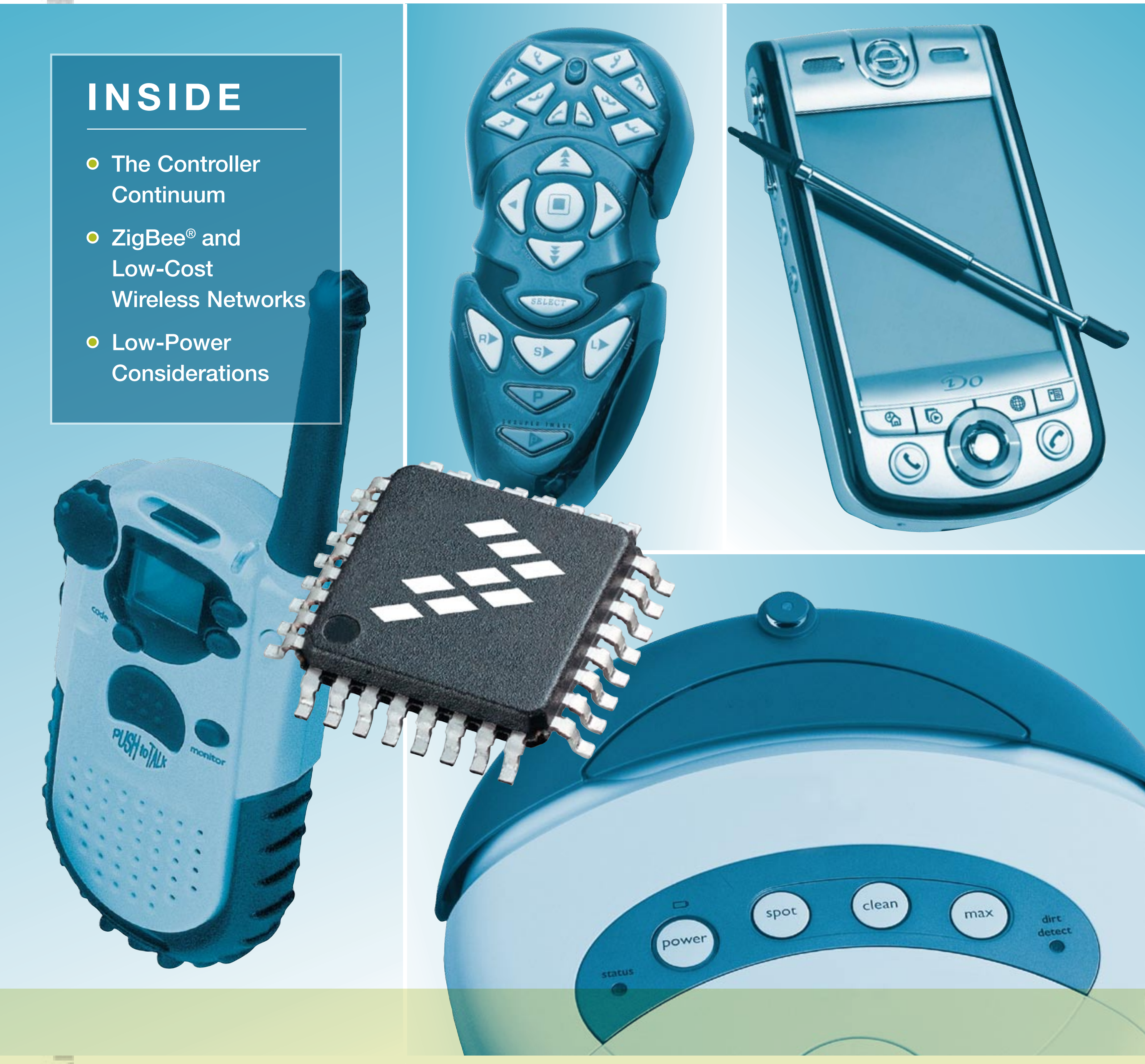
Designer Resource. Accelerate. Simplify.

Issue 2 Summer 2007
Document Number: BR8BITBYNDBITS/Rev 1

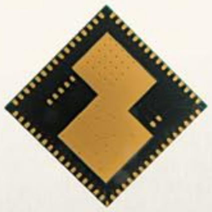
Beyond Bits



INSIDE

freescale.com/beyondbits

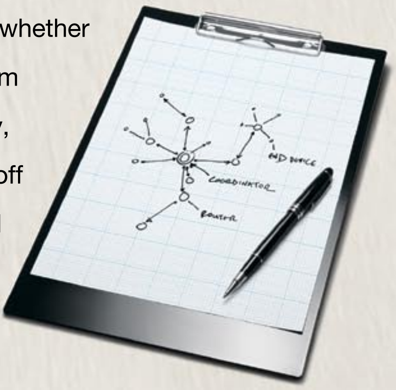
freescale™ semiconductor



It’s 0 to ZigBee in minutes.

BeeKit™ Wireless Connectivity Toolkit

In fact, you can have a small network up and running in just 30 minutes. The secret is our development software environment and its highly intuitive GUI. It lets your team collaborate in an immediate, centralized environment—whether they’re in another building or on another continent. Team members are free to experiment and innovate on the fly, before finalizing their software structure. Shave weeks off your development time on everything from complicated mesh networks to simple point-to-point networks. And make a bee-line to market with your ZigBee® designs.



Get your BeeKit at freescale.com/zigbee

freescale™ semiconductor

Issue 2

Beyond Bits
Expanding our boundaries with our second edition

Welcome to the second edition of *Beyond Bits*! In this issue, we've expanded our focus to cover several technologies in the Freescale portfolio, including 8-bit microcontrollers (MCUs), 32-bit ColdFire® controllers, our award-winning 16-bit digital signal controllers (DSCs) and ZigBee® wireless technology. We've also added content from one of our design alliance partners, as well as examples of how Freescale customers are benefiting from our products and services.

The technical articles included here have a common theme of providing you with practical solutions and problem-solving reference material. The content is well-suited for all levels of experience, and I'm confident you'll find it useful as you grow your expertise and develop new designs.

Many of the articles in the second edition of *Beyond Bits* will help you with your designs for cost-sensitive applications. Some articles discuss exciting new technologies that will help you achieve your design goals. And, a few articles are focused on the Freescale Controller Continuum—the industry's first and only roadmap of compatible 8-bit and 32-bit architectures.

Whether you develop products for the industrial, consumer or automotive market, we have a broad portfolio to meet your needs, delivering great value for your investment. So, please take a deep dive and discover how Freescale's innovative technology can help ease your design process.

This publication has evolved since the inaugural issue—in large part due to the great suggestions we have received from you. Please enjoy this issue and keep the feedback coming. And, most of all, thank you for considering Freescale in your current or future designs.

Best Regards,

[illegible]

Mike McCourt
Vice President and General Manager
Freescale Semiconductor Microcontroller Division

We want to hear from you!
Send your product ideas, tips or questions to us at **frescale.com/beyondbits**.

