Session #: _____

FOREST VS TREES – OUR EXPERIENCE OF TEACHING EMBEDDED SYSTEM

Guoping Wang

Department of Engineering,
Indiana University Purdue University Fort Wayne, Fort Wayne, Indiana 46805
Email: wang@engr.ipfw.edu

1. INTRODUCTION

Embedded computer systems are changing more often than other computing environments since the scope of their application domain is expanding. In the past, embedded system development was largely focused on 8-bit, standalone systems written directly in assembly language. These systems were characterized by slow CPUs (Central Processing Unit), kilobytes of memory, and limited peripheral devices on the chip. Now they are embracing ever-widening application domains to include not only 8-bit standalone systems but real-time, networked, Operating-System based, wireless systems with megabytes of memory and 32-bit CPUs, with very rich peripheral devices such as I/O ports, timers, PWM (Pulse-Width Modulation), UART (Universal Asynchronous Receiver/Transmitter), SPI(Serial Peripheral Interface), I^2C, A/D, D/A, CAN(Controller Area Bus), USB(Universal Serial Bus), Ethernet and on-chip flash memory programming, etc. The datasheet of those CPUs are several hundred pages, or even more than a thousand pages. In the past, the traditional teaching approach was focused on the function illustrations of various peripheral registers such as data register, control registers, status registers, and simple program techniques, etc. However, as there are thousands of peripheral registers on current 32-bit microcontrollers, this approach will not work very well, otherwise you will not see the forest, but only see the details of registers, just like trees in a forest and are overwhelmed by the detailed register functions. In this paper, we shared our experience of teaching embedded system using available industry firmware libraries, industry standards, RTOS (Real-Time Operating System), and RTOS Middleware etc. First, one peripheral (General Purpose Input/Output, for example) is chosen to illustrate some features of the CPU, and then the driver library, industry standard library, operating system, etc. are covered to explain the functions registers, input/output parameters of peripherals, programming techniques using RTOS, etc. This approach not only helps students to see the trees, but also more importantly, it lets them to see the big picture and use them in the embedded system design projects.

2. EMBEDDED MICROCONTROLLER

An embedded system is a special-purpose computer system in which the computer is completely encapsulated by the device it controls. Unlike a general-purpose computer, such as a personal computer, an embedded system performs pre-defined tasks, usually with very specific requirements. Since the system is dedicated to a specific task, design

engineers can optimize it, reducing the size and cost of the product. Embedded systems are often mass-produced, so the cost savings may be multiplied by millions of items.

Some examples of embedded systems include ATMs, cell phones, printers, thermostats, calculators, and videogame consoles. Handheld computers or PDAs are also considered embedded devices because of the nature of their hardware design, even though they are more expandable in software terms. This line of definition continues to blur as devices expand.

A microcontroller (sometimes abbreviated μC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of flash is also often included on chip, as well as a typically small amount of RAM (Read Access Memory). Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

As the heart of an embedded system, the microcontroller has seen significant advancements in the last thirty years, from 8-bit to 32-bit, from very limited on-chip resources to very rich peripherals among which you can find almost everything needed for embedded applications nowadays.

MCS-8051, one example of early 8-bit MCU, was introduced by Intel Corporation in 1981[1]. This MCU has only 128 bytes of RAM, 4K bytes of on-chip ROM (Read Only Memory), two timers, one serial port, and four ports (each 8-bit wide) all on a single chip. Intel's original 8051 versions were popular in the 1980s and early 1990s and enhanced binary compatible derivatives remain popular today.

From 1996, the 8051 family MCU was continued with enhanced 8-bit MCS-151 and the 8/16/32-bit MCS-251 family MCUs. While Intel no longer manufactures the MCS-51, MCS-151 and MCS-251 family, enhanced binary compatible derivatives made by numerous vendors remain popular today. These enhanced MCUs also embed many peripherals on-chip, such as flash RAM, A/D and D/A converters, Real-time Clock, watch dog, even USB on the chip. Those MCS-51 compatible MCUs find popular applications in low-end embedded systems.

