World Transactions on Engineering and Technology Education
Vol.6, No.1, 2007

Teaching embedded systems using a modular-approach microcontroller training kit

Yao Li

Christchurch Polytechnic Institute of Technology
Christchurch, New Zealand

ABSTRACT: In this article, the author presents a new microcontroller training kit for teaching computer engineering in the Bachelor of Engineering Technology (Electrotechnology) programme (BEngTech) at Christchurch Polytechnic Institute of Technology (CPIT) in Christchurch, New Zealand. The microcontroller training kit was designed and developed in-house with a modular approach. Central to the training kit is the ATmega128 microcontroller, which is an industry relevant microcontroller and has intensive on-chip peripherals, a large amount of flash memory and static RAM, as well as free development software support. The training kit has a built-in JTAG debugger, a USB interface, simple human interfaces and expansion ports so that it can be used not only for teaching in class but also for students' design projects. Modules used to teach computer engineering courses of the BEngTech using the onboard features are discussed. The modules to be used for teaching embedded operating systems are also outlined.

INTRODUCTION

Computer engineering curricula have shifted their focus from computer architecture and CPU design to systems built around highly integrated microcontrollers and embedded systems. To reflect these changes, Christchurch Polytechnic Institute of Technology (CPIT) in Christchurch, New Zealand, uses in-house developed microcontroller training kits to teach embedded systems and microcontrollers to students.

For more than a decade, the 68HC11 training kits have been used for classes and students' design projects at the CPIT. The training kits, designed and developed in-house, use a modular approach, allowing students to plug in various modules, each of which contains hardware to teach the use of various hardware devices and software techniques that are used by designers in industry. Based on these training kits, a tiny real-time operating system was developed and applied to a sport training machine [1][2].

The training kits have been very successful and have provided many years of relatively trouble-free operation in a student environment. Today, these kits are becoming outdated. The 68HC11 microcontroller, developed by Motorola in 1984 and now the legacy to Freescale Semiconductor, is no longer used for new designs being considered as an obsolete microcontroller [3]. The training kits themselves are beginning to age and require more frequent maintenance.

A new 8-bit microcontroller training kit was recently designed and developed at the CPIT. Central to the training kit is the ATmega128 microcontroller, which has intensive on-chip peripherals, a large amount of flash memory (128 Kbytes) and static RAM (4 Kbytes), and free development environment support. The training kit was designed with a modular approach. It has a built-in Joint Test Action Group (JTAG) debugger, a Universal Serial Bus (USB) interface, simple onboard human interfaces and expansion ports so that it can be used not only for teaching in class, but also for students' design projects.

Various modules are used with the training kit for teaching computer engineering in the BEngTech programme at the CPIT. The BEngTech is a three-year degree (technologist level) qualification. It is aligned to the international Sydney Accord [4]. There are three computer engineering courses in the programme: *Computer Engineering I*, *IIA* and *IIB* [5]. *Computer Engineering I* is the compulsory course in the second year. *Computer Engineering IIA* and *IIB* are two courses for students who select computer engineering as their specialisation in the third year.

In this article, the author firstly introduces the new micro-controller training kit, which was developed as a platform of teaching microcontrollers and embedded systems. Secondly, teaching modules based on this training kit to teach computer engineering courses of the BEngTech are discussed. The article concludes with conclusions and future work.

THE AVR 8 MICROCONTROLLER TRAINING KIT

The microcontroller training kit consists of four functional blocks: the AVR microcontroller, the debugger, the USB interface and the I/O module. To make the kit more flexible for students' learning and for their projects, the first three blocks were built into a single module as the daughterboard and the I/O module as the motherboard. The daughterboard is also referred to as the AVR-CPU Module. A block diagram of the training kit is shown in Figure 1.

The AVR Microcontroller

The ATMega128 microcontroller and its support components are the basis of this training kit [6].

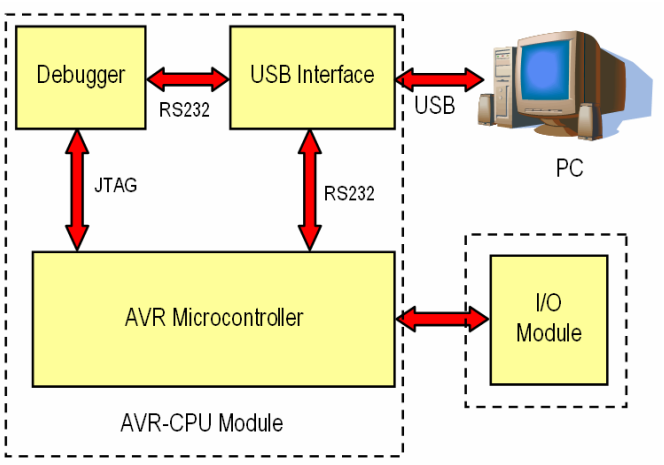

Figure 1: A block diagram of the training kit.