Issue 2

Beyond Bits
Table of Contents

Jeff Bock and Joe Circello

The Controller Continuum
Solving the 8- to 32-bit transition problem

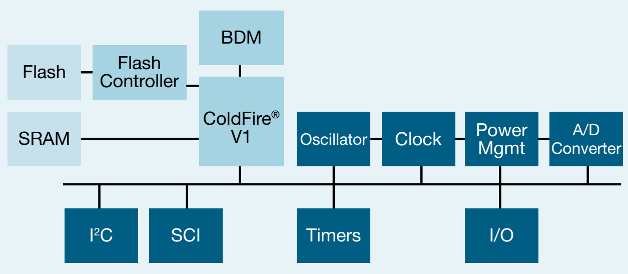
Transitioning from an 8-bit to a 32-bit MCU has historically been a significant challenge for designers. Many factors contribute to establishing a continuum of processor capability; however, established expertise in both 8-bit and 32-bit is essential to understanding the users' needs in each area. This article will explain how Freescale recognized the transition problem, modified hardware and software design philosophy, and ultimately, defined new 8-bit and 32-bit products with compatible packaging, peripherals and tools. The silicon design methodology is expected to simplify and speed performance upgrades in numerous end products and thus enable customers to get to market faster.

The Performance Path

Microcontrollers address applications that range from simple replacements of traditional electromechanical functions with solid state technology to extremely complex controls that have evolved over the years and today require sophisticated 32-bit MCUs. While several companies offer individual solutions for a broad range of applications, none have been able to provide products that make transitioning from an 8-bit to a 32-bit MCU easy. Today the low end 8-bit and mid-range 16-bit requirements for MCUs are expanding rapidly. As these applications demand more performance, the answer could be found in a number of low-cost 32-bit architectures. Nevertheless, for users migrating up from 8- and 16-bit MCUs, the 32-bit architecture feels very different. Usually, new hardware tools are required and as a result, the migration is both costly and time consuming.

By taking advantage of Freescale's extensive experience in both 8- and 32-bit MCUs, we defined an approach that simplifies the transition for customers and allows the reuse of software and tools. Mid-range performance MCUs are frequently defined by price point, power consumption, set of peripherals, and/or performance level required for a specific application—not by bits. In most cases, customers do not care about bits, just about solving the system problem. By bridging this gap with an 8- to 32-bit continuum, customers can start with today's system requirements and plan for tomorrow's improvements without making significant changes and leverage prior tool and software investments.

Figure 1



- ▨ Memory Configurations
- ▨ Core Complex
- ■ Periperal Modules (S08)

At the Core

The ability to transition seamlessly between 8- and 32-bit cores starts at the 8-bit level. The well-established HCS08 (S08) core provides the foundation for 8-bit performance and features an extensive peripheral library. The more recently developed RS08 core reduces the size of the S08 central processing unit (CPU) by 30 percent and provides an even greater range for the continuum. Targeting low-cost applications where electronic control replaces electro-mechanical devices, the core mates to less than 16 KB of flash memory in small-pin-count packages and uses fewer instructions with very compact and efficient coding. This smaller, more efficient core takes advantage of the peripherals developed for numerous S08 applications.

At the high end, an advanced 32-bit ColdFire® V1 core provides the missing link in the transformation chain. The V1 platform includes the V1 core, local memories (flash and the SRAM), a background debug module (BDM) and a bus bridge. The bus bridge provides the interconnection between the core's local high-speed bus and the slave peripheral modules. It is the bus interface that converts one protocol to the other.

The V1 platform portion will typically be running twice as fast as the peripheral modules. In addition to providing the functionality of the bus protocol conversion, the bus bridge module also serves as the clock domain boundary.

Using well-established and easy-to-use 8-bit peripherals with a 16-bit internal architecture and a 32-bit core makes linking the 8- and 32-bit portfolios easier and provides design flexibility for engineers.

The choice to move from an 8- to a 16-bit MCU is based on a number of considerations. With today's processing technology, the 32-bit V1 core costs roughly the same price as a 16-bit core. However, the available performance is significantly higher.

Common Peripherals

The 32-bit V1 ColdFire connects to 8-bit S08 peripheral modules through the IPS bus. The bus bridge module allows the V1 platform portion to run twice as fast as the peripheral modules.

Interchangeable Cores
Either an 8-bit S08 core or the 32-bit V1 ColdFire core can be used with the numerous S08 peripherals and even packaged in a pin-for-pin compatible package. The single-pin BDM allows the use of the same CodeWarrior Development Studio for both 8-bit and 32-bit MCUs.

Compatible Peripherals, Packaging and Tools

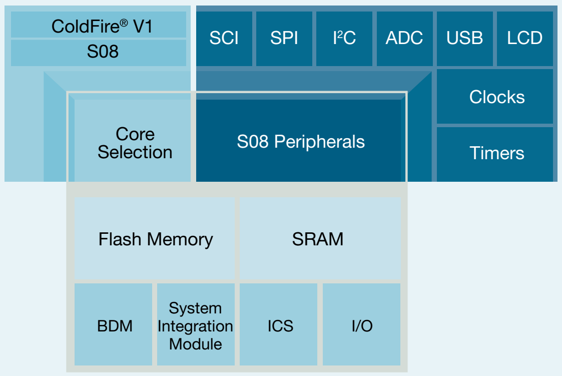
A number of factors contribute to a seamless transition between an 8- and 32-bit architecture. The improvements start at the core and involve peripherals and packaging as well as development tools. Since its initial implementation over a decade ago, the ColdFire instruction set architecture (ISA) has changed from an optimization philosophy for 32-bit operands with some support for 8- and 16-bit operands to one of expanded and improved support for handling 8- and 16-bit operands. For the V1 class of applications, this is particularly important for users migrating from 8- and 16-bit applications. At the processor core level, it is important for the implementation of the instruction set to directly address some of the issues that involve efficient handling of 8- and 16-bit operands.

The V1 core microarchitecture implements a number of optimizations that improve the performance of those instructions handling 8-bit and 16-bit operands. Not only does the V1 support those instruction types, their execution is optimized in the V1 pipeline. In applications with a significant amount of data referenced as 8- and 16-bit operands, both the instruction support and fast execution time for those instructions provides a considerable performance improvement.

The Controller Continuum takes advantage of existing peripheral IP. Peripheral blocks connect to a common bus structure so complete reuse of S08 IP is ensured. As a result, the investment that both Freescale and users have made in the 8-bit domain is preserved in any transition including the re-use of driver software for peripherals. While timing and other issues may need to be evaluated, this approach leverages existing driver software for peripherals.

One of the challenges to keeping the pin count the same was the background debug mode (BDM). The BDM in a traditional ColdFire was implemented on a three-pin serial interface (clock, data-in, data-out) while a fourth pin provided breakpoint capability. The BDM provides the ability to serially insert commands that read/write registers or read/write memory, run and stop the processor, and more. Because the S08s are used in the lower end application space with restricted pin availability,

Figure 2



the debug functionality of a classic ColdFire core was remapped into a single pin interface for the V1 exactly like the S08. The second part of the debug story has to do with ColdFire core support for real-time trace.

