OUR EXPERIENCE OF TEACHING EMBEDDED REAL-TIME OPERATING SYSTEM

Guoping Wang

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

1. ABSTRACT

In the last twenty years, embedded computer systems have seen significant changes. The embedded microcontrollers (MCU) have shifted from 8-bit slow CPUs, kilobytes of memory and limited peripheral devices to 32-bit, networked, megabytes of memory and rich peripheral devices such as I/Os, timers, PWMs, UART, SPI, CAN, USB, Ethernet, etc. In the last 10 years, 32-bit ARM MCUs have become dominant in embedded system designs. Recently, in ARM Cortex-M series MCUs become very popular both in industry and engineering education. The ARM Cortex-M processor family is a range of scalable and compatible, energy efficient, easy to use processors designed to help developers meet the needs of tomorrow's smart and connected embedded applications. Those demands include delivering more features at a lower cost, increasing connectivity, better code reuse and improved energy efficiency. For Cortex-M MCUs, the development toolchains range from open-source to propriety commercial versions. Many kinds of Real-Time Operating Systems (RTOS) are also available for Cortex-M series, such as CMSIS RTX, Free RTOS, uC/OS-II/III, etc. In this paper, we shared our experiences of teaching ECE 485 – Embedded Real-Time Operating System at Indiana University Purdue University at Fort Wayne (IPFW). Several popular development toolchains are compared for their pros and cons. The features of two RTOSs (CMSIS RTX and uC/OS-II) are described. Project assignments using RTOS are illustrated.

2. INTRODUCTION

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.

Recently, ARM Cortex-M series MCUs have found popular applications both in industry and education. The ARM Cortex-M processor family is a range of scalable and compatible, energy efficient, easy to use processors designed to help developers meet the needs of tomorrow's smart and connected embedded applications. Those demands include delivering more features at a lower cost, increasing connectivity, better code reuse and improved energy efficiency. The Cortex-M development toolchains range from open-source (such as Coocox [5]) to propriety commercial versions (such as Sourcery CodeBench [6], Keil ARM MDK[7], IAR Embedded Workbench[8]). Many kinds of Real-Time Operating Systems (RTOS) are available for Cortex-M series, such as CMSIS RTX [9], Free RTOS[10], uC/OS-II/III[11], etc. In the following, we shared our experiences of teaching ECE 485 – Embedded Real-Time Operating System at IPFW. Several popular development toolchains are compared for their pros and cons. The features of two RTOSs (CMSIS RTX and uC/OS-II) are described. Student project assignments using RTOS are illustrated.

3. ARM CORTEX-M DEVELOPMENT TOOLCHAINS and RTOS

In this section, three popular Cortex-M development toolchains (Coocox, mbed.org[12] and Keil MDK-ARM) are described and compared.

Coocox is a free and highly-integrated software development environment for ARM Cortex MCU based MCU, which includes all the tools necessary to develop high-quality software solutions in a timely and cost effective manner. It integrates CoBuilder and CoDebugger for simplicity and ease of use. The Coocox uses open-source GNU Compiler for ARM, which is the entry level C/C++ compiler for ARM. It features:

- Free to use
- Full functional IDE
- Component-oriented development platform
- Internet-based, efficient integration of network resources
- Integrates RTOS CoOS
- Peripheral registers

However, compared to propriety toolchains, it has the following limitations:

- Limited devices supported: Only some devices are supported and some recent Cortex series MCU are not supported by Coocox toolchain.
- Very limited debug features: The debug feature only supports entry-level debug features such as run, step, breakpoint, etc.
- Not easy to setup and limited debug hardware support.
- Emulator not included.

Unlike most development kits, the mbed doesn't include a compiler or IDE that reside in a PC. Instead, developers use an online C/C++ compiler and mbed libraries. That approach gives them the flexibility to move from computer to computer and still have tools available. The compiler and documents exist on the mbed.org website. The information of developers from the mbed community is very helpful. Mbed is being developed by ARM, its partners and the contributions of the global ARM mbed development community. The mbed platform provides free software libraries, hardware designs and online tools for professional rapid prototyping of products based on ARM microcontrollers.

The platform includes a standards-based C/C++ SDK, a microcontroller HDK and supported development boards, an online compiler and online developer collaboration tools.
It features:

- The mbed compiler provides a lightweight online C/C++ IDE and ARM CC C/C++ compiler to let your quickly write your programs, compile and download them to run on mbed MCU.
- It is a web app and you can log in from anywhere and carry on where you left off, and you are free to work on any PC.
- RTOS supported for mbed MCU board from mbed.org
- The limitations of mbed toolchains are:
- You must use mbed.org compatible boards for design, thus, it may not be suitable for custom build board developments.
- Due to its online features, limited debug features are only supported.

