AC 2007-1887: NEW DEVELOPMENTS FOR COURSES IN EMBEDDED MICROCONTROLLERS

Todd Morton, Western Washington University

Todd Morton has been teaching the upper level microprocessor and digital courses for Western Washington University's Electronics Engineering Technology program for 18 years. He is the author of the text 'Embedded Microcontrollers', which covers assembly and C programming for the 68HC12. He has also worked as a design engineer at Physio Control Corporation and has worked several summers at NASA's Jet Propulsion Laboratory as an ASEE-NASA Summer Faculty Fellow. He has a BSEE and MSEE from the University of Washington.

New Developments for Courses in Embedded Microcontrollers

Abstract

This paper describes new course outcomes and laboratory tools used for teaching Embedded Microcontrollers at Western Washington University. These developments include the new Freescale 9S12 University board and the application of both 'all-in-one' and 'stamp' type development boards in the curriculum. Basic fixed-point DSP programming, and an introduction to the Controller Area Network (CAN) will also be discussed.

Introduction

In this paper I will talk about some of the changes we have made here at Western Washington University's Electronics Engineering Technology program[8]. Specifically, changes to the junior and senior level microprocessor, embedded systems, and senior project courses. Most of these changes have been made practical because of the continued developments in the embedded microcontroller industry. New microcontrollers continue to get more powerful – or, in some cases, less powerful – with increases in memory, on-chip resources, and bus speeds.

We have traditionally used Motorola (now Freescale) microprocessors and microcontrollers including the 6800, 6809, 68HC11, 68HC12, and now, the 9S12 family of microcontrollers. Because of this, we will focus on the 9S12 family in this paper, though many of the new technologies apply to other manufacturers as well.

We have found that the 9S12 family is a very versatile family of parts. This allows our students to leverage the skills they have received in previous courses to work efficiently on their senior projects. Examples of projects using embedded microcontrollers can be found on our project website[1]. The seniors can choose any microcontroller that makes sense for their project. However, because of their familiarity, most choose one of the 9S12 parts. As shown in Table 1, in the last three years, the seniors used 25 9S12/68HC12 parts out of 33 microcontrollers used. Again, this speaks more towards the versatility of these parts, not that they were the 'best' solutions for the specific project. After all the students will tend to choose what they are familiar with.

MCU Used	**Quantity**	**Comments**
9S12DP256/512	17	Very versatile, general purpose MCU
9S12C32/64/128	7	Ideal 9S12 part for small low-cost projects
Misc. USB Modules	3	A pre-designed USB module for simple USB designs
Cypress PSoC	2	Ideal for small designs that can benefit by having analog on the MCU.
9S12NE64	1	A 9S12-family Ethernet Solution
Coldfire 52233	1	Very good embedded Ethernet Solution.
Other 9S12, HC12	2	Older parts

Table 1 – Microcontrollers Used in Senior Projects (2005-07)

Notice the new technologies represented - USB, Ethernet, On-chip analog. These represent a small but typical example of the evolution of embedded microcontrollers.

Our program currently uses the 9S12DP512, 9S12DG128, and 9S12C32 microcontrollers in the junior and senior level courses. The rest of the paper will focus on curriculum based on these parts.

Course Outcomes

Course outcomes should represent the fundamental components that, when combined, build an appropriate skill set in the area. A specific application or technology should not be included in the outcomes unless it is truly a fundamental requirement. All of our course outcomes are mapped to the ABET program outcomes a-k along with the IEEE program outcomes for Electronics Engineering Technology[4,5]. Most of our course outcomes have not changed with changes in technology. Many of the fundamental concepts remain the same. However, there are changes that truly reflect fundamental changes:

Outcome: Understand decoders and memory expansion techniques
Changes: We do still cover the basics of bus interfacing and memory expansion. However, because we use microcontrollers exclusively and there no longer is an accessible bus system, this area has been deemphasized. Approximately, 8 hours of material down to the equivalent of three hours.