For the V1, the real-time trace operation had to be modified. The CPU outputs processor status information, addresses and debugs data. However, instead of immediately outputting this information like the V2 and other ColdFire MCUs, the V1 incorporates a PST (processor status) and debug data trace buffer. With this approach, users program start and stop conditions for recording information that comes out of the trace port, then the information gets collected, compressed and loaded into the trace buffer. Next, the trace buffer can be read out using the single pin BDM capabilities. This is a very powerful concept that preserves the trace capabilities by capturing events spanning hundreds of machine cycles, but maps them into a technique that works with a single-pin interface.

The biggest challenge to an 8- to 32-bit transition is the tools. The main development tool has to be interchangeable and provide 8-bit customers the same simple look and feel as they move up in complexity to 32-bit. With a new version of the CodeWarrior® Development Studio, an 8-bit S08 can be removed and replaced with a 32-bit ColdFire V1 core using exactly the same tool, same cable and the same set of CodeWarrior tools. The user simply recompiles C-developed code.

What Users will Do Differently

With the Freescale Controller Continuum, a customer who starts with an S08 microcontroller can transition easier than ever before to a 32-bit architecture. With the transition, users get the processing power of a 32-bit core that is significantly higher than that of an 8-bit S08 core. The 32-bit V1 provides an estimated performance improvement up to 10X over the 8-bit S08 architecture running at the same speed. The performance improvement over an RS08 would be even greater.

This amount of performance improvement opens up new application areas and can change the way a company plans their long-term product roadmap. Based on the ease-of-use of the 32-bit architecture, this approach is ideally suited for 8-bit users seeking an easy entry to 32-bit performance.

The continuum approach also works for 32-bit users looking to reduce power consumption and cost in their application. Using an advanced low-voltage, low-power process, the V1 provides a very low power consumption 32-bit MCU for both standby and run current. Furthermore, the power density is expected to be unique and create a number of new applications.

Since the ColdFire V1 MCU products have up to 10x the performance of S08 devices, it is possible for products to significantly increase performance within an application without increasing frequency. This can be an important benefit to applications sensitive to EMC such as appliances and washing machines. These approaches have been discussed and validated with many customers. Once designers evaluate the potential for implementing this new methodology in their design process, they immediately envision new ways to address future designs.

Getting There from Here

The continuum builds on extensive 8-bit S08 as well as 32-bit ColdFire processor experience by taking action on the feedback from customers. This feedback has led to a number of improvements to the instruction set, hardware (including debug capabilities) and development tools for these devices that today makes an upward transition quite simple. In addition, Freescale is introducing the Flexis™ series—the first 8- and 32-bit pin-, peripheral- and tool-compatible microcontrollers at the “connection point” of the Controller Continuum. Using the S08 and ColdFire V1 cores, the MC9S08QE128 and the MCF51QE128 are the first products in the Flexis series that make this compatibility roadmap a reality.

In most embedded control designs, there rarely is one major factor that swings a design decision one way or the other. The decision typically depends on an accumulation of many minor, but still important, details. By taking a “leave no stone unturned” mentality in terms of optimizing power, size, performance, peripheral usage and development tools, the 8-bit to 32-bit Controller Continuum delivers unique capabilities to system designers. In addition, with the ability to seamlessly transition upward or downward, the continuum provides companies a long term planning capability that promises to change the product roadmaps in many industries.

Reference

Freescale Controller Continuum
www.freescale.com/files/abstract/overview/TSP8773_CONTINUUM.html

Freescale Flexis™ MCU Series
www.freescale.com/flexis

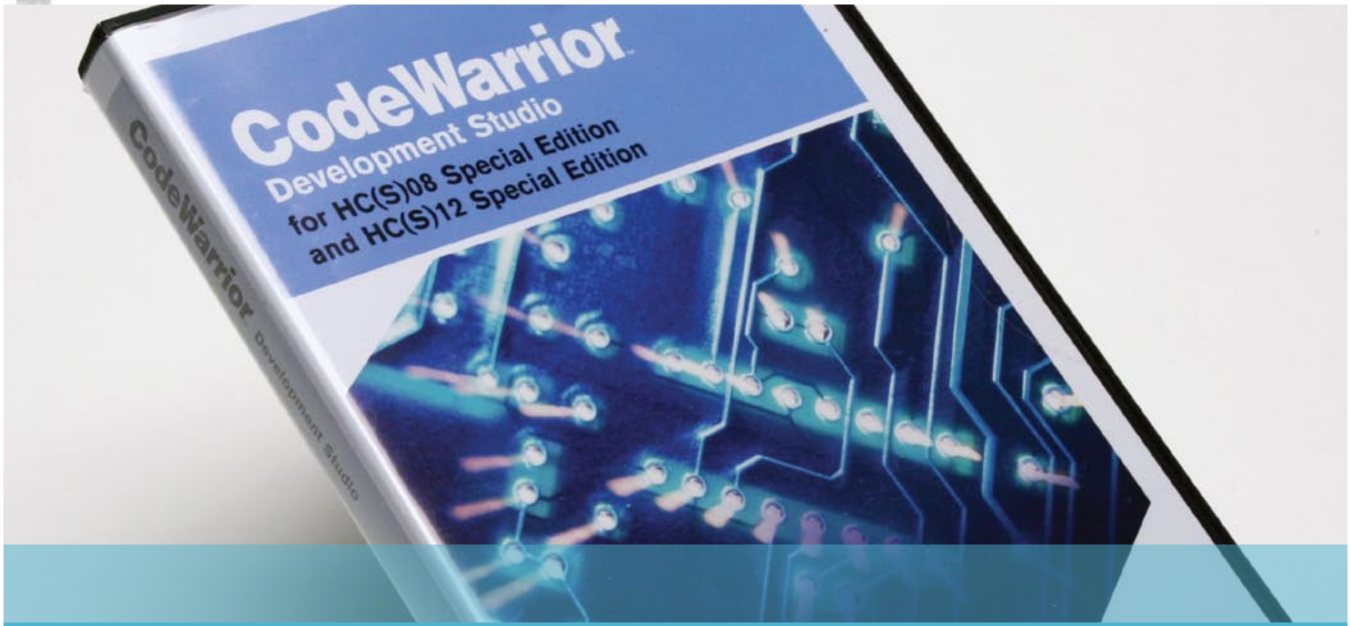
Jeff Bock is global product marketing manager within the MCU Division of Freescale. Jeff has general responsibility for driving product launch and promotion activities for microcontrollers targeted at the consumer and industrial spaces. He has over 10 years of experience within the semiconductor industry.

