2006-1969: USING EMBEDDED SYSTEMS TO TEACH ALL LEVELS OF PROGRAMMING TO ELECTRICAL ENGINEERING STUDENTS

Joerg Mossbrucker, Milwaukee School of Engineering

JÖRG MOSSBRUCKER Dr. Mossbrucker is Assistant Professor of Electrical Engineering and Computer Science at the Milwaukee School of Engineering (MSOE). He did graduate studies at Michigan State University and received the Ph.D. degree from the University of Kaiserslautern, Germany. He has extensive industrial experience and teaches courses in analog and digital circuits, microprocessors, and computer programming.

Using Embedded Systems to Teach All Levels of Programming to Electrical Engineering Students

Abstract

This paper describes the implementation of a programming sequence in Electrical Engineering starting at the freshmen year which places a considerable emphasis on embedded systems. Freshmen are introduced to procedural programming techniques using a popular 8-bit micro-controller. Assembly language programming is introduced in the sophomore year, followed by an embedded systems design course, which introduces students to a mixed assembly language and high-level language software design. A final course in the sophomore year covers object-oriented concepts.

A development platform has been designed which is able to support high-level language programming for freshmen as well as embedded systems design in a mixed-language environment. This platform also allows a seamless transition to a fully independent embedded systems design course. The integrated development environment allows complete design in a procedural programming language, assembly language, and object-oriented language without changes in the software or hardware platform.

I. Introduction

Graduates of Electrical Engineering programs nowadays are required to show proficiency in programming languages of various levels, ranging from assembly languages to object-oriented or even dynamic languages[1]. This is very often covered in a sequence of courses in Electrical Engineering programs.

Informal surveys of freshmen in the Electrical Engineering program at Milwaukee School of Engineering (MSOE) show a very diverse background in programming, ranging from no programming experience to almost proficiency in object-oriented programming. Hence the following course sequence was chosen.

Programming using a procedural programming language is introduced in the freshmen year. This course covers all aspects of procedural programming techniques, including algorithm development, elementary data types, standard operations, arithmetic and assignment statements, logical expressions and control constructs, looping techniques, one and two-dimensional subscripted variables, library functions, and user-defined functions. Parallel interfacing techniques and input/output streams are also introduced. This course also introduces the students to the integrated development environment (IDE) common to all programming courses in the EE program. Detailed installation and getting-started documents can be found at the author's web-site[10].

Assembly language programming is introduced in the sophomore year. All microcontroller programming aspects are covered, including structured assembly

language program design, interrupt driven programs, assembly programs using the timer subsystem, ADC, and UART, as well as synchronous and asynchronous communication and standards.

Object-oriented programming is also introduced in the sophomore year. This course covers the object-oriented part of C++, including classes, objects, encapsulation, inheritance, polymorphism, vectors, queues, lists, and pointers. This course builds on knowledge gained in the programming course in the freshmen year.

Finally, an embedded systems design course requires the students to design a mobile robot using an almost identical hardware platform. The software development platform in this course is the same IDE as in the previous courses, enabling the students to develop mobile robotics programs in assembly, C, C++, and mixed-language mode.

The complete course description of all courses can be found at the MSOE web-site.[2]

II. Software development platform

Primary concern among any programming language course is the software development environment or platform. Integrated development environments (IDE) have come a long way from their terminal-based assembler or compiler origin. The major compiler development houses have settled on a semi-standard layout and philosophy of their IDE's. Excellent examples are Visual C++ from Microsoft[3] and the IAR compiler family[4], very similar to the layout shown in Figure 1. These

