Proceedings,
The 2013 ASEE North Midwest Section Conference,
October 17-18, 2013, Fargo, North Dakota.

ASEE-NMWSC2013-0039

Teaching the ARM Microcontroller to Keep Up with the Embedded Industry Technology Change

Department of Electrical, Computer Engineering and Technology
Minnesota State University, Mankato, MN 56001
Han-Way Huang, han-way.huang@mnsu.edu
Nannan He, nannan.he@mnsu.edu

Abstract

The widespread use of mobile devices in the last decade has changed the embedded system industry. Mobile devices require the embedded microcontroller to have high performance and low power. The winner is the ARM-based processor. In 2011, 7.9 billion ARM processors were shipped. More than 95% of mobile phones, 90% of hard drive controllers, 40% digital TVs and set-top boxes, 15% microcontrollers, and 20% mobile computers are using the ARM processors.

To keep up with the embedded industry change, we have updated the contents of our microcontroller courses with the goal to keep up with the technology change and make our graduates more marketable. We have taught the ARM Cortext-M4 MCU in our second microcontroller course and plan to also teach the simpler version of the ARM Cortex-M MCU in our first microcontroller course.

Three major issues must be addressed in order to teach a new microcontroller. First, we need to choose an appropriate Cortex-M4 demo board for students to perform laboratory experiments and design projects. Second, we need to find an easy-to-use and affordable IDE software for students to develop and debug their program. Third, we need to find an appropriate textbook or prepare lecture notes. This paper presents our experience of teaching the ARM Cortex-M4 microcontroller in our second microprocessor course of our computer engineering program.

A brief history of ARM

In 1983, the British company **Acorn** saw the need to move beyond 8-bit CPU architecture and a project to invent a new CPU instruction set was started. In April 1985, the first 32-bit ARM processor implemented in the 3 m process came to life. It was a 32-bit RISC machine built with only 25,000 transistors. This processor was used as the processor of the Acorn's multimedia PC. A later version of the ARM processor was selected by Apple as the processor for its Newton product line. In 1990, a joint venture **Advanced RISC Machine** (**ARM**) between Acorn, Apple, and VLSI Technology was formed. The goal of this company is to design low-cost, low-power,

and high-performance RISC chips. Instead of manufacturing and selling its own chips, ARM licenses its processor architectures to any interested parties.

The decade of 2000s saw the boom of the cellular phones and mobile devices (e.g., iPOD). The nature of the cell phones and mobile devices requires the processor of the device to consume as little power as possible and also has enough performance to execute appropriate applications. Apparently, the ARM processor met this requirement perfectly. In addition, the ARM processors are available from multiple vendors, which guarantee the cell phone and mobile device vendors that they can always find a supplier for their processor of choice. Within a couple of years, the ARM processor dominates the cell-phone and mobile device markets. The applications of ARM processors also quickly expand to other areas such as hard drive controllers, HDTVs, set-top boxes, and the traditional microcontroller (i.e., embedded system) market.

The ARM Cortex-M core

The ARM Cortex-M is a group of 32-bit RISC ARM processor cores intended for microcontroller applications, and consists of the Cortex-M0, Cortex-M0+, Cortex-M1, Cortex-M3, and Cortex-M4. This group of processors implements a subset or the whole Thumb-2 instruction set.

The **Cortex-M0** implements a subset of the Thumb2 instruction set (50 16-bit and 6 32-bit instructions) using a 3-stage pipeline. The core of the Cortex-M0 is so small that it can be implemented in 12000 gates. Combining the low gate count and the supported sleep modes, the Cortex-M0 has a power consumption of 4 μW/MHz when implemented using the 40 nm CMOS technology. Cortex-M0 is intended to be a replacement for the 8- or 16-bit microcontrollers for its high code density, low cost, and low power consumption.

The **Cortex-M0+** supports the same instructions as does the Cortex-M0 and is implemented using a 2-stage pipeline. When implemented using the same 40 nm CMOS technology, the Cortex-M0+ consumes a power of 3 μW/MHz. Like the Cortex-M0, the Cortex-M0+ is also a perfect replacement for the 8- or 16-bit microcontrollers for the same reason.