Joe Circello is a processor/platform architect in the MCD of the TSPG Group at Freescale Semiconductor specializing in pipeline organization, control structures and performance analysis. In 15+ years at Motorola/Freescale, he has served as the chief architect for the MC68060 and the ColdFire family of devices.

Robert Lewis—Design Alliance Partner

The Path to Success

Embedded processors and C



The project began as projects often do—with a reasonable marketing plan, a functional specification and a realistic time line. And it probably would have all worked exactly as planned if the inevitable had not happened. The predictable change of scope… "we need to add would it be possible to, we could really improve the product if we just added, it can't be that hard. After all, it's just software."

But, let's go back to the beginning—before the amoeba evolved into the eagle. The project was simple enough, take in a frequency, convert it to another based on a user-selected setting and output it under certain conditions. It all could have been done with a simple 8-bit micro, but we wanted extra debugging capabilities and at that time the 908 had only one break point, so we chose the HCS12 family. We didn't want to program in assembler, so C seemed the only reasonable alternative and a good solution. Assembler would have worked fine, but the overhead in managing consistent programming standards and sorting out debugging because of naming problems and register addressing errors was not worth the extra risk.

Next, we started a search for a good ANSI compiler. We looked at the CodeWarrior® tool set, but given the size of the project and the limited functions we would be performing, it didn't seem reasonable to spend a significant amount of the budget for the tool set and the large amount of documentation with seemingly endless switches and options. This suggested a lot of time would be spent just learning the interactive development environment (IDE).

So we began to look into free and third-party C compilers. The free compilers seemed to be stable enough, but it became readily apparent that a lot of expertise would be required to set up the system for compiling and linking, and the only support was through Internet forums. At that point in time, we didn't have either the time or background to take the risk with that many unknowns, so we focused on what seemed to be a good third-party alternative product. The users in the forums said great things about the stability and support of the full ANSI C compiler, so we settled on using the chosen product.

Now that we had the compiler, the debugger was next. It would have been nice to buy a $20,000 in-circuit emulator (ICE), but that didn't seem like a reasonable fit with our $300 compiler or the budget, so we ruled it out rather quickly. Instead, we looked into a product that required "no in-circuit emulator." After running the product in evaluation mode for a few days, I knew we had made a good decision. The product had an excellent and well thought out interface, was solid and had no crashes, hang-ups, dumps, lockups or other usual issues. (The latter features we are all aware of in the wonderful world of "bling" web pages and little substance.) We could see this product was going to work well.

We had just a few more tools to put together before we could start. We needed a multi-pane editor, a source code repository to track revisions and a source code analysis system for cross referencing and finding symbols. The open-source product for source code analysis was an amazing free product with a lot of functionality but, alas, no documentation.

By now you may see the evolution toward what we failed to see. We were effectively building our own IDE and therefore would also be responsible for integrating and supporting every aspect. Where there was no documentation, we would be required to discover how to make the tool work. This proved to be no small task.

As expected, it took a while to get the compiler and debugger to work. The debugger was fine, but this was a new release for the compiler and there was a lot of hair pulling trying to get the object code to flash correctly and matching up correct access to the paging register. Unfortunately, this problem occurred while the compiler vendor was away on a much needed vacation. In the end, we had good support from the debugger group. They sorted out the inconsistencies in the compiler and when the vendor returned, the compiler was changed to flash and loaded correctly.

For a time, all went well, the code was in design and the coding standard was more or less established with respect to naming conventions, function calls and so forth. Then, the first change came along. Instead of a simple BCD switch, we now needed a four line by 16 character LCD display and a push button to select the options. ("What options? A BCD switch doesn't have options… it has switches! They are on or off and they are read at startup! Well we need a display and a switch…")

Now we had to write routines to set up the LCD control registers, strobe the address and data, write all the primitives to position characters, build lines from characters and about 40 other display related routines. Having done that, an interrupt handler had to be added for running the five-way switch. Additional routines were needed to track where the cursor was on any given line within the menu. There were now sets of lines within sets of functions, so we needed a menu to keep track of the options.

The code had grown, but was still manageable until the next request came along, which was a way for users to save their options. My response was, "if you use a BCD switch, you don't need to save anything, you just look at the switch." Since that suggestion was rejected, we moved on at first to a serial data link. But now we also required a PC application. After bread boarding a serial link and working with a sophisticated fourth-generation tool for the PC application, I knew the inherent error-prone connectivity of serial transfer was not for me. The system would hang if it got out of sync, or if it miscued a byte everything would get out of step and do bizarre things. The only solution would be a significant amount of handshaking software. After much casting about, we decided to add an SD card with a FAT file system rather than use an active link to the PC. This is where everything took a turn for the worse.

The SD card addition added about 20 KB to the object output and a significant problem appeared. The debugger started showing the code stepping off into regions that were completely unrelated. At seemingly random times, the execution would jump into unrelated functions. We spent days writing code to try to trace possible stack problems and more time trying to trap what could possibly be errant interrupts, all without success. Finally, with the help of the author of the debugging tool, we determined that the compiler was producing incorrect symbol tables and linkages because of the order of the include functions and because the functions were included in-line rather then linked in. The vendor of the compiler gave us a work-around and we were back in business again, albeit somewhat worse for the wear. We trudged on waiting for a permanent fix, but at least we were able to proceed.

Next came the statement that the character display was "under whelming" and devalued the product—we needed a graphics display. Nothing too fancy. It could be simply monochrome, something that was not as crude and archaic as our current character display. My reply that a "BCD switch was stylish, compact and came in a variety of colors" was taken as neither constructive nor helpful.

The addition of the graphics routines with the font tables added another 20 KB to the object code and further problems appeared in the code execution. The compiler appeared to be having problems with setting the paging register. The vendor of the compiler was dedicated to doing his best to support the product, but it was also apparent from cross posts in the forums that his company was devoting their time to a new compiler for a different manufacturer's product.

This was the last straw; we had to finally admit that if we were going to maintain the code base we had developed, we needed a product that had guaranteed stability. The only option was to rethink using the CodeWarrior tool set. We were not naïve enough to think that the CodeWarrior IDE would be without its own particular brand of problems, but we did know that large multinational companies like Freescale that made the processor and provided the tool set would have a very strong motivation to keep the tool set current and operational.

So, we bought the full version of the CodeWarrior IDE but without the full version of Processor Expert™ beans. This version allowed us to build any size of code base and it also had some basic Processor Expert objects. At the time, we knew nothing about the Processor Expert beans. It seemed that just mastering the CodeWarrior tool set would be enough to start with.

I went through some of the training modules on the Freescale Web site, and the process seemed less daunting than I had first imagined. Next, I started to convert the code. I never adjusted any of the compiler or linker switches but used the defaults.

In about two days, we had the code compiling and linking without error. There were very few changes since the compiler we had used was ANSI-compliant. The main changes were in the file system naming, pragma statements and building against the CodeWarrior definition files for the port and ECT assignments. The file system had used a few reserved function names such as fopen( ), fclose( ) that were found in the ANSI library. They were easily corrected by renaming our function calls to fopen_imn, and so on. The pragma changes were obvious and very simple to correct.

