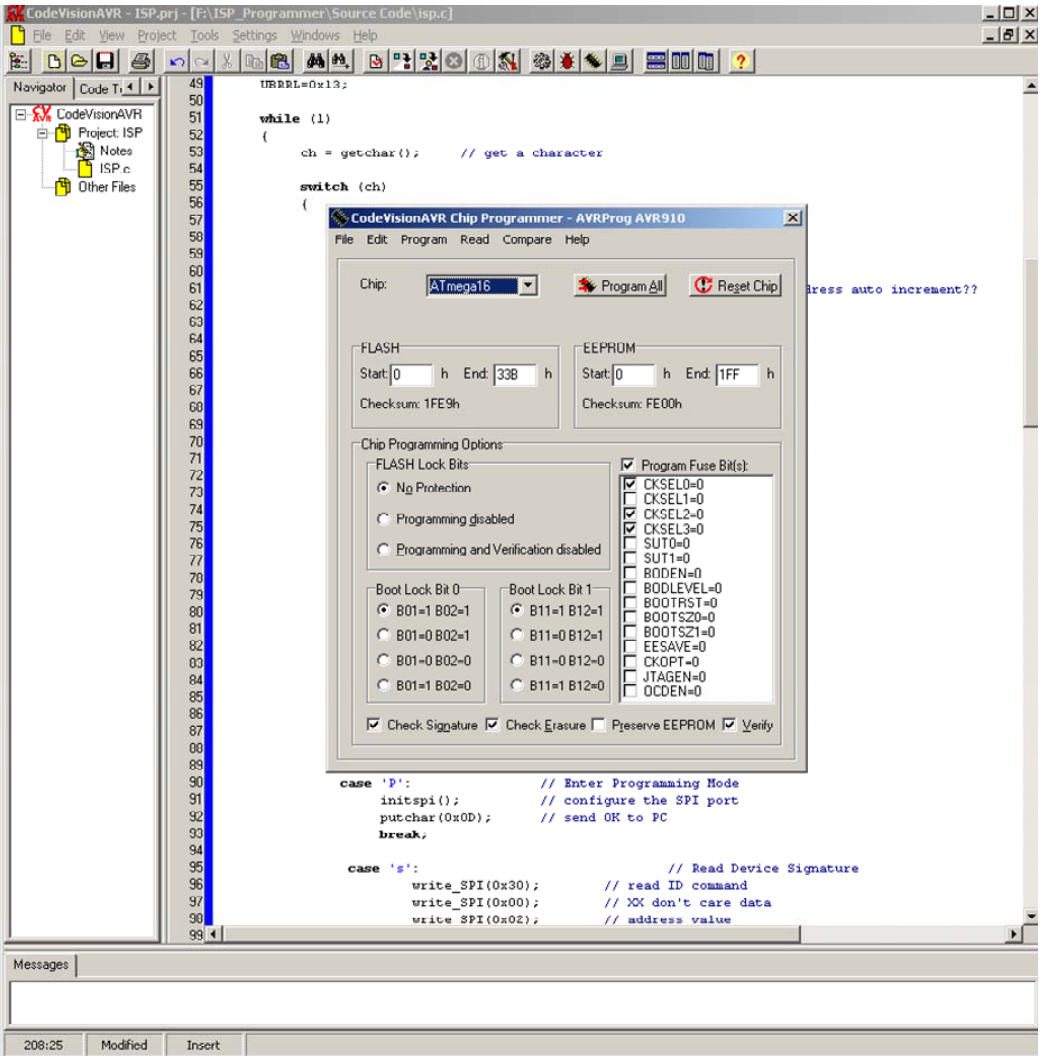

Figure 7 – Screen Shot of the CodeVisionAVR IDE Chip Programmer Front-end Software.

Development Computer

Having the physical hardware and software for an in-system programmer is only one piece of the solution. An application is needed on the development computer to facilitate the transfer of data to the target microcontroller through the in-system programmer. The CodeVisionAVR integrated development environment (IDE) is the development environment of choice for the embedded microcontroller sequence and supports a variety of chip programmers including one based on Atmel's AVR 910 application note protocol, Figure 7. A free evaluation version of the CodeVisionAVR IDE software package and can be downloaded from http://www. hpinfotech.ro/. The IDE allows the students to create and modify source code for the target microcontroller and then download the application code through the microcontroller development board based in-system programmer with just a few clicks of a mouse.

