

---

---

## Application Portability for 32-Bit Microcontrollers – Reality or Myth?

---

---

*Author: Erlendur Kristjansson  
Microchip Technology Inc.*

### INTRODUCTION

In November of 2008, ARM<sup>®</sup> announced the availability of the Cortex™ Microcontroller Software Interface Standard (CMSIS). They claim that this will reduce the cost of designing software when creating projects for new devices or migrating existing software between Cortex-M based microcontrollers from different silicon vendors. This sounds very good, but is it valid? This paper investigates these claims to determine just how valid they are. We will take a look at the components of a typical microcontroller and then see just what can or cannot be gained by adding an abstraction layer on top of the typical peripheral firmware libraries.

According to ARM, 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 for silicon vendors and middleware providers, simplifying software re-use, reducing the learning curve for new microcontroller (MCU) developers and reducing the time to market for new devices... The creation of the CMSIS enables silicon vendors to focus their resources on the differentiating peripheral features of their product, and eliminates the need to maintain their own individual and incompatible standards for programming a microcontroller.”<sup>1</sup>

To put this in context, we need to look at how microcontrollers fit into a design. A microcontroller is a highly integrated system solution. Actually, you can call it a System-on-Chip (SoC). As with any SoC, the chip's function is defined by the hardware peripherals that are integrated with the CPU core. Of course, the performance and architecture of the CPU core defines what kind of code can be executed on the chip; but the chip's interaction with other parts of the system happens through the peripherals. What makes microcontrollers unique SoCs is the flexibility of their peripherals. Because of this flexibility, microcontroller setup and control can be fairly complex. A number of registers need to be set before anything actually happens. To help the designer with this task, the silicon vendor supplies firmware libraries that simplify the code development. These libraries include all the necessary functions to setup all the registers and control all the

different parts of the microcontroller. Each silicon vendor will deliver a unique library, as their microcontroller will have different peripherals and features, which differentiates them from their competition. Let us look at the different parts of the microcontroller that comprise this differentiation from vendor to vendor.

### SYSTEM ARCHITECTURE

Each microcontroller manufacturer has its own way of implementing the overall system integration – i.e., system buses, clock trees and memory – even when the CPU core is the same. These implementations allow them to create advantages that make their microcontroller a better solution for a customer. Let's explore each of these system elements:

- The clock tree provides the heartbeat of the system, coordinating the timing of all of the other functions. The clock tree is designed to optimize the system for speed and economy of operation. The features and peripherals that are included on the chip, as well as the problems the chip is designed to solve, directly influence the structure of the clock tree. Therefore, the clock tree structure is unlikely to be the same between manufacturers. And, the clock tree registers need to be programmed with the right values before anything else will happen.
- The architecture of the system buses defines how all of the MCU functions are integrated. Some manufacturers will use one or more peripheral buses, depending on the peripherals they integrate. All of this will change what and how many registers need to be set, in order to utilize all the available functions.
- Even though all microcontrollers have both non-volatile memory, like Flash, and volatile memory, like SRAM, the exact integration is different for each manufacturer. Some connect the Flash memory directly to the core and to a bus matrix, while others connect it to a system bus or bus matrix. RAM is sometimes located in two independent banks to allow for simultaneous access by the core and peripherals. These varied memory structures might affect how the code is written, and they often have a direct impact on the performance of an application when a designer is moving from one manufacturer to another.

# APPLICATION PORTABILITY FOR 32-BIT MICROCONTROLLERS

## PERIPHERALS

Each manufacturer provides a mix of standard and dedicated hardware peripherals on its microcontrollers.

- Standard peripherals provide common functions, such as serial communication via UART or SPI. They can also be timers or PWMs. While standard peripherals are common among all MCU manufacturers, they may have certain enhancements that give the customer more flexibility and/or features. As with any part of the microcontroller, the standard peripherals all have their own registers. Even though they might have identical functionality, it is nearly certain that each manufacturer implements them differently, resulting in different register structures among MCU vendors.
- Dedicated peripherals are provided for specific tasks that are unique to certain applications. For example, brushless motor control PWMs, I2S for audio playback, or encryption/decryption. Depending on their complexity these dedicated peripherals might have a few registers or 30+ registers.

## FIRMWARE LIBRARIES

As mentioned previously, each MCU manufacturer will have its own peripheral firmware libraries to help customers implement their designs and quickly move to the prototyping stage. A firmware library consists of code that has been developed by the chip manufacturer to set up all the registers for the various aspects of the chip, such as clocks, buses and peripherals. Providing function calls to easily set up the various registers helps developers concentrate on entering the parameters required for their applications. The function takes care of writing these parameters to the appropriate memory locations. By using the functions from the firmware library, the developer will not have to learn all of the registers and their locations to get the chip up and running. This saves time that can be focused on more application-specific tasks, such as developing the appropriate algorithms for the application.