We loaded the code and ran it, and to our amazement it worked perfectly—the first time. Not only that, but because the CodeWarrior IDE was an optimizing compiler and our old compiler was not, the CodeWarrior code was more than 30 percent smaller and significantly faster. All of the problems and bugs we encountered in our previous code base disappeared. And, as a bonus, our old friend the (no in-circuit emulator) debugger worked with the CodeWarrior ELF output. We could use either the CodeWarrior debugger or our previously chosen tool to debug. We currently use the X-Gate processor; therefore we most often need to use the CodeWarrior debugger.

In hindsight, what can be learned from all of this? Even though I had been doing very large control system projects for more than 30 years, I was still caught by the trivial traps I had often cautioned others against. When it was not my money, I bought the best tools and whatever else was needed to ensure the fastest project completion with the least risk. When it was my money (partly), I traded my time for a lower cost tool set because I trusted my expertise to make it all work; and it did… up to a point. But there comes a point at which you cannot control the outcome of a project when others are involved. If I were to calculate my time at even a very low rate, we paid for the CodeWarrior tool set several times over. Additionally, that doesn't account for the emotional expense or the lost revenue due to the project being late. Sometimes, it may not be an option to come up with the money for tool set. However, now with the free CodeWarrior tool set supporting up to 32 KB program sizes and with a compiler that optimizes so well, a lot of projects can be built that would previously not have fit. Later, once the project is large enough, there may be the funds to afford the extra expense.

When you are dependent on a small vendor with a small customer base, that vendor can be influenced by factors that they cannot control. The loss of a key employee, sickness and change in revenue base, for example, can remove any vendor's ability to deliver.

This may not be a factor worth considering in all projects. Whether this plays a part in your decision process depends on the end user. If your product is one-of-a-kind and a personal customer that you can support locally and quickly with a work around, there may be no risk at all. You may never encounter compiler and linker bugs, either because your code base is small or so thoroughly tested from past projects that the new code is easily debugged and separate. However, if you are building a product that will go into a real-time control system or be manufactured in any sizable volume, finding an error once the product is with the end user could cause a sizeable revenue consequence and loss of credibility. This type of project needs an immediate response, and it may be wise to rethink your "insurance policy." That is, when you pay the extra money for a proven, supported product, you can escalate the support problem and get a response. This may not solve the problem immediately, but it's better than being told "gee we've never seen this before; check on the forum to see if anyone else has a suggestion." That is a very lonely feeling in a crisis.

Did I mention our journey through the twilight zone of Processor Expert beans? No? Well, perhaps another time.

Robert Lewis is an engineer at iMn MicroControl Ltd. He holds Bachelor of Science and Master of Science degrees in electrical engineering, with a specialty in microprocessor-based systems.

Inga Harris

Low-Power Considerations
In consumer and industrial applications

The consumer and industrial markets are demanding greater energy efficiency in new product designs. Portable products need to extend battery life with each new generation, and larger products, such as white goods, require higher energy efficiency ratings to remain competitive. In many cases, replacing mechanical components with electronics improves board power consumption, and additional software control can fine tune application power performance.

Advances in product components enable engineers to use new concepts and technologies to conquer the power use limitations of earlier products. The microcontroller (MCU) is key to these power reduction innovations for two reasons. First, the MCU itself is consuming power, and second, the MCU is controlling the other system components that also consume power. Therefore, the MCU manufacturers need to incorporate very low power features into their latest product designs.

This article highlights what Freescale is doing to address this market need and describes the methods used and what features are added to reduce power consumption. Most importantly, it explains how to take full advantage of these methods and features in application design.

The power savings are highlighted by demonstrating a general purpose application on the MC9S08QE128, our latest low-power 8-bit MCU. By turning on each power saving feature, one at a time, it shows what impact each feature has on the application's power consumption.

Power Versus Performance

System designers have to balance the power consumption (Amps) and performance (MIPS) to suit application demands. Generally, the faster components are clocked the more power they consume. In an application that is running most of the time, the most efficient solution is one that will run as slow as the system can tolerate. A motor controller is an example of such a system. If the key function is to convert a signal via an analog-to-digital converter (ADC) channel, then the ADC cannot be clocked slower than the minimum sample frequency due to Nyquist's law; therefore, limiting how slow you can run the MCU.

In many cases, the application will perform with lowest overall power by performing the scheduled tasks as fast as possible to allow the MCU to quickly revert to a very low-power state. For example, a smoke detector only needs to wake every five seconds, take a reading, make a decision and go back to sleep. This method is very common in battery operated devices. Such systems are asleep approximately 99 percent of the time.

To enable such systems, Freescale has incorporated fast clock switching and fast wake up routines to help application designers meet their target power consumptions.

Smoke Detector Case Study

Figures 1 and 2 show a smoke detector block diagram and the power vs. time graph. The MCU has one main input from the smoke chamber and two outputs—an alarm and a light-emitting diode (LED). The real-time clock (RTC) module on the MCU can wake the device periodically from Stop 2 mode (described later) to enable the Op Amp to take a reading and process the data to make a decision on the LED and alarm status. The MCU can then re-enter Stop 2 mode to conserve power. The HCS08 has a slightly lower Stop 2 mode current than the ColdFire V1 equivalent part, but the S08 CPU has fewer built-in data processing capabilities. Figure 2 compares the power and time plots of both devices, enabling you to choose the solution that best fits your application profile. In applications where task execution speed is of the utmost importance, the MCF51QE128 ColdFire V1 device may be the better solution. The pin compatibility of these two products means that migrating from one strategy to the other is quick and easy, as the same board, software and tools can be used.

Figure 1: Simple Block Diagram of Smoke Detector

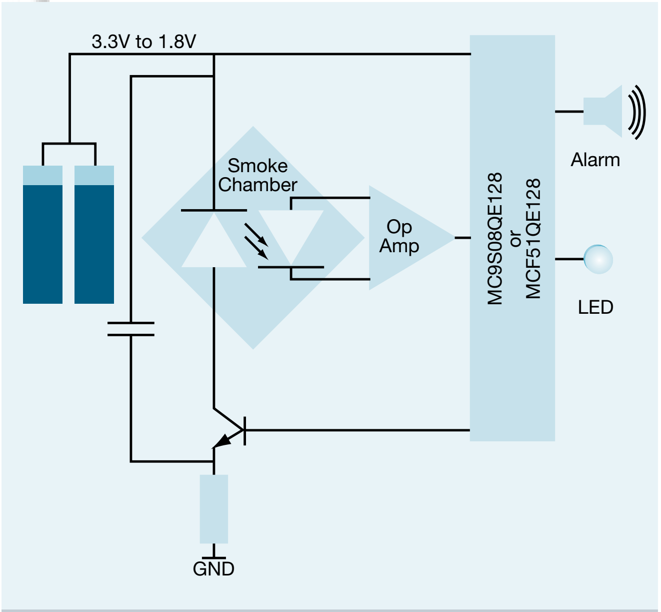
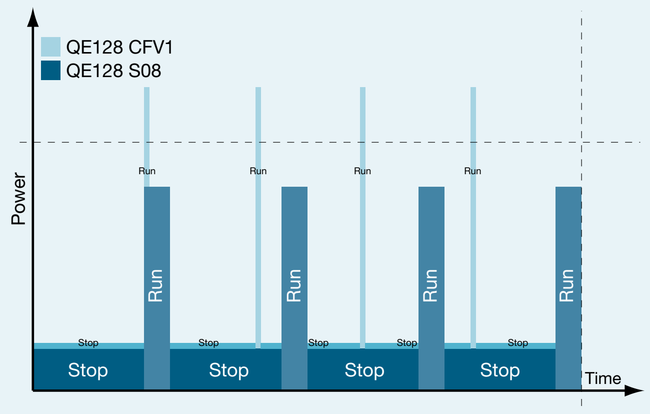