The **Cortex-M1** is designed for implementation in FPGAs and has been licensed by Actel, Altera, and Xilinx. Cortex-M1 implements the same set of instructions as do the Cortex-M0 and Cortex-M0+. This approach allows the designer to add circuits that implement the special algorithms required by the application but are not available in off-the-shelf microcontrollers. Using the 65 nm CMOS technology, Cortex-M1 can run at 200 MHz.

The **Cortex-M3** is designed for applications that require real-time performance and low power consumption. The target applications of the Cortex-M3 include microcontroller, automotive body

system, industrial control systems, and wireless networking and sensors. It implements the whole Thumb2 instruction set using a 3-stage pipeline. The Cortex-M3 achieves the performance of 1.25 DMIPS/MHz. When implemented in the 40 nm CMOS technology, the Cortex-M3 core occupies an area of 0.03 mm^2 and consumes a power of 7 μW/MHz.

The **Cortex-M4** is designed for markets that require a blend of digital signal processing (DSP) and control capability. The target applications include motor control, automotive, power management, embedded audio and industrial automation markets. In addition to executing the complete Thumb2 instruction set, the Cortex-M4 also supports a DSP instruction set and may optional implements a single-precision floating-point instruction set. All instructions are executed using a 3-stage pipeline. When implemented using the 40 nm CMOS technology, the Cortex-M4 consumes a power of 8 μW/MHz and achieve a performance of 1.25~1.52 DMIPS/MHz.

Cortex microcontroller software interface standard (CMSIS)

Although the licensees of Cortex-M4 vendors are free to add their peripherals to differentiate their products from each other, they have a common CPU core which contains the same core peripherals. The core peripherals include the nested vectored interrupt controller (NVIC), system control block (SCB), system timer (SysTic), and memory protection unit (MPU). Because different vendors may choose to use different way to name the registers of peripherals, it will be hard to convert the software written for one vendor to another.

In order to simplify the conversion and makes the life of real-time operating system (RTOS) and middleware vendors easier, ARM Inc. created the Cortex Microcontroller Software Interface Standard (CMSIS) [1]. Microcontroller is often abbreviated as MCU. This standard defines:

- A common way to access peripheral registers and exception vectors.
- The names of the registers of the core peripherals and the core exception vectors.
- A device-independent interface for real-time operating system (RTOS) kernels, including a debug channel.

CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M4 processor. The CMSIS simplifies software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. In the most general sense, **middleware** is computer software that provides services to user software applications beyond those available from the operating system.

Which Cortex-M microcontroller to teach?

Since there are five versions of the Cortex-M microcontroller, we need to choose one or two versions of them to teach. A Cortex-M microcontroller manufacturer tend to add fewer and simpler peripherals to less powerful processors and add more and more sophisticated peripherals to more powerful processors. This has been the practice for microcontroller vendors to differentiate their products. In addition, the Cortex-M4 implements the most instructions among all the Cortex-M processors. It is logical to choose the Cortex-M4 as the target microcontroller to be taught in microcontroller classes. By teaching the Cortex-M4, the students can learn the most extensive peripherals among all of the Cortex-M microcontrollers. They can easily migrate to other less powerful Cortex-M processors when carrying out the actual product design.

Development tools for teaching the Cortex-M microcontrollers

At the time of this writing, there are six major Cortex-M4 microcontroller vendors in the world: Atmel, Freescale, Infineon, NXP, ST, and TI. The major consideration for selecting the Cortex-M4 device from these six companies to teach is the availability of software and hardware development tools. We need both software and hardware tools to teach a course on the Cortex-M4 microcontroller. Software tools include text editor, assembler, compiler, linker, project manager, simulator, debugger, and device drivers. These tools are often integrated into a single package called the **integrated development environment** (**IDE**). Commercial IDEs are expensive.

There are many IDEs for the ARM Cortex-M4 microcontrollers. Some Cortex-M4 microcontroller vendors provide IDEs for free. For example, Atmel and Infineon have free IDEs for their users. Freescale provides a special version of its Code Composer IDE for free to its users. Commercial IDEs usually have demo versions for new users to learn their features. These IDEs have a size limit (usually 32 kB). The user can only debug a program no larger than certain size. For an introductory microcontroller course, this size limit is generally not a problem. A list of commercial IDE vendors can be found from ARM's web site (www.arm.com). IDEs from Cortex-M4 microcontroller vendors are listed in Table 1.