Outcome: Competency in assembly language programming and programming tools including a programming editor, assembler, and debugger.
Changes: This outcome has become essential. In the distant past, it was ok to have students hand assemble code. Now, a student must be able to use basic software development tools. This has had little impact on lecture time but the student is much more efficient in the lab.

Outcome: Have a basic understanding of common on-chip resources and their applications including general-purpose I/O, timers, serial ports, and A-to-D converters
Changes: Traditional microprocessor classes covered peripheral devices and interfacing these devices to the CPU. However, much of the time was spent covering the hardware interface. With microcontrollers the hardware is done, so the focus can be placed on the driver software and applications of the peripherals. One peripheral that once was an afterthought and now is essential is the Serial Peripheral Interface (SPI). The SPI is a high-speed, low-cost, synchronous serial communication module. Without a bus system, most peripheral devices added to a system use the SPI or another compatible synchronous interface. Over half of the senior projects in the last three years have required the SPI.

Outcome: Understand requirements and be able to implement a stand-alone MCU-based design.
Changes: In the past, with the educational tools of the time, it was difficult for students to develop stand-alone systems (final products). Using the new, flash-based, systems, there is no good reason for students to not be able to develop their code all the way to a final product.

There are several other minor changes to outcomes but the ones listed reflect fundamental changes due to the change in current technology. There are also some potential changes to other courses in the future. For example, now that all of our MCU boards include CAN, it is realizable to develop labs for the digital communications course to demonstrate the CAN bus. Some simple DSP labs can also be added to this course.

Development Boards

For most microcontroller families there are a wide range of development boards available. In general, we can divide board types into two types – 'all-in-one' education boards and 'stamp' type development boards. The all-in-one type boards have primarily evolved from the classic education trainer systems such as the Heathkit 6800 system or the Intel SDK80 system. The 'stamp' type boards evolved from the microcontroller vender's development boards such as Microchip's PIC developments systems – the system that made the term 'stamp' popular.

The all-in-one boards traditionally were designed for student laboratories, so they were designed for user programs stored in RAM and included extra circuitry for laboratory experiments. Because of the cost of the boards, the university normally purchased the boards for the students to either use in lab or to checkout. The primary advantages of these boards are that they were simple to program, many labs could be developed that required no circuit construction, and if made popular, there would be many supporting textbooks, lab experiments, and supporting documentation.

The problem with these systems was that they did not work well as practical final products. Because they were designed for programs that are run in RAM, it was difficult for a student to create a non-volatile solution. In addition, because of the added circuitry, the systems tend to be larger than most projects require. In some cases, because of the extra circuitry, some on-chip resources may not be available for use in a different application. These boards are fine for a standard educational laboratory environment. They are not good solutions for projects that require prototypes that are close approximations of a real product design.

Another problem with the all-in-one paradigm is that the technology started changing at a rapid pace. With the introduction of new MCU families, flash ROM, a plethora of on-chip resources, and on-chip debugging systems, current systems quickly became obsolete. The market continued to change so it no longer was practical to invest in an expensive trainer system and expect it to last for several years.

While technology was changing, curriculum changed too. More schools developed senior project courses, an area of weakness for the all-in-one boards. In reaction, many schools started using simple development boards and required the student to purchase the board. This made it possible for the schools to keep up with technology and allowed the student to keep his or her project intact. However, this came at a price. The change to small development boards caused the education market in this area to fragment, resulting in fewer textbooks and other supporting materials. Publishers are reluctant to publish textbooks if the market is too fragmented[9]. It seems now that everyone uses a different MCU, a different software development system, and a different trainer board. All of these require significant amounts of learning material.