**TABLE 1:**

Feature	Vendor A	Vendor B
Communication Peripherals	5 X USART	4 X USART
	3 X SPI Bus	3 X SPI/SSP Bus
	USB 2.0 OTG FS	USB OTG
ADC	12-bit – 1 Msps	12-bit – 200 ksps
Timers	16-bit	32-bit
Internal Clock Frequency – Max	72 MHz	100 MHz
RAM – Max	48 kB	64 kB

The peripheral firmware library also includes functions to control the peripherals, which is also done through registers. Again, this eliminates the need to focus on the register bits and location, and allows the developer to just pick the right function for their code.

Abstraction layers can be applied to peripheral firmware libraries to help streamline code development and here CMSIS fits into the picture. Still the question is: Does it result in code compatibility?

## A LOOK AT THE FACTS

Now that we have a clear picture of what each microcontroller manufacturer is delivering, we can look at what, if anything, an abstraction layer will do for the designer. First, let's see what CMSIS tells us they are offering, while keeping in mind the system architecture, peripherals and firmware library as outlined above.

CMSIS Version 1.3

1. Core Peripheral Access Layer: contains name definitions, address definitions and helper functions to access core registers and peripherals. It also defines a device-independent interface for RTOS kernels that includes debug channel definitions.”<sup>2</sup>
2. These software layers are expanded by silicon partners with:
  - a) a device peripheral access layer, which provides definitions for all device peripherals;
  - b) access functions for peripherals (optional): provides additional helper functions for peripherals.”<sup>2</sup>

Through a careful reading of this description, we can see that what CMSIS gives us is a common language with which to describe the different components of the MCU.

Next, let's take a look at a sampling of some of the features included in the MCUs produced by two different “Silicon Partners” (MCU manufacturers who use ARM Cortex-M processor cores).

What is immediately obvious from this sampling is that the key features differ between just these two 32-bit MCU manufacturers, even though both use the Cortex-M3 core. The features listed in the table are among the most standard in the microcontroller, yet they differ. What this means is that, even though both of these manufacturers use the same core, they need to make adjustments to the software in order to run even the most fundamental programs, such as toggling I/Os or using the UART. Software cannot be ported between A and B without some degree of rewriting, just to make adjustments for these basic specs.

According to their definition, CMSIS provides a standardized language for accessing elements of the core, but the silicon partners/MCU manufacturers must provide their own firmware to interface with the device peripherals. The two most basic functions of any MCU entail (1) how the system architecture relates to the peripherals, and (2) how the MCU peripherals are structured to produce the required control and/or functions. This is what the manufacturer-supplied firmware libraries help with, and is at the core of the code-compatibility question. Adding an abstraction layer to these libraries will help them become more portable, but will not address any differences in features between two MCUs. Any advanced peripheral function cannot be ported to another MCU if it doesn't exist, with or without an abstraction layer. These enhancements, which the microcontroller manufacturers create, are there to help distinguish an MCU from the competition and help designers make their solution better. This results in code incompatibility.

## GETTING DOWN TO SPECIFICS

The concept of an abstraction layer relates to the design of operating systems. It creates a standard way for the OS to access the processor on which it is running, and can make it simpler if you want to change processors. What has to be recognized is that a processor (Microprocessor Unit, MPU) is a core with minimal integrated system components, which allows for greater similarity between manufacturers than with MCUs. Also, the code that is running on the processor is software, not firmware. That is, the code is not normally controlling any hardware. A microcontroller, on the other hand, is a system-on-chip, which implies that there is a much closer link between the core and the integrated peripherals. With MCUs, the concept of an abstraction layer makes most sense if you are looking to run an RTOS. On the other hand, the use of an RTOS disconnects the application code from the hardware, thus removing the tight control that is normal in microcontroller applications.

If an application is running on an RTOS, it can be ported to any microcontroller with support for that specific RTOS. Code developed on a Cortex-M3 microcontroller running Micrium's  $\mu$ COS II could be ported to a MIPS M4K based microcontroller, since  $\mu$ COS II is also available on that platform. The application is isolated from the hardware and, as long as the new part has all the relevant features, porting should be relatively painless.

On the other hand, this is not the case when there is no RTOS. As mentioned earlier, each manufacturer will integrate the core, memory and peripherals in its own way. Some of these differences make it very difficult to create a single standard abstraction layer that will work among different manufacturers and the question is: Do they want it to be easy?

In most microcontroller applications, this strong connection between the code that is running on the core and the hardware peripherals is critical. In fact, this connection makes it very difficult to create any kind of standard abstraction layer that will translate between MCUs from different manufacturers. An abstraction layer can help with standard functions, such as UART or SPI. But even with a UART, the code would have to be rewritten if the original code used a firmware library function for a UART with a 9-bit mode, when the new MCU does not have a UART with a 9-bit mode and, therefore, no library function for this. To complicate things further, in most cases it is not the standard functions that define the value of an MCU, but rather its unique peripherals that allow the designer to develop an optimal solution. Being able to control a motor with higher precision and safety, for example, can distinguish you from your competition. This is where dedicated libraries come into play, making things much more complicated.