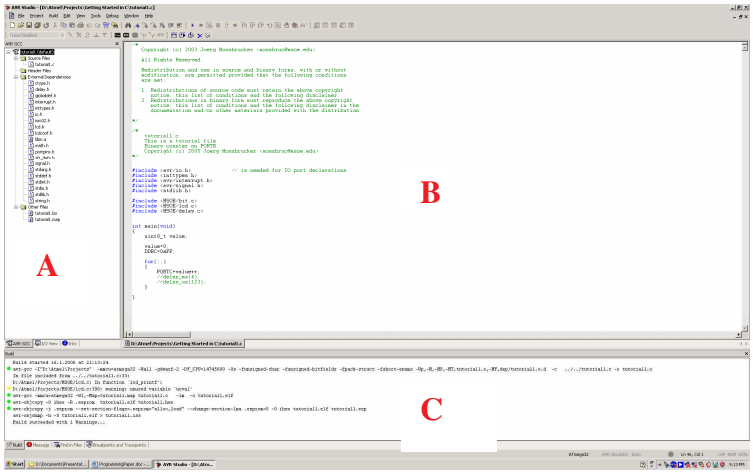

Figure 1: Typical IDE

These IDE's are project-based, showing sources (region A in Figure 1), source file dependencies (region B), and status reports (region C) among other information in a main window. This gives the developer a complete overview of the structure and content of even complex programs with multiple source files.

One major recent advance in the IDE philosophy is the inclusion of plug-in compiler front-ends. This allows a single IDE to cover multiple source codes written in different programming languages completely transparent to the user. The source code editor, compiler, assembler, linker, and simulator are all independent of the programming language of the source file. This allows a single IDE to be used to cover all levels of programming, from assembly languages to object-oriented languages.

Today's compilers can be broadly categorized by their target platform into compilers for personal computers and compilers for embedded systems. Even though the source code editors and compiler front-ends are identical or nearly identical, the compiler back-ends consisting of optimizers, simulators and debuggers vary greatly. Compilers for personal computers in general do not have a simulator incorporated into their IDE. This is mostly due to the fact that PC hardware platforms are not standardized and that debugging is mostly done on a source-code level, rather than on an assembly level. On the other hand, compilers for embedded systems mostly focus on their optimizing strategies and on their ability to simulate the embedded system as accurately as possible. Source-code level debugging is also done, but very often accompanied by assembly level debugging using the build-in simulator. The debugger is very often substituted by a real-time simulator or even hardware emulator connected to the target platform. However, due to limited resources on most embedded systems, compilers for small-to-medium microcontrollers very often do not incorporate front-ends for higher-level programming languages, such as object-oriented or dynamic ones.

In order to use a single IDE for compiling object-oriented sources as well as assembly language sources for embedded systems, an IDE with build-in simulator and debugger as well as a plug-in front-end has been chosen[6]. This IDE has all features required for compiling source codes of different levels for embedded systems, including a sophisticated simulator/emulator. The front-end covers assembly language sources via the build-in assembler, while sources in C, C++, and Ada are covered by a plug-in compiler using gcc[7] in the WinAVR[8] distribution as the compiler, optimizer, and linker. AVRLibC[9] is used for the standard library giving access to all standard library functions of ANSI C99[5]. The compiler allows projects with multiples sources written in different languages. The simulator/emulator allows simultaneous source-code level debugging via the standard dwarf2 interface[11] and assembly-level debugging using Intel-hex-files.

Support functions have been included in the distribution, giving the students access to hardware components. These include input/output streams for almost all input/output devices on the hardware platform. Hence, input devices such as a keyboard/keypad or the serial receiver as well as output devices such as an LCD display or serial transmitter are accessed identically to input/output devices on a personal computer.

III. Hardware platform

The hardware development platform must fulfill several requirements. Elementary input/output devices such as a keyboard/keypad and a textual display must be present for covering basic programming techniques on all levels of programming languages. Support

circuitry for all microcontroller subsystems must be incorporated to allow complete coverage of the embedded system. Finally, circuitry usually found on mobile robotics platforms enable a seamless transition to an embedded systems design course. After a market survey it has been found necessary to develop a custom hardware platform based on a target microcontroller discussed in the previous section. This platform includes the following:

- 8-bit RISC Harvard-architecture microcontroller (Atmel AVR Mega32)
- 32KB Flash program memory
- 2KB SRAM memory
- 1KB EEPROM memory
- Alpha-numeric LCD display (Optrex compatible)
- Keypad and PC-Keyboard interface
- Standard RS232 serial interface
- I^2C (2-wire) serial EEPROM and Real-Time-Clock
- Parallel input and output devices
- Motor drivers