On the other end of the spectrum from the trainer systems are the ‘stamp’ type boards. These boards have very little extra circuitry beyond that required to run the microcontroller. Therefore, application-specific circuitry can be added with the result being very close to a real final product. In education, the stamp board essentially replaces the IC as it can be used as a surface-mount to through-hole adaptor. Because of this, stamp type boards are ideal for student projects where circuitry is constructed one time. However, in the standard laboratory environment, the students have to construct all external circuitry required by each lab. This can limit the lab experiment possibilities due to time constraints.

Here at Western we have been primarily using the simple stamp-like development boards. In our case, we have been using two Technological Arts development board systems – the Adapt9S12 family and the Nanocore12 family. These are well designed boards and have served our students very well – especially for senior projects.

This year, we changed directions for one of the laboratory-based courses. This change was made because of the release of Freescale’s S12 University board (S12UB). It is an all-in-one type board that has a very good combination of features and takes advantage of the technologies in the 9S12 family of microcontrollers. At $85, it is priced about the same – or less than – the simple development boards. We will be using the S12UB for the laboratory courses then the students will have the option to use the S12UB or one of the other two development boards.

With the low price of the S12UB, we can use that same cost structure we have been using for simple development boards. This cost structure relies on students to pay all of the costs for the development boards but allows them to keep whichever board they choose for their project.

We do this by charging a use fee in the three junior and senior courses. We adjust this fee, based on the costs of all boards required for the three courses. Currently the fee is $45 per course. However, with the new S12UB board, we may be able to reduce this fee. The fee also covers all software development tools (software and BDM PODs), board failures, serial cables and power adaptors, if needed.

Then, when the student finishes their project, they can choose one of the boards to keep. Of course, they normally choose the board used in their project so they can keep their project intact. This system keeps the board supply fresh and allows us to change to new boards with new technology as it arises. When we started this system, the department did pay approximately $1000 for the first two years. From now on, it should be self sufficient through the use fees.

Freescale 9S12 University Board

Figure 1 shows the S12UB board with the different blocks identified[15]. It includes a Background Debugging Module (BDM) POD on the board, power drivers and connectors, a Local Interconnect Network (LIN) interface, a CAN interface, a serial interface for the target and the POD, LEDs, switches, A/D inputs, a piezo buzzer, and a convenient and versatile prototyping area. An I/O kit can also be ordered, which includes an incandescent light, LCD display, keypad, relay, mechanical encoder wheel, and DC motor.