## DEDICATED PERIPHERALS AND FIRMWARE LIBRARIES

The peripheral firmware libraries do support all the peripherals that are on a given microcontroller, including standard and dedicated. But, to better support customers, some manufacturers also supply specialized firmware libraries for specific applications, such as motor control, graphics, networking, etc. These specialized libraries are always proprietary and, in many cases, contain intellectual property that cannot be transferred between manufacturers. In most cases, the applications in question are complicated and designers would prefer to not have to develop the applications from scratch. Also, manufacturers usually integrate all of the special features that their peripherals support into the library, making it impossible to transfer to another manufacturer's MCU without major modification to the code.

# APPLICATION PORTABILITY FOR 32-BIT MICROCONTROLLERS

---

## PORTABILITY

In our example, the microcontrollers from vendors A and B both used ARM Cortex-M3 cores, and had CMSIS-compliant firmware libraries. Does this mean that, for example, their motor control libraries can be ported to each other?

Not really. These two manufacturers take entirely different approaches to their peripherals and firmware. Vendor A uses a dedicated library of algorithms that have been developed, probably with several man-years of effort, to most efficiently utilize the specialized peripherals on its microcontroller. Vendor B, on the other hand, has concentrated on using more generic peripherals and building a generic peripheral library; anything application specific is covered by sample code and application notes.

These two very different approaches make it very hard to port, for example, motor-control subroutines between the MCUs from these two manufacturers, even though both use the same cores. Also, the library function naming is not the same, which means that users will have to rewrite any library function call in the code, and figure out what variables and values to submit to the functions. Not exactly what you would call portable.

## THE BOTTOM LINE

The bottom line for designers is how quickly, efficiently and reliably they can develop the code for a particular end use. Vendors A and B take different approaches to these goals. Vendor A uses a dedicated hardware-based approach, whereas B's approach is more focused on generic hardware. Vendor A makes available libraries for specific applications that have been painstakingly developed to best utilize its microcontroller's features. Vendor B gives only the basic building blocks and lets the developer build their own solution.

On the question of speed and reliability, A's approach has the advantage, since much of the application development work has already been done and tested, and is incorporated in its firmware library. As for efficiency, A also has the clear advantage, since its software is specifically optimized for their hardware. But, none of these advantages has anything to do with CMSIS or portability. The comparison of MCUs from one vendor with other manufacturers should not be based on abstraction layers, but rather on how well it can be used to implement its intended end use. Those manufacturers who develop a firmware library tailored to their particular microcontrollers have the clear upper hand. Designers seeking a controller could then concentrate on comparing how appropriate particular algorithms are for their needs. Although an API like CMSIS is useful for hiding the complexity of the hardware from, for example, an RTOS, making a seamless interface between the two, it in no way guarantees that the software is portable between manufacturers.

## PHILOSOPHY

The final point is to think about the basic microcontroller philosophy. MCUs were developed to bring the programmability and flexibility to control everyday devices. This goal has been achieved by integrating a CPU, non-volatile memory and peripherals, both analog and digital. By eliminating the higher-level functions that make computers so generic, microcontrollers can make the most of a very small amount of hardware. Firmware libraries dedicated to the particular peripheral structure of an MCU dramatically reduce the time and effort required to develop an application. As mentioned earlier, the addition of an abstraction layer to these libraries will help improve the development process and make it easier to reuse the code across multiple projects, but at a cost. There is both a code size and performance penalty when using an abstraction layer. Adding in an abstraction layer moves the microcontroller closer to being a computer. It is, in some ways, contrary to the special qualities that form the underlying philosophy of microcontrollers. In fact, it adds a complication to the development process by requiring that the application adhere to the specific language requirements of CMSIS, and it does not guarantee code compatibility.

## REFERENCES

- <sup>1</sup> [www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php](http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php)
- <sup>2</sup> <http://www.onarm.com/download/download395.asp>

---

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

**Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC<sup>32</sup> logo, rPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICTail, REAL ICE, rLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2010, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-60932-622-7

*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC<sup>®</sup> MCUs and dsPIC<sup>®</sup> DSCs, KEELOQ<sup>®</sup> code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

**QUALITY MANAGEMENT SYSTEM  
CERTIFIED BY DNV  
== ISO/TS 16949:2002 ==**



# MICROCHIP

## Worldwide Sales and Service

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://support.microchip.com>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Cleveland**  
Independence, OH  
Tel: 216-447-0464  
Fax: 216-447-0643

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**Kokomo**  
Kokomo, IN  
Tel: 765-864-8360  
Fax: 765-864-8387

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**Santa Clara**  
Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

**Toronto**  
Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8528-2100  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Chongqing**  
Tel: 86-23-8980-9588  
Fax: 86-23-8980-9500

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4123

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Yokohama**  
Tel: 81-45-471- 6166  
Fax: 81-45-471-6122

**Korea - Daegu**  
Tel: 82-53-744-4301  
Fax: 82-53-744-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Kuala Lumpur**  
Tel: 60-3-6201-9857  
Fax: 60-3-6201-9859

**Malaysia - Penang**  
Tel: 60-4-227-8870  
Fax: 60-4-227-4068

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-6578-300  
Fax: 886-3-6578-370

**Taiwan - Kaohsiung**  
Tel: 886-7-213-7830  
Fax: 886-7-330-9305

**Taiwan - Taipei**  
Tel: 886-2-2500-6610  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820

08/04/10