The AVR 8 is a relatively new 8-bit microcontroller compared to other commonly used microcontrollers such as the Intel 8051 and the Microchip PIC [7][8].

The AVR has a Harvard architecture and uses separate memories and buses for programs and data. The program memory space is the on-chip In-System Programmable (ISP) flash. This allows the program memory to be reprogrammed in-system through an SPI serial interface by a conventional non-volatile memory programmer or by an on-chip boot program running in the AVR core.

The AVR CPU has a simple programmer's model. It has thirty-two 8-bit general registers. Six of them can be combined as three 16-bit indirect address register pointers for data/program space addressing. The status register and the stack pointer are located as I/O registers.

The whole AVR family has the same instruction set and the same CPU core with differences in peripheral/RAM/ ROM capabilities: from the Tiny AVR controller ATtiny11 with 1KB flash ROM, no RAM (only the 32 registers) and 8 pins, up to the Mega AVR controller ATmega2560 with 256KB flash ROM, 8KB RAM, 4KB EEPROM, 16 ADC channels of 10 bits each, timers, analogue comparators, JTAG, etc.

There is an open source AVR C compiler suite available based upon the GNU C compiler for Windows called WinAVR [9]. WinAVR is supported by Atmel's freeware professional development software, *AVR Studio 4* [6].

Features such as free development tools and compilers, plus the ability to integrate a low cost in-circuit debugger into the AVR-CPU Module made the AVR an excellent choice for the new training kits. Atmel has shipped over 500 million AVR microcontrollers, making it the world's largest selling 8-bit flash microcontroller [10]. This allows students to develop a popular microcontroller architecture that is widely used in the electronics industry.

When the AVR-CPU Module was designed, the ATMega128 was the most feature-packed AVR device available. The ATMega128 has large built-in ROM (128K bytes) and RAM (4K bytes) capability and also provides a wide range of internal peripherals and seven 8-bit I/O ports (Ports A to G). All of these features have made the ATMega128 an ideal choice for the training kit.

The Debugger

The AVR microcontroller provides an IEEE Standard 1149.1 compliant JTAG test interface for accessing the on-chip debug system and programming. With the JTAG debugger, users can not only program their code into the target AVR but also control and monitor the execution of their code directly.

The debugger is designed to be used with *AVR Studio*, an integrated development environment, allowing the developer to create, debug and program AVR applications within one application. *AVR Studio* has a software simulator and a plug-in architecture to allow the support of external devices like programmers, compilers and development boards. *AVR Studio*'s ease of use and powerful features make it a very popular tool with AVR developers. Integrating the debugger into the AVR-CPU Module gives many advantages to the user. It removes the need for an external debugger device and cables, providing an *all-in-one* solution.

The USB Interface

The USB interface provides the means for the debugger to communicate with the host computer. This is achieved by a serial Universal Synchronous/Asynchronous Receiver and Transmitter (USART) interface on the side of the debugger and USB interface on the side of the computer. The FT2232C developed by FTDI was used to implement the interface [11].

The FT2232C has two RS232 ports. One of them is connected to the debugger and another connected to USART0 of the ATMega128 so that the microcontroller can communicate to the computer via the USB port.

Drivers of the USB interface are available for both Windows 2000 and XP from FTDI. These drivers create a *virtual COM port* in the computer. The controlling application running in the computer communicates with a RS232 device and is not aware that it is actually communicating with the device through a USB connection.

Another feature offered by the FT2232C is the controllable power supply source. It provides the power supply to the entire training kit with 5V of voltage and up to 500mA of current.

The I/O Module

The I/O module was built in a separate printed circuit board. It is the motherboard of the training kit so that the AVR-CPU module can be plugged onto it, as shown in Figure 2.

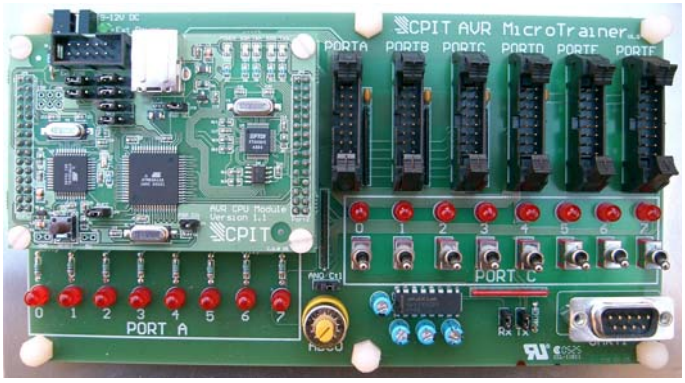

Figure 2: The AVR-CPU and I/O modules.