Educational Benefit

First and foremost, the in-system chip programmer detailed in this article can save the students money by not requiring them to purchase additional hardware to accomplish the same results as a commercial chip programmer. The costs of the various programmers discussed in this article are shown below in Table 1.

Table 1: Cost of ISP Programmers.

Programmer Description	Cost
Atmel AVRISP2 Programmer	$34
PRIIO – CableARV Programmer	$89
Development Board Based Programmer	NA

The in-system chip programmer serves as a vehicle to provide a series of teaching modules within the embedded microcontroller sequence. The following is a list of educational learning objectives made possible through the use of the in-system chip programmer.

- Students are introduced to chip programming and the various techniques utilized to accomplish this task including high voltage parallel programming, serial in-system programming, and self programming
- Students compare and contrast the various programming methodologies
- Students analyze the benefits of utilizing in-system programming in a production environment
- Students learn the hardware design criteria required to incorporate in-system programming into a new design
- Students apply the information to design and construct their own version of an in-system chip programmer
- Students are able to explore the effects of altering Atmel AVR microcontrollers to change the operating voltage through the modification of the internal brown-out detection circuitry and set points
- Students analyze the performance characteristics of internal RC oscillators circuits versus external crystal oscillator circuits for critical timing of events and serial communication
- Students utilize multiple microcontrollers in designs other than the standard ATmega16 that comes in the development board.

The in-system chip programmer also allows the students to utilize a wide variety of AVR based microcontrollers in their future designs. This allows the students to make appropriate microcontroller choices based on the requirements of the system, not based on a microcontroller that they happen to have because of a previous course. This feature allows the students in upper level courses to implement top-down design methodologies instead of bottom-up design. When utilized in the capstone senior design sequence, the students download the source software from the web site previously listed and only modify the IO connections as necessary to facilitate the connection to their target microcontroller. The in-system chip programmer has been successfully utilized to program the following AVR based microcontrollers: ATmega8, ATmega16, ATmega162, ATtiny2313, and ATtiny13 providing the students a variety of options and features to choose from. The highlights of these various microcontrollers are shown in Figure 8.

Test Results

The development board based in-system programmer was utilized to configure the fuse bits of a variety of microcontrollers from the Atmel AVR family. To simplify the displaying of the results, only the Atmega162, ATmega8, ATtiny2313 and ATtiny13 are shown here. Note, these are the same AVR based microcontrollers listed previously which provide a wide variation of form, fit and function.