Table 1 IDE provided by Cortex-M4 microcontroller vendor

Cortex-M4 vendor	IDE name	Comment
Atmel	Atmel Studio	Free of charge to users
Freescale	Code Warrior	A special edition is free to end user
Infineon	Dave	Free of charge to users
NXP	none	Users of NXP chips are expected to use third party IDE tools
ST	none	Users of ST chips are expected to use third party IDE tools
TI	Code Composer	Not free

The most important hardware tools for teaching the Cortex-M4 microcontroller are demo boards and debug adapters. The debug adapter connects the demo board to the PC via an USB cable. Some of the demo boards may have an on-board debug adapter already. All Cortex-M4 vendors provide demo boards for evaluating their Cortex-M4 products. Some of them are feature-rich demo boards whereas others are bare kits. The Cortex-M4 demo boards provided by the Cortex-M4 microcontroller vendors are listed in Table 2.

Table 2 Demo boards provided by Cortex-M4 microcontroller vendors

Cortex-M4 vendor	Demo board
Atmel	SAM4E-EK, SAM4L-EK, SAM4S-EK, SAM4L-Xplained-Pro, SAM4S-Xplained-Pro, SAM4N-Xplained-Pro
Freescale	TOWER system
Infineon	Hexagon application kit, XMC4500 Relax Lite kit
NXP	LPC4330-Xplore bundle
ST	STM3240G-EVAL, STM3241G-EVAL, STM32437I-EVAL, STM32303C-EVAL, STM32373C-EVAL, STM32F4Discovery, STM32F3Discovery
TI	TM4C123G Development kit, TM4C123G LaunchPad

The SAM4E-EK and SAM4S-EK from Atmel are feature-rich demo boards and need a debug adapter (for example, SAM-ICE) to interface with a PC to perform program debugging. Most of the SAM4E and SAM4S peripheral functions can be evaluated with these demo boards. SAM4L-Xplained-Pro, SAM4N-Xplained-Pro, and SAM4S-Xplained-Pro are less expensive demo kits with segmented LCD, light and temperature sensors, OLED, LEDs, virtual COM port, and prototype area. I/O signals are made available via four connectors. Debug adapter has been built into the Xplained-Pro kits. Atmel university program provides 50% discount on these kits to its members.

The Freescale **TOWER system** consists of two parallel boards with four pairs of slots, which allow the user to insert the MCU module and other I/O modules to them. Debug adapter has been added to the MCU module.

Each of the Infineon's **Hexagon application packages** consists of a microcontroller board and several I/O boards. Up to three I/O boards can be connected to the MCU board at the same time.

The **Xplore board** from NXP is an inexpensive demo board that provides an SPI flash memory, Ethernet connector, audio codex and audio jacks, micro SD card slot, and on-board debug adapter. Unused I/O signals are made available to the user via a header.

The first five demo boards from ST in Table 2 are feature rich-demo boards with built-in debug adapter. They allow the user to test all of the Cortex-M4 peripheral functions. The STM32F4Discovery and STM32F3Discovery are bare kits but have on-board debug adapter. The

unused I/O signals are made available to the user via headers on two sides of the kit. These two boards cost less than $20. Expansion boards and additional expansion peripheral kits are available for the STM32F4Discovery and STM32F3Discovery kits from ST and third party (from eBay).

The TM4C123G development board from TI has an on-board debug adapter, a color OLED, microCD card slot, 3.0 V reference, 3-axis accelerometer, and five buttons. All I/O signals are brought out to two headers. The TM4C123 LaunchPad is a low cost bare kit with an on-board debug adapter. All I/O signals are also made available to the user via headers.

The demo boards made by the Cortex-M4 vendors are designed for system designers rather than microcontroller learners. Some of the peripheral devices that come with the boards are not be suitable for the introductory microcontroller courses. There are also third party demo boards for Cortex-M4 microcontrollers. Keil, IAR, Hi-Tech, Raisonance and others all sell Cortex-M4 demo boards. Some Cortex-M4 demo boards are also available from eBay.

Textbooks for Cortex-M4

Because the Cortex-M4 microcontroller was still a relatively new microcontroller, there are no textbook dedicated to it. To teach this microcontroller, the instructor will need to prepare lecture notes by adapting Cortex-M4 datasheets and reference manuals supplied by the vendor.