There are eight toggled switches connected to Port C and 16 LEDs connected to Ports A and C of the ATMega128 microcontroller. These switches and LEDs are simple human interfaces. A variable resistor is connected to one of the ADC channels. All of these onboard I/O peripherals are intended to be used for introductory courses.

The I/O module also provides six I/O port connectors connected to Port A to Port F, respectively. All these port connectors have a standard interface to the existing peripheral modules in the School at the CPIT, such as keypads, seven-segment multiple-digit displays, LCD displays, etc. These can be utilised for advanced courses, especially for course projects or students' design projects.

A serial port connector was designed in the I/O module to make use of the second serial port in the ATMega128.

The microcontroller training kit was designed to be used in a student environment and is housed in a small zip-up carry case. The case provides adequate physical protection by the use of a protective soft cover with a reinforced backing. The complete kit connected to the laptop computer is shown in Figure 3.

Figure 3: The training kit is connected to a laptop computer.

TEACHING MODULES

Based on the modular design and *all-in-one* solution of the training kit, various teaching modules were developed to teach computer engineering in the BEngTech and other programmes. Among the three computer engineering courses in the BEngTech, *Computer Engineering I* (BCEN240) is an introductory course taught in the second year. Upon completion of this course, students will be able to:

- Identify and describe common modern computer systems, architectures, instruction sets and common peripheral components;
- Apply programming techniques in assembly and high-level languages to write control programs for a selected microprocessor/controller.

The prerequisite of this course is *Electronic Principles* (BETR120), where students have learnt Boolean logic, logic signal concepts and basic digital devices.

BCEN240 is taught over 13 teaching weeks with three-hour lectures and a one-hour tutorial each week, plus six four-hour laboratories.

After the architectures of a computer system and the CPU itself are introduced in the lectures, students are expected to explore the microcontroller using introductory modules.

The introductory modules utilise the onboard simple human interfaces: LEDs and toggle switches. Students are expected to set up the training kit by connecting the USB cable to the host computer, installing the drivers if not yet installed and starting up *AVR Studio 4* in the host computer. Students write an assembly language program to carry out binary counting in Port A LEDs. This session is a guided self-exploration for the students so there are lots of interactions between the lecturer and students. Indeed, students usually can get their first program built and run without too much trouble after the assembly program template is introduced. When running their program, they realise that the counting in the output's LEDs is too fast to be observed. At this stage, common debugging tools, such as Single Step and Break Points, are introduced.

After a short discussion among students, they soon figure out the following:

- To implement loops for delay between counts so that the binary counting in the output port can be easily observed;
- To implement subroutines for delay loops so that the source code can be re-used.

When the modified program is built and run again, it crashes. This creates curiosity among the students involved. To find out what has happened, students are guided to use debugging tools. After students have determined that the program crashed when a subroutine tried to return to the calling routine, the use of the stack and the stack pointer for subroutine calling and returning is introduced.

After the introductory module and with more materials studied in the lectures, students are expected to write more assembly language programs, such as *Knight Rider*, *Bar Indicator*, etc, using the onboard input/output interfaces. Students are then required to complete a project implementing an electronic game called *Copy the Lights*. The game displays a random binary number in Port A, waiting for the user to set the toggle switches in Port C. If the input number matches the number in Port A within five seconds, the game outputs another random number; otherwise, the game is over. It is found that writing electronic games stimulates students' interests. Many students add new features to the game.

Starting with assembly language programming enhances the understanding of how the microcontroller works. The focus of assembly language programming is on data transferring so that students can better understand the addressing modes of the CPU. Figure 4 is a diagram summarising all the addressing modes for data transferring, which is discussed in lectures and found to be very useful for assembly language programming.

To progress in pace with teaching in lectures, the next modules are those implemented in the C programming language. The built-in USB interface has two RS232 ports, one of which is connected to USART0 of ATMega128 so that the micro-controller can communicate with the host computer via the USB port. To teach the C language, the standard I/O can be re-directed to USART0. The HyperTerminal in the host computer connected to the virtual COM port can then be used as input/output devices for printf(), scanf() and gets() functions. The header file that redirects the standard I/O to USART0 is provided to students.