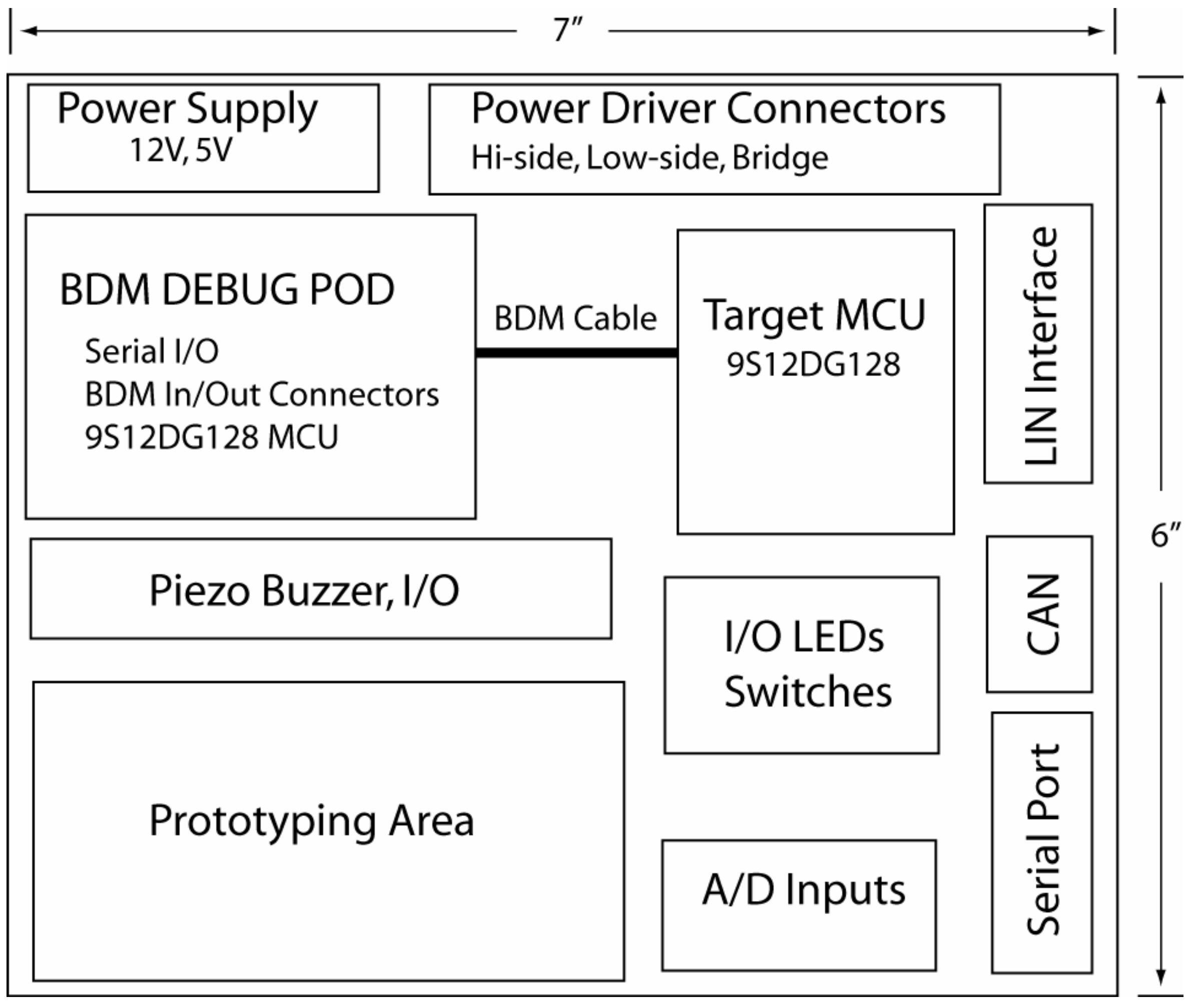

Figure 1 – Freescale S12 University Board

One of the most significant features of this board is the on-board debugging POD. This allows for a wide range of debugging options. Here are some of the options and their strengths and weaknesses.

Resident D-Bug12 Monitor. The simplest debugging option is to download the monitor, D-Bug12, onto the target MCU. This provides the traditional command line based monitor with commands for loading the program, reading and writing to memory locations, and setting breakpoints[2]. D-Bug12 is easy for students to learn, there are plentiful learning materials covering D-Bug12, and only a terminal emulator[3] is required on a PC. This environment is the easiest to learn and use for simple assembly programs. However, because the monitor is a resident program, the student can not create a final product and it does not allow for good real-time debugging. It is even a larger handicap when programming in C as it is not a source-level debugger. We currently use this system for the first five introductory labs in our junior course.

A simple variation to this is to use Freescale's Serial Monitor[16] program resident on the target MCU. The serial monitor has a binary communications protocol so it is much more efficient with respect to code space and communications speed. In combination with the CodeWarrior[10] debugger, a more modern and efficient debugging environment is possible – including source level debugging. Compared to D-Bug12, the serial monitor improves the interface to the CodeWarrior debugger but still has the other limitations. The smaller footprint makes final products more realizable but it still suffers as a real-time debugger. An additional limitation is that it uses the serial port for the interface to CodeWarrior. Therefore, it does not allow for applications that use the serial port. The 9S12DG128 has another serial port but it is used for the LIN interface.