The Keil MDK-ARM is a complete software development environment for Cortex™-M, Cortex-R4, ARM7™ and ARM9™ processor-based devices. MDK-ARM is specifically designed for microcontroller applications, it is easy to learn and use, yet powerful enough for the most demanding embedded applications.

It features:

- Complete support for Cortex-M, Cortex-R4, ARM7, and ARM9 devices
- Industry-leading ARM C/C++ Compilation Toolchain
- µVision4 IDE, debugger, and simulation environment
- Keil RTX deterministic, small footprint real-time operating system (with source code)
- TCP/IP Networking Suite offers multiple protocols and various applications
- USB Device and USB Host stacks are provided with standard driver classes
- Complete GUI Library for embedded systems with graphical user interfaces
- Complete Code Coverage information about your program's execution
- Execution Profiler and Performance Analyzer enable program optimization
- Numerous example projects help you quickly become familiar with MDK-ARM's powerful, built-in features
- CMSIS Cortex Microcontoller Software Interface Standard compliant

The limitations: The license fee is a little expensive ($5000 per seat).

In ECE 485 – embedded RTOS at IPFW, Keil MDK-ARM toolchain is used through the donation of ARM University Program. With the support of MCB1700 development board, Keil MDK-ARM suite provides students a great opportunity to be exposed to industrial standard tools and hands-on projects.

Next, two RTOSs are illustrated which are used in ECE 485 – CMSIS RTX and uC/OS-II.

The Keil RTX (Real Time eXecutive) is a royalty-free deterministic RTOS designed for ARM and Cortex- M devices. It is one of the components of RL-ARM, the RealView Real-Time Library (RL-ARM). RTX and its source code are available in all MDK-ARM Professional Editions.

RTX allows one to create programs that simultaneously perform multiple functions (or tasks, statically created processes) and helps to create applications which are better structured and more easily maintained. Tasks can be assigned execution priorities. The RTX kernel uses the execution priorities to select the next task to run (preemptive scheduling). It provides additional functions for inter-task communication, memory management and peripheral management.

The main features of RTX include:

- Royalty-free, deterministic RTOS with source code
- Flexible Scheduling: round-robin, pre-emptive, and collaborative
- High-Speed real-time operation with low interrupt latency
- Small footprint for resource constrained systems
- Unlimited number of tasks each with 254 priority levels
- Unlimited number of mailboxes, semaphores, mutex, and timers
- Support for multithreading and thread-safe operation
- Kernel aware debug support in MDK-ARM
- Dialog-based setup using µVision Configuration Wizard

MicroC/OS-II (commonly termed as µC/OS-II or uC/OS-II), is the acronym for Micro-Controller Operating Systems Version 2. It is a priority-based pre-emptive real-time multitasking operating system kernel for microprocessors, written mainly in the C programming language.

Its features are:

- It is a very small real-time kernel.
- Memory footprint is about 20KB for a fully functional kernel.
- Source code is written mostly in ANSI C.
- Highly portable, ROMable, very scalable, preemptive real-time, deterministic, multitasking kernel.
- It can manage up to 64 tasks (56 user tasks available).
- It has connectivity with µC/GUI and µC/FS (GUI and File Systems for µC/OS II).
- It is ported to more than 100 microprocessors and microcontrollers.
- It is simple to use and simple to implement but very effective compared to the price/performance ratio.
- It supports all type of processors from 8-bit to 64-bit.

In our teaching of ECE 485, both CMSIS RTX and uC/OS-II are used in the classroom and labs. The main features of CMSIS RTX are illustrated with example projects, functional reviews, etc. Students are assigned two design projects using CMSIS RTX RTOS in their labs. For uC/OS-II RTOS, because of the source code availability and a very well-written uC/OS-II book *µC/OS The Real-Time Kernel* by Jean J. Labrosse[11], the developer of uC/OS-II, it has been used to explain the key concepts of RTOS Kernel structure, task management, time management, intertask communication (semaphore, mutex, mailbox and queue), and memory management, etc. Students are assigned with two design projects using uC/OS-II. The detailed descriptions of these projects are as follows.