In the last ten years, along with the surge of mobile devices, 32-bit ARM processor has been very popular. In 2005, about 98% of all mobile phones sold used at least one ARM processor[2]. The low power consumption of ARM processors has made them very popular today. As of 2013, 37 billion ARM processors have been produced, up from 10 billion in 2008[3]. The ARM architecture is the most widely used architecture in mobile devices, and most popular 32-bit MCU in embedded systems[4]. Today, one of the popular ARM architecture Cortex-M3 chip includes numerous peripherals such as General Purpose I/O ports, timers, PWM, UART, SPI, I^2C, A/D, D/A, CAN, USB, Ethernet and on-chip flash memory programming, etc.

Along with the advancement of MCUs, the teaching pedagogy of embedded system has significantly changed. In the infancy years of embedded system, for the 8-bit 8051 with limited peripherals, usually, assembly language was used in the classroom teaching. Assembly language was used to access registers, control LEDs (Light-emitting Diode),

switches, etc. Students were instructed to understand the bit configurations of all the special register functions (SFR). For example, to control/setup timer, two registers, TCON, and TMOD are needed to be configured. For serial communication configurations, SCON and SBUF registers are used. Since the number of SFRs is limited for MCS-8051, the approach worked very well in early days. With lecture notes and lab practices using assembly language focused on limited 8051 SFRs, students learned how to program/design a simple embedded system.

As time progresses, use of microprocessor-specific assembly-only as the programming language reduces, embedded systems move to C as the embedded programming language of choice. C is the most widely used programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.

As assembly language programs are specific to a processor, assembly language didn't offer portability across systems. To overcome this disadvantage, several high level languages, including C, came up. Some other languages like PLM, Modula-2, Pascal, etc. also came but couldn't find wide acceptance. Amongst those, C got wide acceptance for not only embedded systems, but also for desktop applications. Even though C might have lost its sheen as mainstream language for general purpose applications, it still is having a strong-hold in embedded programming. Due to the wide acceptance of C in the embedded systems, various kinds of support tools like compilers & cross-compilers, ICE, etc. came up and all this facilitated development of embedded systems using C.

In this stage of embedded-system teaching, embedded C language became the dominant language to teach in the classroom. For example, one of the popular books "Embedded C"[5] by Michael J. Pont was widely used in embedded system lecture for 8051-based system. As C language is a 'mid-level', however with 'high-level' features (such as support for functions and modules), and 'low-level' features (such as easy access to hardware via pointers) and it is also very efficient, popular and well understood. C's strengths for embedded system development made it very popular in the embedded system classroom teaching.

In the last ten years, along with the surge of mobile devices, 32-bit ARM processor has become very popular in embedded system design. As more and more peripherals are embedded on one single ARM chip, the volume of the user manual has significantly increased to several hundred, even several thousand pages. It is impossible to cover and explain all the instruction set, peripheral configurations, embedded system programming technique, etc. to the students in the classroom. For example, take ARM Cortex-M3 LPC1768 chip from NXP company as an example, for one UART, there are fifteen registers associated with it[6]. It is not effective to teach all these registers with the old approach.

In the following, our experience of teaching embedded system is presented. After a brief introduction of ARM Cortex M3 processor, we chose one peripheral (General Purpose

I/O) to explain the ARM-feature registers, then the Peripheral Firmware Driver Library, industry standard CMSIS (Cortex Microcontroller Software Interface Standard) from ARM[7] are covered to illustrate the configuration functions, input/output parameters of other main peripherals and programming techniques.

After that, students were taught how to develop Board Support Package (BSP) using available Firmware Driver Library and code embedded system using RTOS and RTOS Middleware. This approach not only helps students to see the trees (the register configurations), but also more importantly, it lets them to see the big picture of embedded system architecture and use them in the embedded system design.