D-Bug12 POD. In this configuration D-Bug12 is programmed into the POD MCU and run in POD mode. This provides for a D-Bug12 interface to the PC and a BDM interface to the target. It combines the ease of use of D-Bug12 with the non-invasive debugging features of the BDM. For example, the BDM system allows the programmer to view memory locations while the target is running in real time. Also, since there is no resident monitor on the target, this system allows student to produce a final product. Because of the command-line interface, a limitation of this system is that memory locations must be read manually. i.e. the user has to type the memory dump command for a refreshed display of the contents. In general the debugging display is not as informative or interactive as a full debugging application like CodeWarrior. The assembly laboratories that come with the board use this system along with the last two labs in our junior-level course.

Serial Monitor POD. The combination of the serial monitor installed in the POD MCU and the CodeWarrior debugger has the potential to be a great, low cost, source-level debugging solution for real time systems. It combines the interactive, source-level display of CodeWarrior with real-time (periodic) updates made possible by the BDM. And, since the BDM is used, the target can be running final product software. Unfortunately, at this time, the CodeWarrior debugging configuration for the serial monitor is limited. Using the serial monitor as a POD is new, so developments are currently being made to the CodeWarrior debugger. Once these improvements are complete, the system will essentially have a full source-level BDM system without the extra cost of an external BDM POD.

External Debugging POD. An even more powerful debugging solution is one that uses an external POD connected to the target BDM connector. It then connects to either the parallel or USB port of a PC. The system we have been using for several years in the senior level embedded systems course and senior projects course is the Noral Flex debugger. It can be used with any 9S12 board or final product because it connects directly to the BDM of the target MCU. The only extra resource needed on the board is an inexpensive BDM connector. It is a very good system but is relatively expensive (~$2500). It is typically too expensive for a hobbyist or student. Since the debugging application is much better than the CodeWarrior debugger, an engineer working on product development would gladly pay the extra cost. Another less expensive POD solution is the P&E Micro Multilink USB (~$50). But this POD does not come with a full debugging application so it is normally used with the CodeWarrior debugger.

Clearly the BDM solutions are the preferred solutions for normal product development. However, there is an advantage to using the resident monitor for beginning students. When a student is first starting to program, we need to make it as simple as possible. After all, they are already overloaded the large number of concepts needed just to start programming a microcontroller. One of the things a resident monitor does is perform all of the required MCU initialization in its own startup code. This allows the student to write simple program snippets without dealing with startup code or reset vectors.

For example, when using the 9S12DG128 MCU, the memory map and the oscillator PLL are not optimally configured out of reset. The resident D-Bug12 configures the memory map and sets up the bus frequency during its initialization. The code for doing this is fairly simple but not for a beginning student. Also, if code is being loaded into RAM, the RAM location must be moved before the program is even loaded. By using a resident monitor, the beginning student can be sheltered from these added complexities and from the confusion of introducing multiple memory maps.

In Western's junior-level course, students are introduced to D-Bug12 resident on the target for the first five labs. This simplifies the task of writing the programs and introduces them to a simple command-line resident debugger – a tool they are likely to encounter in industry. Then once they are introduced to the required startup code, they jump to using the D-Bug12 monitor on the POD. From that point, all of their labs are programmed into target flash and are essentially final products. As Freescale develops a better CodeWarrior interface with the new serial monitor, we plan on transitioning from the D-Bug12 POD to the serial monitor POD.

Software Tools and Teaching Materials

In addition to the 9S12UB hardware, Freescale has done a very good job of providing the required documentation, software tools, and several good laboratory exercises on the included CD. Though I have created my own labs to fit our program, it is apparent to me that a course could depend solely on the laboratory exercises included with the CD. Labs will continue to be developed and placed on the 9S12UB website.[15]

The board also comes with the special edition of the CodeWarrior development system. Currently, the special edition is free and has a 64kByte limit, which is plenty for use in the educational environment. At this point, we have not had a single senior project that required more memory than the special edition limit.