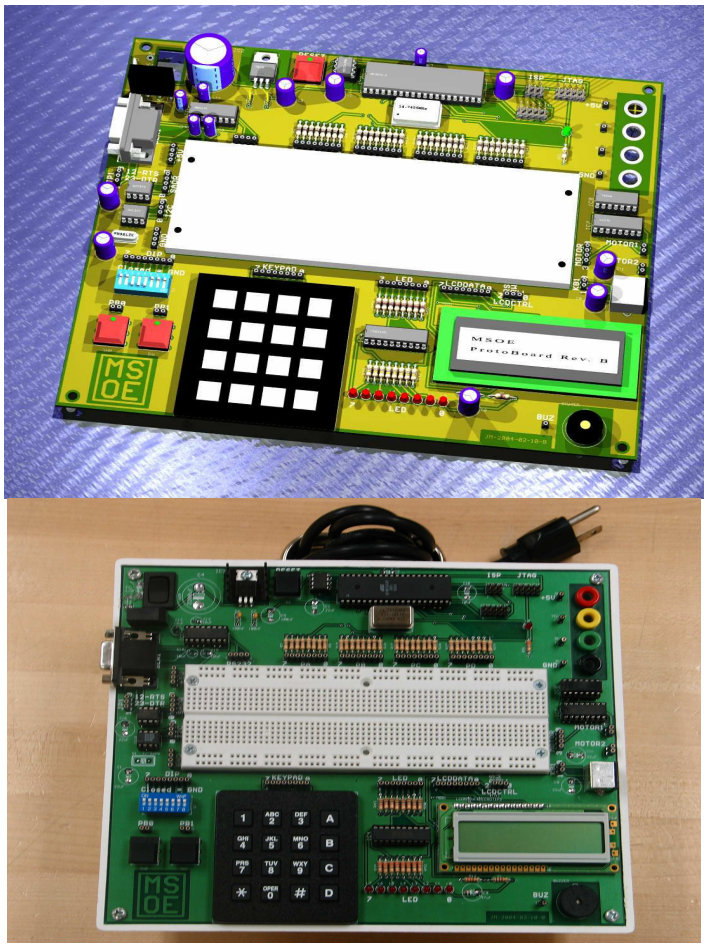

Figure 2: Hardware Development Platform

A 3-D rendering image and a photograph of the final hardware development platform can be seen in Figure 2. A mains power supply has been incorporated into the housing making the development platform independent of external power supplies. Schematics as well as all necessary Gerber files and detailed descriptions of the hardware and software development platform can be found at the author's homepage[10]. The large prototyping area in the center of the hardware development platform allows design and test of complex interfacing and signal conditioning circuitry. This enables usage of the platform in future courses in embedded systems and robotics.

IV. Conclusions

After one complete course sequence it can be summarized that both, the software and hardware development platform fulfill their role as the primary vehicle for teaching students all three levels of programming languages; assembly, procedural, and object-oriented. The IDE chosen is nearly identical in layout and philosophy to the industry leaders, giving the students a "real" hands-on experience. Existing sources primarily intended for compiling on personal computers can be compiled with minimal changes on the embedded system, therefore enabling the instructor to teach all aspects of procedural and object-oriented programming. The rich set of supporting circuitry on the hardware development platform allows complete coverage of all microcontroller subsystems as well as design and test of mobile robotics applications.

Bibliography

1. See the description of the Electrical Engineering discipline at the IEEE USA web-site at: http://www.ieeeusa.org/careers/yourcareer.html
2. See the MSOE web-site for course description at of all courses mentioned in this paper at: http://www.msoe.edu/eecs/cese/courses/curriculum.php?progcode=EE15.1&abet=0
3. See Microsoft's web-site at: http://msdn.microsoft.com/visualc/
4. See the web-site of IAR, one of the industry's leader in IDE's for embedded systems at: http://www.iar.com/
5. See the ISO/IEC9899 standard, available on-line at: http://www.open-std.org/jtc1/sc22/wg14/
6. See the web-site of Atmel Corp. at: http://www.atmel.com/dyn/products/tools.asp?family_id=607
7. See the web-site at: http://www.gcc.gnu.org
8. See the web-site of GNU at: http://www.winavr.sourceforge.net
9. See the web-site of the FSF at: http://www.nongnu.org/avr-libc/
10. See the author's web-site at: http://people.msoe.edu/~mossbruc/
11. See the web-site at: http://www.arm.com/pdfs/TIS-DWARF2.pdf