3. OUR EXPERIENCE OF TEACHING EMBEDDED SYSTEM

ECE 485-Embedded Real-Time Operating Systems is a 4-credit hour course which covers the topics of: an introduction to embedded system and real-time operating systems, with an emphasis on embedded system software development, tasks, inter-task communications and synchronization as well as network software. Students who successfully complete this course will have demonstrated:

1. Familiarity with many of the issues involved with embedded systems.
2. Familiarity with key Real-Time Operating System terms and concepts.
3. Ability to program using system calls in a uC/OS-II environment.
4. Ability to program an embedded system with tasks and executive. Understanding and ability to use tools to build an embedded real-time system.
5. Ability to specify, design and implement a small embedded system.
6. Understanding and ability to use tools to build an embedded real-time system.
7. Ability to present design information effectively in the forms of technical reports and oral presentations.

Students should have taken C/C++ language courses, microprocessor course which covers computer organization and assembly language before taking ECE 485 at IPFW (Indiana University Purdue University Fort Wayne). ECE 485 is composed of both lecture and lab components. In the end of the semester, the instructor and students will assess the course outcomes and lab facilities through assessment forms with questions regarding to course outcomes and lab equipment. Starting from fall 2013, 32-bit ARM has become the main CPU used both in lecture and lab sessions.

The ARM Cortex-M3 is a 32-bit microprocessor made by ARM based on the ARMv7 architecture, with a Harvard architecture, i.e. separate code and data buses allowing parallel instruction and data fetches, with three profiles, A for high-end applications, R for real time applications and the microcontroller targeted M profile which is popularly used in embedded system.

LPC1768 is a Cortex-M3 core based MCU from NXP company[8] which we used it in ECE485 teaching. Keil MCB1700 development board[9] is adopted in the lab practices. MCB 1700 board features with 100Mhz ARM Cortex-M3 MCU LPC1768, with on-chip memory 512KB and 64KB RAM, Color QVGA TFT LCD, 10/100 Ethernet port, USB 2.0 full speed support, 2 CAN interfaces, 2 Serial ports, SD/MMC Card Interface, 5-

position joystick and push-button, analog voltage control input, amplifier and speaker, and with 20pin JTAG or 10-pin Cortex debug support. Keil MDK-ARM toolchain software evaluation version [10] can support up to 32KB compiled code which is enough for all the lab practices. In the following, we will cover the outlines of the lectures and lab practices from our experiences.

3.1. Lecture Outlines

The lecture covers the following topics:

1) Introduction to ARM: This lecture briefly introduces ARM MCU history, various versions, features, development toolchains of both open source and commercial ones, etc.
2) ARM Assembly Language: The ARM instruction set and assembly language are covered in this section.
3) Embedded C vs ANSI C: Features and differences of embedded C vs ANSI C are covered.
4) ARM CMSIS: CMSIS is a vendor-independent hardware abstraction layer for the Cortex-M processor series. The CMSIS enables consistent and simple software interfaces to the processor and the peripherals, simplifying software re-use, reducing the learning curve for new microcontroller developers and reducing the time to market for new devices.
5) LPC1768 and General Purpose IO: Using GPIO of LPC1768 as an example, the registration configurations, interrupt control and sample codes to control GPIO are covered.
6) LPC1768 Firmware Driver Library: Instead of the detailed registration configurations of peripherals, only the features and functions of LPC1768 other peripherals, such as, NVIC (Nested Vectored Interrupt Controller), Ethernet Port, USB device/host controller, UART, CAN port, SPI, I^2C, timer, etc. are explained. Then the firmware driver library (CSP Chip Support Package) from NXP [11] is introduced and students learn how to use this library in embedded system programming.
7) MCB 1700 Board Support Package (BSP): BSP development package using LPC1768 CSP is covered in this lecture. Students learn how to develop MCB1700 BSP package using LPC1768 Peripheral Firmware Driver Library. Students were assigned to develop LED, joystick, LCD board drivers.
8) Embedded Real-Time Operating System: Embedded Real-Time Operating system (uc/OS-II) using semaphore, message, timer, kernel, task schedule is introduced and covered in this session.
9) RTOS Middleware: Other RTOS middleware features and their application examples such as, File system, USB, GUI (Graphics User Interface), CAN Driver are introduced. Students are introduced to RTOS middleware and how to use them in their design projects. Since most Middleware are not available due to prohibitive license fees, we covered the concepts, examples of open-source versions, and application examples of those middlewares.