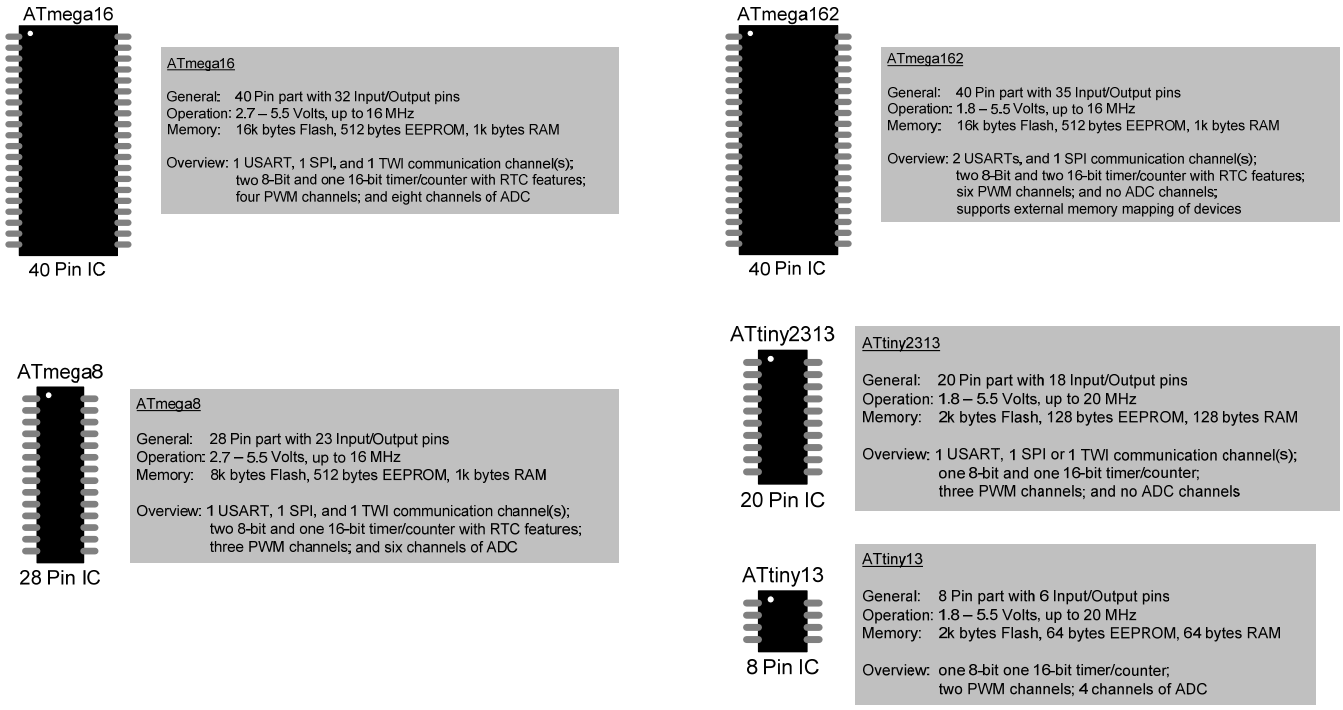

Figure 8 – Overview of Various Microcontrollers Made Available for Student Use.

Table 2: Programming Test Results.

	Individual Results				Overall Results
Test Procedure	**ATmega162**	**ATmega8**	**ATtiny2313**	**ATtiny13**	**Pass or Fail**
Read Microcontroller's Signature	Pass	Pass	Pass	Pass	Pass
Set Fuse Bits	Pass	Pass	Pass	Pass	Pass
Erase the Flash Memory	Pass	Pass	Pass	Pass	Pass
Program the Flash Memory	Pass	Pass	Pass	Pass	Pass
Verify the Flash Memory	Pass	Pass	Pass	Pass	Pass

During this test, each microcontroller's fuse bits were modified 25 times and verified.

After the fuse bits are set, a typical programming sequence consists of erasing the flash memory of the target microcontroller, programming the flash memory with the new application, and finally verifying the contents of the flash memory. Multiple tests were conducted with the microcontrollers listed in Table 2 to verify proper operation of the in-system chip programmer. Each microcontroller was erased, programmed and verified 50 times.

Conclusions

The in-system chip programmer described in this article has been successfully utilized to program a wide variety of the available Atmel AVR based microcontrollers. The programmer has also been successfully utilized by numerous upper level students to program various microcontrollers in their capstone junior and senior design courses without the need to purchase a commercial chip programmer, ultimately saving the students money. The students find that this is a fitting use for the

development board and feel its use as a programmer contributes to the boards overall value.

The development board based in-system programmer also allows the students to implement top-down design methodologies during their upper-level courses by providing a wide selection of microcontrollers to choose from. The in-system programmer also serves as an educational vehicle to allow a multitude of topics related to programming embedded microcontrollers to be studied.

References

1. O. Svendsli, "Atmel's Self-Programming Flash Microcontrollers", http://www.atmel.com, 2003, (accessed March 2009).

2 Anon, "Atmel ATmega16 Datasheet", http://www.atmel.com, 2005, (accessed March 2006).

3 L. O'Cull, "Mega-AVR Development Board", http://www.priio.com, 2008, (accessed March 2009).

4 Anon, "AVR109: Self Programming", http://www.atmel.com, 2004, (accessed March 2009).

5 http://www.priio.com, (accessed March 2009).

6 http://www.mouser.com, (accessed March 2009).

7 http://www.digikey.com, (accessed March 2009).

8 Anon, "AVR910: In-System Programming", http://www.atmel.com, 2000, (accessed March 2009).

Biographical Information

Professor Richardson teaches courses in the embedded microcontroller sequence and the capstone senior/junior design sequence within the Electrical and Computer Engineering Technology Department at Purdue University. His current research interest include exploration of various methods for effectively teaching embedded microcontroller fundamentals and the effects of learning styles in open-ended problem solving.