- RTOS Project 1: Students are assigned to design a stopwatch project using CMSIS RTX. There are two buttons, START/STOP and RUN. The stopwatch time in minutes:seconds is displayed on MCB1700 LCD screen.
- RTOS Project 2: Students are assigned to design an air condition control system using CMSIS RTX. Four switches are used to control the air condition system and one temperature sensor as input (from the potentiometer) which is connected to analog channel 2 input of LPC1768. Two LEDs indicate the status of the AC compressor (P2.2) and fan (P2.3). When the LED is turned on, the AC or fan is on. LED P1.28 toggles every 500 ms for power indicator. The ADC converter precision is a 12-bit unsigned number from the potentiometer. In this project, 0x000 stands for 0 F, and 0xFFF stands for 127F. Simply the 12-bit number is shifted 5 bits to the right, then you can get the temperature reading. For example, for the AD converted number 0x789, 0x789>>5 = 0x3C = 60 F.
- RTOS Project 3: Students are assigned to write a simple RTOS system using uC/OS-II. There are four tasks in this system.
  - Task 1: Toggle LED P1.28 every 500 ms
  - Task 2: Left/right shift the LEDs using joystick left/right keys
  - Task 3: Stopwatch, controlled by the up/down keys of the joystick.
  - Task 4: Reading the potentiometer analog inputs of ADC channel 2 and display the value on the LCDs.
- RTOS Project 4: Students have the choice to use either CMSIS RTX or uC/OS-II to design the game Tic-Tac-Toe.

  - They must use a keyboard or a joystick to control the game. Students can use either the keyboard of the host PC via UART or connect directly a different keyboard to the board's USB connector. Or students can play the game directly use joystick. It is up to students to decide which way they want to work with the keyboard or joystick (if using the USB, students will have to bring their own separate keyboard and figure out how to use it via USB as it will not be covered in the labs).
  - It is up to students to specify how exactly players must use the keyboard or joystick to play a game. It is however required that their implementation uses the key “N” or INT0 button, which is pressed at any time, would start a NEW game irrespective of whether a game is being played currently.
  - After the game starts, within 5 seconds, if a player doesn’t press any key (or joystick), that player will lose his/her turn. The 5-second counting down is displayed on the screen for each player when it is his turn. The time passed by since the game starts is also displayed on the screen.
  - The design must display all the actions and the current status of a game on the LCD display of the MCB1700 board. The status of the game is displayed using O's and X's (or with different color blocks) within a 3x3 table.

- At the end of each game, the game must print on the LCD display, under the 3x3 table, one of these messages: "Player A wins!" or "Player B wins!" or "Tie!" depending of the outcome of the game.
- At the end of each game the game must generate a simple sound that it should play through the speaker of the board (this is something that students will have to figure out on their own)

Through the practice of these design projects, students will have very sold understanding of RTOS using CMSIS RTS and uC/OS-II and also get familiar with industrial-standard Keil MDK ARM development suite.

4. SUMMARY

In this paper, we shared our experience in teaching ECE 485 – embedded Real-Time Operating System at IPFW. Three popular Cortex-M development toolchains are compared. Coocox provides a royalty-free development suite with IDE, CoBuilder, CoFlash and CoDebugger and can be configured using GCC ARM with some limitations. The online-based mbed toolchain provides an online IDE, compiler, debugger toolchain with the support ARM and its partners, however, it can only support its own mbed MCU development boards. For Keil MDK ARM, it is easy to learn and use, yet powerful enough for the most demanding embedded applications, however, it comes with a very high license fee. The features of two RTOSs -- CMSIS RTX and uC/OS-II are illustrated. Design projects using them are described in details. Our experiences will be helpful to educators who teach similar courses in other colleges.

Reference

[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]. CooCox Free/Open ARM Cortex MCU Development Tools, http://www.coocox.org

[6]. Sourcery CodeBench, http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/overview/

[7]. Keil MDK-ARM Suite, http://www.keil.com/arm/mdk.asp

[8]. IAR Embedded Workbench, http://www.iar.com/Products/IAR-Embedded-Workbench/

[9]. CMSIS-RTOS RTX, http://www.keil.com/pack/doc/cmsis_rtx/index.html

[10]. Free RTOS, http://www.freertos.org/

[11]. MicroC/OS-II The Real-Time Kernel, 2nd edition, by Jean J. Labrosse, CMP Books, 2002, ISBN No: 1-57820-103-9

[12]. ARM Mbed, http://www.mbed.org

[13]. Keil MCB1700 Board, http://www.keil.com/mcb1700/