Through these lecture notes, students not only are exposed to the LPC1768 feature, but also they have learned the rich peripherals and how to develop an embedded system with current available firmware driver library, CMSIS industry standard, RTOS in the embedded system design.

3.2. Lab Practices

The lab part is inseparable part in the classroom and it is in alignment with the lecture notes.

Lab01: Introduction to Keil MDK Toolchain. The purpose of this lab is to get students familiar with Keil uVision development tools. Keil uVision is the name of a software package dedicated to the development and testing of a family of microcontrollers based on 8051, 251 and ARM technology from Keil ARM company.

Lab02: C Pointers: The purpose of this lab is to help students further understand the concepts of pointers, array pointers in C language for embedded applications.

Lab03: Introduction on MCB1700: The purpose of this lab is to get students acquainted with MCB1700 board and write a simple program to control LEDs and switches with GPIO.

Lab04: Stopwatch: The purpose of this lab is to let students learn how to develop MCB1700 BSP with the support of LEDs, switches, LCDs through a stopwatch project.

Lab05: AC Control with CMSIS and RTX RTOS System: The purpose of this lab is to let students to get familiar with ARM CMSIS protocol and RTX real-time OS through an AC control system using BSPs from lab04.

Lab06: uC/OS-II and RTOS Middleware: The purpose of this lab is to let students get familiar with uc/OS-II RTOS system and CAN Driver RTOS Middleware.

In this embedded system course, through this organized lecture notes and lab practices, students get exposed to the Cortex-M3 LPC1768 features, rich peripherals, industry-standard CMSIS, Firmware driver library (CSP), Board Support Library, RTOS and RTOS Middleware and they are well prepared with a good foundation in the design of an embedded system.

4. SUMMARY

With the advancement of modern microcontroller, the training of current students to prepare them for the design of embedded system has to adapt. In our practices, through the combination of lecture notes and lab practices, with the industry standard firmware driver library, CMSIS protocol, RTOS, and students-developed board support package, this approach helps students to see the detailed technical side, more importantly, it lets them to see the big picture (Industry standard CMSIS and Firmware Driver Library, RTOS applications) and use them in embedded design projects.

References

[1]. John Wharton: An Introduction to the Intel MCS-51 Single-Chip Microcomputer Family, Application Note AP-69, May 1980, Intel Corporation.

[2]. Tom Krazit, "ARMed for the living room", CNet.com, April 3 2006.

[3]. Dan Grabham, "From a small Acorn to 37 billion chips: ARM's ascent to tech superpower", TechRadar, July 19 2013.

[4]. J. Fitzpatrick, "An interview with Steve Furber", Communications of the ACM, Vol 54, No 5, 2011.

[5]. Michael J. Pont, “Embedded C”, Pearson Education, 2001.

[6]. “An Introduction to the ARM Cortex-M3 Processor”, White Paper, ARM Company, October 2006.

[7]. CMSIS- Cortex Microcontroller Software Interface Standard, available online at http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php, 2013.

[8]. UM10360 LPC17xx User Manual, available online at www.nxp.com/documents/user_manual/UM10360.pdf, 2013.

[9]. MCB1700 Evaluation Board, online at http://www.keil.com/mcb1700/.

[10]. MDK-ARM Microcontroller Development Kit, http://www.keil.com/arm/mdk.asp.

[11]. LPC175x_6x CMSIS-Compliant Standard Peripheral Firmware Driver Library (Keil, IAR, GNU), online at http://www.lpcware.com/content/nxpfile/lpc175x6x-cmsis-compliant-standard-peripheral-firmware-driver-library-keil-iar-gnu.