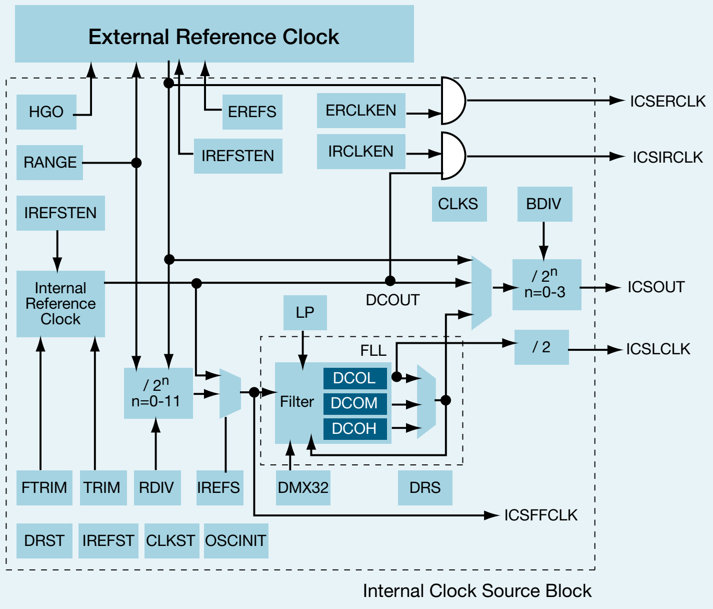
Figure 2: Power versus Time Plot for Smoke Detector Application



Clock Selection

The QE128's heart is the internal clock source module (ICS) as shown in Figure 3. This module enables designers to select an external reference clock (ERCLK) from 32 kHz up to 16 MHz or an internal reference clock (IRCLK) that is trimmable from 31.25 kHz to 39.06 kHz. The heart of the ICS is the frequency lock loop (FLL) block which multiplies its input clock up to a maximum 50 MHz. The input clock to the FLL, also known as the FLL reference clock, must be in the 31.25 kHz to 39.06 kHz range for the FLL to operate correctly. This is simple with the internal FLL reference clock as all you have to do is trim to the correct range. The external reference clock can be as high as 16 MHz and can be divided by a reference divider (RDIV), which is programmable from one to 1024. The FLL can also be bypassed if a low frequency bus is required. A second divider block, bus frequency divider (BDIV), can divide the clock signal down by one, two, four or eight before it is put out onto the ICSOUT signal. The MCU bus clock is ICSOUT divided by two.

Figure 3: Internal Clock Source Heart



The ICS also controls an independent 1 kHz very low power oscillator (LPO), which can be used by the RTC and the watchdog (COP). It is a feature that can be used to conform to EN60730, the standard for automatic electrical controls for household use and similar applications.

Table 1 shows which clock source can be used by each module on the MC9S08QE128. Other clocks shown in the table are OSCOUT, which is a direct path to the external clock and XCLK, which is the signal going into the FLL block—either the internal oscillator or the external clock source, post RDIV.

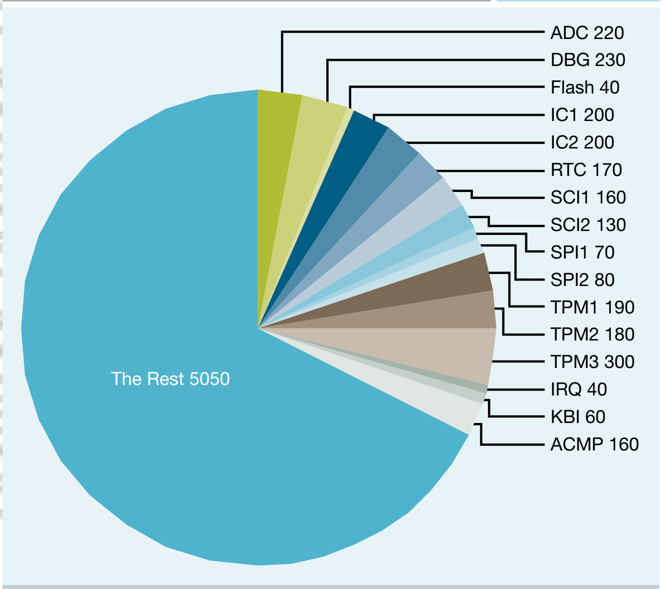
Table 1: Available Clock Sources Per Module

	ICSOUT	IRCLK	ERCLK	LPO	Other
ACMP	/2				
ADC	/2		•		
COP	/2			•	
CPU	/1				
Flash	/2				
I²C1	/2				
I²C2	/2				
RTC		•		•	OSCOUT
SCI1	/2				
SCI2	/2				
SPI1	/2				
SPI2	/2				
TPM1	/2				XCLK or External Source
TPM2	/2				XCLK or External Source
TPM3	/2				XCLK or External Source

The key to the MCU's flexibility is the ability to use the internal oscillator and the external oscillator source for different modules at the same time. This means by running some modules slower than others, power consumption can be reduced.

A secondary power saving feature related to clocks has been incorporated into the MC9S08QE128 design. The system clock gating control registers 1 and 2 (SCGC1 and SCGC2) gate on (1) or off (0) the clock source control the clock gating to the timers, ADC, inter-integrated circuits (I²Cs), serial communication interfaces (SCIs), debug module, flash memory, external interrupt request (IRQ), keyboard interrupt module (KBI), analog comparator, RTC and serial peripheral interfaces (SPIs). By gating off the clock to unused modules saves precious micro amps (µA) in the MCU's Run and Wait modes (described later). This feature is especially important on devices like the MC9S08QE128 because the MCU incorporates many communications modules and timers.

Graph 1: µA Saved through Clock Gating @ 10 MHz bus in FEE Mode



Gating modules on only when needed and by gating them off when the job is done is a significant power savings technique. The clock is gated on/off immediately after the register is written. However, this method has vulnerabilities that need to be considered.