Experience on teaching the Atmel SAM4L Cortex-M4 microcontroller

We have followed the development of Cortex-M processors in the last four years and concluded that we should also teach this microcontroller. We have attended many seminars held by Cortex-M processors vendors including Atmel, NXP, and ST to find out the vendors' product plan, availability of IDEs, demo boards, and debug adapters.

After comparing the availability of free IDEs, functionality and pricing of demo boards, we decide to teach the Cortex-M4 from Atmel. We have two required courses on microcontrollers in our computer engineering program. The first course teaches microcontrollers in assembly language and the second course teaches microcontrollers using the C language. We decide to introduce the ARM Cortex-M4 microcontroller to the second microcontroller course first and then decide whether we should also teach the Cortex-M microcontroller in the first microcontroller course.

At the time of this writing, Atmel has four Cortex-M4 product subfamilies: SAM4E, SAM4L, SAM4N, and SAM4S. SAM4E is the only subfamily that implements the Thumb2, the DSP, and the single floating-point instruction set. The other three subfamilies do not implement the

floating-point instruction set. Atmel introduced the SAM4S first, followed by the SAM4L. We compared the features of the SAM4S and SAM4L and the availability demo kits and decided to teach the SAM4L. The SAM4L[2] adopts a low power design and implements many on-chip peripheral functions: LCD controller, capacitive touch module, advanced encryption/decryption module (AES), peripheral direct memory access (DMA), peripheral event system (PEVC), 12-bit ADC, 12-bit DAC, watchdog timer, asynchronous timer (AST), frequency meter (FREQM), full-speed USB 2.0, USART, SPI, TWI (same as I^2C), 6-channel 16-bit timer, audio bitstream DAC (ABDACB), real-time clock (RTC), random number generator, and inter-IC sound controller (IISC). The **peripheral event system** is unique in that it allows the **event** of one peripheral to trigger the operation of another peripheral. An event refers to the state change (e.g., ADC completion, timer time-out, etc) of a peripheral module. For example, we can use the timer overflow to start an A/D conversion. Without the event system, the user will need to use interrupt to achieve the same goal, which would involve some CPU overhead. DMA is commonly used to improve the data transfer efficiency in a computer.

We have been using the Atmel Studio in teaching the Atmel 8-bit AVR microcontroller for several years. It is natural for us to continue to use it in teaching the SAM4L. The Atmel Studio is free and easy to use. It provides the Atmel Software Framework (ASF) which includes many utility functions for the Cortex-M peripherals and numerous example projects to be run on Cortex-M demo kits made by Atmel. Atmel Studio uses **project** to manage all programming efforts. The user can create a project, enter programs in assembly or C or C++, build the project, download the project onto a demo board, and carry out the debugging activities all in the same environment without quitting any program. Common debugging commands such as setting breakpoint, setting watch list, executing program to cursor position, and so on, are all supported.

We first used the SAM4L-EK kit to do the lab assignment. When the ATSAM4L-XSTK starter bundle was introduced, we switched to it. This bundle consists of the following kits:

- SAM4L Xplained Pro kit. This kits includes a SAM4LC4 microcontroller, a reset button, a user push button, one user LED, debug adapter (via an USB connector), 4 Xplained Pro extension headers (makes I/O pins available to the user), and provide a virtual COM port via an USB connector. The kit use a 12-MHz and a 32-kHz crystal to generate all the clock signals used in the kit.
- I/O1 Xplained Pro kit. This kit includes a microSD card holder via an SPI interface, a PWM-controlled LED, a light sensor, a temperature sensor with I^2C interface.
- OLED1 Xplained Pro kit. This kit includes an OLED with a 128x32 resolution via the SPI interface, 3 LEDs, and 3 push buttons.
- SLCD1 Xplained Pro kit. This kit is a 96-segment LCD display.
- PROTO1 Xplained Pro kit. This is a prototype board with 10x20, 100-mil pitch prototype area.