The CodeWarrior IDE can be a frustrating development system to learn and use. However, with careful development of stationary, the students can be sheltered from the potential pitfalls. We have developed, and currently use a CodeWarrior stationary for absolute, single-file, assembly development. With this stationary, once the students catch on to some of the quirks, the system can be very productive. It would also be very easy to run the assembler from the command line using the command window or Cygwin. Underneath the IDE, the CodeWarrior assembler and C compiler are excellent. For our senior-level embedded systems and projects course, which use C, we use Borland's CodeWright editor in combination with GNU *make*, to run the CodeWarrior tools from the command line. Though certainly not perfect, this system is much better than the CodeWarrior IDE. It also allows students better access and a better understanding of the build process.

Initial 9S12UB System Assessment

The new S12UB board has been used in one course so far. As a preliminary assessment, I have compared the student success rates for the final laboratory experiment, a realtime security system. This system is a final product so it primarily relates to the students ability to meet the final product outcome for the course. Table 2 shows the student scores for this lab during the last five years. The table shows a significant improvement using the new S12UB board.

Year	Normalized Average Score	Development Board
2007	0.919	S12UB
2006	0.629	Nanocore12
2005	0.750	M68EVB912B32
2004	0.846	M68EVB912B32
2003	0.828	M68EVB912B32

Table 2 – Student Success in the Final Lab in ETec374

In addition, this year an extra credit option was added because the S12UB board includes a piezo buzzer connected to the on-chip pulsewidth modulator. In past years, students would have had to build the circuit on the solderless breadboard. This year 11 out of 14 students chose to complete the extra credit option. From all of the buzzing going on in the lab and the comments by the students, this was a very enjoyable experience (Note – there is a jumper on the board to turn the buzzer off).

Compare this to 2006 when we tried to use the Nanocore board. It turns out that it is not as practical to create final products using the resident serial monitor, which we used on these boards. So, developing a final product is a weak spot for these boards and it shows in the assessment above. These boards are however very good for senior projects where an external BDM POD is used to program the flash and debug the system.

More data will be needed and new labs will continue to be developed. But it clearly looks like we are on the right track.

Embedded Networking

One area that has really become important in the embedded microcontroller industry is communication systems, or networks. Processing has become more distributed, which in turn, requires more advanced means of communications. Examples that are becoming common in newer microcontrollers include Ethernet, USB, CAN, LIN, and now the implementations for IEEE 802.15.4 wireless networks such as ZigBee[11,12]. These technologies are in addition to the normal synchronous and asynchronous serial ports such as SCI, SPI, and IIC modules that have been common for several years.

All of the 9S12 family parts that we use include a Controller Area Network (CAN)[13] controller so it a convenient candidate to include in the program curriculum. CAN has many applications from the SAE standard vehicle busses to building automation standards like DeviceNet. It can also be used as a case study of a network that is based on the producer-consumer model as opposed to the more familiar client-server model[6]. The CAN and LIN networks are widely used in the automotive industry. So, another reason we have focused on CAN is because of our Vehicle Research Institute (VRI). If our students have some basic understanding of the CAN or LIN standard, it allows them to work on interdisciplinary projects with the VRI. An example of this is Tim Jackson's senior project, which was a CAN network controller for the Viking 32 hybrid car[7].

We have focused on the following outcomes in order to give are students a basic understanding of the CAN bus.

1. Understand the CAN standard and how it fits with other related standards. The CAN standard only addresses the data link layer and medium access control. Examples of different application layers and physical layers are covered.

2. Be able to program a simple driver for the 9S12 CAN module.

Future considerations for embedded networking will include the new Coldfire family, the MCF5223x. These MCUs include Ethernet and CAN on the same MCU with future devices possibly having USB and ZigBee. Most Ethernet applications require more processing power and memory than the 99S12 MCU with Ethernet (S12NE32) provides. Jon Peterson's Voice Over IP senior project[14] really pushed the 9S12NE32 to the limit. He had to make compromises that he would not have felt comfortable with in a final product. The Coldfire MCU would have been an ideal fit.