- After a reset all the clocks are gated on, so to keep power consumption down the clocks should be gated off as soon as possible
- Writes to registers associated with a gated off module have no effect
- To avoid erroneous operation, it is good practice to disable the module before gating it off and reinitialize it when the module is gated back on

Graph 1 illustrates the power savings that can be achieved by gating off each module. The numbers are indicative of a bench environment and are not characterized or guaranteed by Freescale.

Modes of Operation—Clocks

As described earlier, the ICS is routed to all the power consuming and component controlling peripherals. This allows designers to choose the power reduction and performance levels they need. All other factors being equal:

- Using the low range, low gain external oscillator is more power economical than using the internal circuitry
- Bypassing the FLL, or phase-locked loop (PLL), will save power by reducing the frequency as well as the power saved bypassing the FLL circuitry
- It is possible on the MC9S08QE128 MCU to disable rather than bypass the FLL further power reduction

Note that when choosing between the PLL and FLL, the FLL is the lowest power option, although it's less accurate in the long term.

The ICS on the MC9S08QE128 MCU has six modes of operation: FLL engaged internal (FEI), which is the default mode; FLL engaged external (FEE); FLL bypassed internal (FBI); FLL bypassed external (FBE); FBI low power (FBILP) and FBE low power (FBELP). Each is activated by the LP bit in ICS control register 2. FBILP and FBELP reduce the voltage swing on the oscillator tracks, lowering overall power consumption.

Modes of Operation—CPU

Choosing the right mode for the right stage of the system design can make or break the product's power consumption targets. The days of simply having the choice of a Run mode and a low power mode are over as the latest generations of MCU products have a variety of modes with various compromises and advantages. Exit paths, wake up times and register retention are all choices that you have to make. Along with the usual Run mode, the MC9S08QE128 has five other power saving modes. Three of them, Wait, Stop 3 and Stop 2, are not new to the S08 core and are common in other cores, although they may be under different names.

In Wait mode the CPU shuts down purely to conserve power. The MCU's system clocks and full voltage regulation is maintained. Wait mode saves between 30 to 60 percent of the run mode current, depending on the bus frequency, and any interrupt will allow Wait mode to be exited instantly.

Stop modes halt the system clocks and place the voltage regulator into Standby. Stop 3 on the MC9S08QE128 MCU consumes the highest power of the two Stop modes and has the advantage of allowing a wake up time of just six micro seconds (μs). There are three versions of Stop 3:

- With BDM enabled
- With the voltage regulator active
- With no frills

Unlike Wait, exit from Stop 3 is limited to the RTC, low voltage detect/low voltage warning (LVD/LVW), ADC, analog comparator with internal reference (ACMP), IRQ, SCI, KBI and Reset. Stop 2 is the name of the lowest power mode on the MC9S08QE128 MCU. The internal circuits are powered down while still maintaining the RAM contents and the pin states. The major difference is that there are only three ways to get out of the mode—Reset, IRQ and RTC. The MCU resets upon exit in the same manner as a power-on reset.

There are two new power saving modes on the MC9S08QE128 device—Low Power Run (LPR) and Low Power Wait (LPW). LPR saves power compared to normal Run mode by placing the voltage regulator into standby. To enter this mode, FBELP, the lowest power ICS mode, must be running. The low voltage protection system must be disabled along with the MCU's internal bandgap and the on-chip in-circuit emulator/background debug controller (DBG/BDC) modules as all three modules consume significant amounts of power, regardless of the bus frequency. Since the voltage regulator is in standby, every opportunity to keep the power down in this mode is recommended.

Low Power Wait (LPW) mode can only be entered from LPR mode and as such the restrictions on LVD circuitry, debugging and ICS mode are the same. The MCU can typically save 50 percent of its current consumption in LPW versus LPR.

Table 2 compares the key characteristics of the CPU operation modes.

Table 2: CPU Mode Comparison Chart

	Typical Bus Frequency	Typical Idd	Exit Sources	Exit Time
Run	8 MHz	5 mA	n/a	n/a
LP Run	16 kHz	25 μA	Clear LP bit	n/a
Wait	8 MHz	2 mA	any interrupt	instantly
LP Wait	16 kHz	4 μA	any interrupt	instantly
Stop 3	n/a	450 nA	RTC, LVD/LVW, ADC, ACMP, IRQ, SCI, KBI	6 μs
Stop 2	n/a	370 μA	RTC, IRQ or RESET	6 μs plus 73 bus clocks

Other MCU Features to Help Save Power

The MC9S08QE128 MCU has flash memory which is reprogrammable down to 1.8V, which helps extend battery life in portable applications. Three ports on the MC9S08QE128 device have Set, Clear and Toggle capabilities, which can speed up pin manipulation, reducing code execution time so low power modes can be re-entered quickly. All of the device's output pins have slew rate control, which, when driving external components, can reduce power by slowing transients and drive strength control.

Conclusion

By giving careful consideration to the clock source, clock distribution and clock mode on a module basis, and by using the right CPU mode at the right time and for the right length of time, the tight power consumption limits placed on modern consumer and industrial products is achievable through the MCU. The clocks and CPU modes are not the only features reducing power or extending battery life, and there are other simple additions discussed further in application note AN3460 available for download from freescale.com.

In summary, an optimized MCU design can halve a system's power consumption. The features Freescale incorporates are simple in concept and easy to use. The diversity of features means all types of applications, whether battery or mains powered, can benefit. They are easy to understand, incorporate and use, offering a real tool for you to exercise a significant impact on critical power aspects of your system.

The S08 product incorporates SuperFlash® technology licensed from SST. This information is subject to change, and the latest reference manual should be used for decision making.

Inga Harris graduated from the University of Strathclyde in 2000 with an honors degree in electronics and electrical engineering. Her Technical Marketer and Applications Engineering career at Freescale has predominantly been with the 8-bit MCU families with a strong focus on Consumer and Industrial markets.

Radomir Kozub, Pavel Lajsner

Ultra-Low-Power Wireless Designs
For real-time communication

Our design team was asked to develop a wireless system to demonstrate a triaxial accelerometer (Freescale MMA7260QT). The goal was to replace the serial cable of the original RD3112MMA7260Q (STAR) demo by a wireless connection using the latest Freescale technology to transfer accelerometer values from sensor to PC in real time.

Key features of the wireless demo include:

- Very small footprint (both hardware and software)
- Low power
- Battery powered
- 20 meter wireless range
- Printed wire antenna
- Real-time response
- Low cost
- May be used as a reference design

After initial evaluation, it was clear that the major design challenge would be to have low power and low current consumption. The target of this demo was to provide accelerometer measurements with a transmission frequency of about 30 transmissions per second. Thus every microAmpere-second (μAs) counts when using a very limited power supply.