Lecture notes are adapted from SAM4L datasheets and 8- and 16-bit microcontroller textbooks [3,4,5]. The lecture notes are prepared using the Microsoft power-point and cover the following topics:

- History of ARM Processor
- C programming for Cortex-M4 microcontroller
- Parallel I/O and DMA Transfer
- Exception processing
- Clock generation
- Event system and Timer functions
- SPI and peripheral interfacing
- USART and peripheral interfacing
- TWI and peripheral interfacing
- A/D and D/A conversion
- CAN communications

Programming Cortex-M4 in C language

The first step in programming the Cortex-M4 in C is to find out how to access the Cortex-M4 peripheral registers and their individual bits. The Cortex-M4 device header file provides the definitions for all interrupt vectors, peripheral modules, register address mappings, and bit mappings. All of the Atmel Cortex-M4 device header files are compatible with the CMSIS standard. Atmel Studio allows the user to use two methods to access peripheral registers: **hierarchical** and **flattened** approach.

To support hierarchical access to registers, the header file **SAM4L.h** defines a pointer (a name) to each peripheral module so that the user can use the pointer to the module to access registers within that module. For example, the pointer to the parallel I/O port 0 is PIO0 and the name of the output value register is GPIO_OVR. The following statement outputs the value 0x0000AA55 to the parallel port 0:

```
PIO0->GPIO_OVR = 0x0000AA55;
```

This method is called **hierarchical approach** because the access goes through the module first.

In the **flattened approach**, the register name consists of three parts connected by two underscore characters:

- Keyword REG
- Module name
- Register name and module number (if there are two or more identical modules)

For example, the output value register associated with I/O port 0 is defined as **REG_GPIO_OVR0**. Using this approach, we can assign the value 0x0000AA55 to I/O port 0 as follows:

```
REG_GPIO_OVR0 = 0x0000AA55;
```

The REG_GPIO_OVRS0 is a register that allows the user to set individual bit of the REG_GPIO_OVR0 register by writing a bit mask to it. The header file SAM4L.h allows the user to refer to the bit mask (2^0~2^{31}) by name. For example, the bit mask definition of bit 3, 4, and 5 are defined as follows:

```
#define   GPIO_OVRS_P3      (0x1u << 3)
#define   GPIO_OVRS_P4      (0x1u << 4)
#define   GPIO_OVRS_P5      (0x1u << 5)
```

The following statement sets bit 3, 4, and 5 of the REG_GPIO_OVR register to 1, 1, and 1:

```
REG_GPIO_OVRS0 = (GPIO_OVRS_P3 | GPIO_OVRS_P4 | GPIO_OVRS_P5);
```

After learning how to access peripheral registers and their individual bits, programming task becomes straightforward.

A screen shot of the project that uses external interrupt 5 to toggle the LED driven by the PC10 pin is shown in Figure 1. The PC03 pin is configured to be the external interrupt 5. Whenever the user press the button connected to the PC03 pin, the LED driven by the PC10 pin is toggled.