Simple DSP

Microcontrollers are getting faster and more powerful. This has opened the window to simple DSP through the voice audio range. After receiving input from our industrial advisory committee, we have focused on the skills required for students to develop basic DSP programs in a microcontroller. They were more interested in this than the standard DSP course using computer applications like Matlab and/or using a digital signal processor. Again, the emphasis should be on fundamentals, which the student can build on through continued learning. With this in mind, the following outcomes were identified to prepare students with some basic DSP skills.

1. A strong understanding of fixed point arithmetic. Students should be able to write programs in assembly and/or C, that use fixed point numbers with varying radix points. They should understand the basic limitations such as resolution and overflow and be able to determine the required word size and radix point.

2. A strong understanding of sampling theory and A/D, D/A technologies. Students should understand the complete A/D and D/A process including analog scaling and filtering. Different A/D and D/A technologies should be understood including PCM, sigma-delta, PWM, companding, and over sampling.

3. Applications in signal generation - synthesis. Students should be introduced and have hands-on experience with signal generation using a microcontroller. Good candidates include sinewave generation using lookup and/or mathematical techniques, noise generators using random number techniques, or generating audio from prerecorded sources such as stored .wav files.

4. Applications in signal modification. Students should be introduced and have hands-on experience with signal modification applications. They should be introduced to basic digital filtering and understand the differences between FIR and IIR filters. They should also be introduced to the DFT and its implementations. Focus should be on implementations that work well with the limitations of a general purpose microcontroller.

Currently, we do not have the resources to introduce a new DSP course. Therefore, the topics listed are covered throughout the curriculum, primarily in the junior microcontroller course, the embedded systems course, and the digital communications course. Again, the evolution of the microcontroller has made this area more realizable and, in turn, it has made these outcomes more important to our employers.

Conclusion

To me, this is an exciting time to be working with embedded microcontrollers. The flash-based MCUs with background debugging systems have made the development process more interactive and easier to use. The processing power and bus speeds are starting to open new doors in the areas of DSP and networking. The only issue is 'how do we fit it all in and when will there be adequate teaching materials available?' The 9S12UB looks like a well designed teaching tool. Freescale is doing a good job of providing documentation and laboratories and I will be doing the same and providing these materials on my website. Hopefully, it becomes a popular board to help defragment the educational MCU market and, in turn more materials, such as textbooks, will be published.

Bibliography

1. Western Washington University Electronics Engineering Technology's Senior Project Website, http://eet.etec.wwu.edu/etec474/index.html, Todd Morton
2. Reference Guide For D-Bug12 Version 4.x.x, Gordon Doughman, Freescale Semiconductor
3. Teraterm Pro, http://www.ayera.com/teraterm/
4. WWU EET Course Outcomes, http://eet.etec.wwu.edu/eetprog/coursespecs/coursespecindex.html
5. ABET Criteria, http://www.abet.org/forms.shtml.
6. Controller Area Network, Conrad Etschberger, IXXAT Press, Weingarten, Germany
7. Viking 32 CAN Controller, Tim Jackson, http://eet.etec.wwu.edu/jacksot3/index.html
8. Western Washington University Electronics Engineering Technology, http://eet.etec.wwu.edu
9. Embedded Microcontrollers, Todd Morton, Prentice-Hall
10. CodeWarrior Development Tools, Freescale Semiconductor, http://www.codewarrior.com/
11. IEEE802 Part 15.4: Wireless Medium Access Control(MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), LAN/MAN Standards Committee of the IEEE Computer Society
12. ZigBee Alliance, http://www.zigbee.org/

13. CANOpen, http://www.canopen.org/
14. Voice Over IP Senior Project, Jon Peterson, http://eet.etec.wwu.edu/peter34/index.html
15. 9S12 University Board Website, Freescale Semiconductor, http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=LFEBS12UB&fsrch=1
16. AN2548 – Serial Monitor Program for HCS12 MCUs, Jim Williams, Freescale Semiconductor