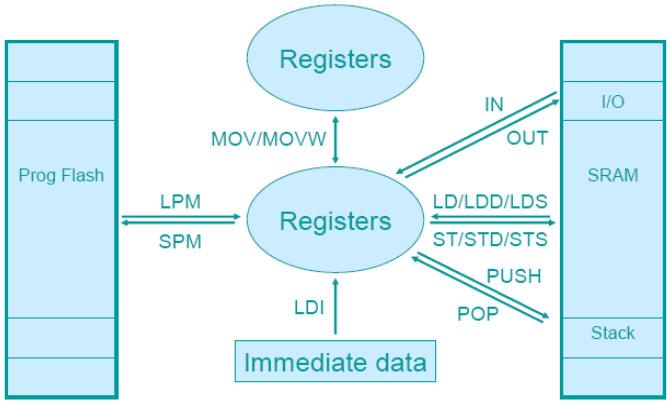

Figure 4: AVR CPU addressing modes for data transferring.

Connected to the host computer through the USB connection, the training kit is a generic and standalone platform (with no extra connections) for teaching the C language. The development software, AVR GCC compiler and *AVR Studio 4*, is freeware, not only available for classrooms, but also for students' home computers. *AVR Studio 4* provides both assembly level and C-source level debugging.

After lectures and laboratory exercises on C data type, input and output, decision-making, looping and functions, students are required to complete a project in the C language; this is another electronic game called the Number of Nim. The game is played by taking turns to remove 1, 2 or 3 counters from a board. The object of the game is to force the opponent to remove the last counter from the board. A random number selection determines if the computer or the player has the first turn. This is a general C program and can be implemented using any C compiler.

In parallel to the project, students continue to use the onboard modules to study the functionality of common on-chip peripherals, such as the timers, USART and ADC. In the ADC module, students rewrite the *Bar Indicator* using the C language in which the output of the Bar Indicator is controlled by the onboard adjustable resistor connected to the ADC channel.

Other on-chip peripherals are not studied in BCEN240. External modules, such as keypad, LCD, MMC adaptor (using SPI) and real-time clock (using I2C), are available and will taught in the third year computer engineering courses of the BEngTech.

CONCLUSIONS AND FUTURE WORK

The AVR microcontroller training kit has been designed specifically as a platform for teaching microcontrollers and embedded systems. It has an integrated JTAG debugger and a USB interface, and is supported by freeware professional development environment. The *all-in-one* solution of the training kit provides onboard modules for teaching computer engineering courses in the BEngTech.

With its modular design, along with the microcontroller's large amount of flash memory and static RAM, the training kit is also a platform for teaching embedded operating systems.

Future work will be to design and develop an embedded MINIX operating system using this kit. MINIX has been designed for educational purpose [12]. Its latest version is also targeted at reliability and security, as well as full functionality for the $100 laptop project [13][14]. MINIX is a UNIX clone, but was designed to be small enough and clean enough for students to understand. The embedded MINIX will maintain this basic design principle and deal with the resource constraint that is common to embedded microcontrollers.

REFERENCES

1. Li, Y. and Wilson, P., PARTOS-11: an efficient real-time operating system for low-cost microcontrollers. *Proc. 1st IEEE Inter. Workshop on Electronic Design, Test, and Applications*, Christchurch, New Zealand, 235-239 (2002).
2. Li, Y., The slack sharing server for embedded microcontrollers. *Proc. 2nd IEEE Inter. Workshop on Electronic Design, Test and Applications*, Perth, Australia, 121-125 (2004).
3. Freescale Semiconductor (2005), http://www.freescale.com
4. Engineers New Zealand, IPENZ, Three Year Engineering Technology Degrees (2006), http://www.ipenz.org.nz/ipenz/Education_Career/accreditation/three_year.cfm
5. Maples, D. and Wilson, P., The Bachelor of Engineering Technology at CPIT. *Proc. Assoc. for Engng. Educ. in Southeast and East Asia and the Pacific Mid-Term Conf. 2004*, Auckland, New Zealand, 48-54 (2004).
6. Atmel (2005), http://www.atmel.com/
7. Intel, MCS® 51/251 Microcontrollers (2005), http://www.intel.com/design/mcs51/
8. Microchip, 8-bit PIC® Microcontrollers (2005), http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=74
9. SourceForge, WinAVR (2005), http://winavr.sourceforge.net/
10. Atmel, Atmel's AVR Microcontroller Ships 500 Million Units (2006), http://www.atmel.com/dyn/corporate/view_detail.asp?ref=&FileName=Ships500M.html&SEC_NAME=Product
11. FTDI Chip, FT2232C Dual USB UART/FIFO IC (2006), http://www.ftdichip.com/Documents/DataSheets/ds2232c_15.pdf
12. Tanenbaum, A.S. and Woodhull, A.S., *Operating Systems Design and Implementation* (3rd edn). Upper Saddle River: Pearson Education (2006).
13. Tanenbaum, A.S., Herder, J.N. and Bos, H., Can we make operating systems reliable and secure? *Computer*, May, 44-51 (2006).
14. MINIX3 (2006), http://www.minix3.org/