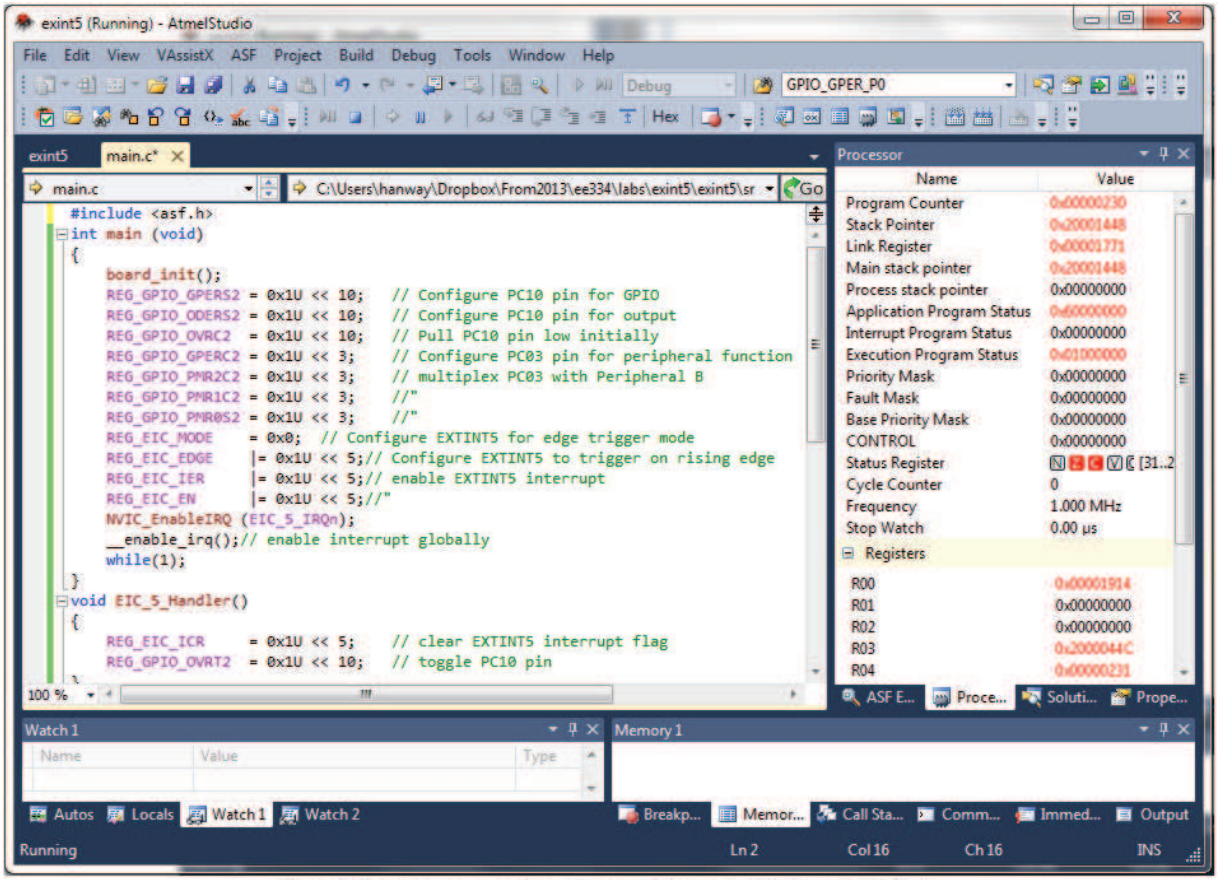

Figure 1. Project that uses external interrupt 5 to toggle LED driven by PC10 pin

Teaching the 32-bit Cortex-M microcontroller is not much different from teaching an 8-bit or 16-bit microcontroller. The complexity of the Cortex-M is due to the fact that it is designed to save power and support real-time operating system. These features need not be taught right at the beginning. Most Cortex-M4 microcontrollers have implemented many peripheral functions. Instructors can only select those common ones to teach. Students are very enthusiastic about learning the most popular microcontroller in the world. They also performed very well throughout the whole semester.

Conclusions

Due to their low power design and high performance, ARM processors have dominated the mobile, hard drive, digital TV and set-top box markets. By teaching one of the available ARM microcontrollers, students will be benefited in their future job hunting. The Cortex-M processors are a family of processors designed for microcontroller market. We decided to teach the Cortex-M4 microcontroller in our second microcontroller course.

The Cortex-M4 microcontroller implements the Thumb2, DSP, and single-precision floating-point instruction sets. The Thumb2 instruction set consists of both 16-bit and 32-bit instructions with the goal to achieve high performance while at the same time minimize the code size. To achieve low power consumption, Cortex-M4 licensees implement several sleep modes and also allow the clock signals of different peripheral modules to run at different frequencies.

The availability of inexpensive (or free) IDE and appropriate demo board are the major considerations when choosing the Cortex-M4 microcontroller vendor. We decided to teach Atmel's SAM4L because of the free Atmel Studio IDE and the inexpensive SAM4L Xplained Pro kit. According to our experience, the programming of the SAM4L peripherals is simpler than Atmel's 8-bit AVR and 8051 microcontrollers.

With the success of teaching Cortex-M4 in our second microcontroller course, we have confidence to also teach Cortex-M microcontroller in the first microcontroller course using the assembly language.

References

1. ARM Inc. "Cortex-M4 Devices Generic User Guide", 2010.
2. Atmel Inc. "ATSAM ARM-based Flash MCU SAM4L", 03/2013
3. Han-Way Huang, "The Atmel AVR Microcontroller Mega and XMEGA in Assembly and C", Delmar Cengage Learning, Clifton Park New York, 2013.
4. Han-Way Huang, "Embedded System Design with the C8051", Cengage Learning, Stamford, Connecticut, 2009.
5. Han-Way Huang, "HCS12/9S12 An Introduction to Software and Hardware Interfacing", 2nd edition, Delmar Cengage Learning, Clifton New York, 2010.