Knowing the battery for the sensor board without first knowing which wireless technology would be used was a challenge. At the beginning AA and AAA alkaline batteries were considered. The allowable current was adequate, but the physical battery size would overwhelm the rest of the project. Since one of the desires of this demo was to demonstrate Freescale's very low power technology, it was decided to use a Lithium primary "coin cell" type battery. The popular 220 mAh CR2032 3V coin cell, with a suitable battery holder, proved to be ideal from a form factor point of view while providing sufficient battery life to make a successful demo. To further emphasize the low power battery source, a transparent cover or even no cover was considered for the sensor board.

Finding the Right Wireless Technology

The next design choice was determining the best low-power wireless technology to transmit data from the accelerometer to the computer. The 2.4 GHz industrial, scientific and medical (ISM) band was selected due to its global acceptance and its ability to take advantage of small board sizes and an inexpensive (printed wire) antenna.

Three well-known standards-based wireless communication technologies operate within the 2.4 GHz ISM band—Wi-Fi, Bluetooth® and IEEE® 802.15.4 technology. Wi-Fi is probably the best known, providing the fastest communication rates, and is ideal for the computer connectivity segment. However it was not designed for extreme low-power operation and long battery life. Bluetooth technology is certainly better for low-power, battery-operated devices, having been designed to provide short-range (10m or less) moderate rate wireless connectivity for phones, headsets and other handheld consumer devices. IEEE 802.15.4 technology, on the other hand, was designed from the beginning to provide months to years of low-rate, short-range (20m to 100m) wireless connectivity for devices operating from a small battery. It is intended for low data rates (such as in sensor and control networks) and excels at extremely low-power consumption. For these reasons, 802.15.4 based radios were chosen for the wireless accelerometer design demo that the team has christened ZSTAR.

Demo Overview

The demo uses the ZSTAR sensor mode and an 802.15.4 wireless data collector module that is connected to a PC (Typically by USB). Freescale has a number of demo modules that can be used for the data collector. The module used by the ZSTAR demo is the wireless sensor board which uses three main Freescale devices:

- The triaxial accelerometer MMA7260QT, known as "triax"
- The MC9S08QG816-pin 8-bit microcontroller (MCU)
- The MC13191 802.15.4 transceiver (radio modem)

All of these devices are capable of very low power operation. To ensure reasonable battery life, it is necessary to take full advantage of the terrific low power operating modes that all of these devices offer.

Table 1: Current Consumption for Modem, MCU and Triax States

Modem Operational Mode	Description	Typical Current Consumption
Off	Total device powered off	0.2 µA
Hibernate	RAM/register content is retained, digital IO retain their states. Reference oscillator is off.	1 µA
Doze	Crystal reference oscillator is on. RAM/register content is retained, digital IO retain their states. Output clock can be available. Timer can be used for wake-up.	35 µA
Idle	Reference oscillator running. Fast transition to Rx and Tx modes.	500 µA
Receive	Receiving mode or CCA	37 mA
Transmit	Transmit mode	30 mA
MCU Operational Mode	**Description**	**Typical Current Consumption**
Stop1	All I/O pins automatically transition to their default reset status. Most of the internal circuitry of the MCU is powered off. RAM content is lost.	475 nA
Stop2	Retains RAM, all I/O pin control signals retain state. Wake-up treated as start from reset.	600 nA
Stop3	Retain all of the internal registers and RAM contents, and I/O pin states are maintained. Wake-up treated as an interrupt service.	700 nA
Wait	CPU core clock stopped, all enabled peripherals running	1 mA
Run	All peripherals off, CPU core clock is 32.768 kHz derived from 32.768 kHz crystal	95 µA
Run	All peripherals off, CPU core clock is ~32 kHz derived from internal oscillator (without frequency locked loop)	150 µA
Run	All peripherals running, CPU core clock enabled (8 MHz)	3.5 mA
Triaxial Operational Mode	**Description**	**Typical Current Consumption**
Sleep	Outputs deactivated	3 µA
Run	Triaxial accelerometer operating	500 µA

Low Power Considerations

Several important factors affect power consumption:

- Power saving modes for the MCU, Triax and Modem are managed for each part of the complete communication cycle (the period in which one set of accelerometer values is sent toward PC)
- The number of packets sent per second (i.e. the rate of the communication cycle)
- The number of packets exchanged per second (the 802.15.4 protocol provides for an acknowledgement for each packet, but to save power only one acknowledgement was sent for every eight transmissions sent from the sensor)
- Packet length (the protocol needs to be simple and lightweight with minimal overhead and minimum data size)

To overview the modes available for the components, Table 1 summarizes the available device states.

System Trade-Offs for Low Power

Operation of the sensor board must be viewed from a power consumption point of view. Since the number of transmitted packets and their size is consistent and does not vary, the power must be managed looking at the various parts of the duty cycle.

A. Low power or quiet time between communications

Because of the broad flexibility allowed for MCU clock modes, the low power considerations can take advantage of choices between accurate timing and less accurate timing.

1. **Accurate (crystal based) receive/transmit timing**—every transmit/receive operation can be precisely timed, in addition the receive window can be made as narrow as possible to save current.

a. MCU with 32.768 kHz crystal, Modem in Off mode. The MCU runs at a low speed while waiting for next communication. During time-critical periods, the MCU switches on the frequency-locked loop (bus clock is typically 8 MHz). The disadvantage of this solution is the transition from the Off to the Idle mode takes between 10 and 25 ms, due to the start-up of the voltage regulators and clock oscillator. In addition, the content of the modem RAM and the GPIO state is lost, so all the configurations must be repeated every time. Two crystals (16 MHz for Modem and 32.768 kHz for MCU) are needed.
Total consumption: 200 nA (Modem) + 95 µA (MCU)

b. MCU with 32.768 kHz crystal, Modem in Hibernate mode, has similar behavior as in the 1a) scenario. The transition time from Hibernate to Idle mode is somewhat shorter (8 to 20 ms). The nice thing about operating in this mode is that the content of the RAM and the GPIO states are retained. This method uses the same two crystals as before.
Total consumption: 1 µA (Modem) + 95 µA (MCU)

c. MCU clocked by internal oscillator or externally (via CLKO output provided by Modem), Modem in Doze mode the MCU provides the internal oscillator, and when the MCU is commanded to exit Doze mode, it can do so quickly. MCU resets and configures Modem. Modem can also provide the reference clock in the Doze mode. The MCU programs CLKO to the desired frequency and waits until CLKO stabilizes. Once the CLKO is stable the MCU switches its clock to external source. Precise timing of long time delays is done in Modem by time scheduled exit from the Doze mode, while MCU sleeps at the same time.
In this configuration, the designer can fully take advantage of automatic initiating the timer-triggered events such as:

- Activating receive mode
- Activating transmit mode
- Activating idle mode

In addition, the received frames are also "time-stamped" during reception, making a precise timing between various events possible. The microcontroller can remain in the low-power modes. This scenario has slightly higher power consumption than using a separate crystal for the MCU, but only single crystal (Modem crystal) is needed.

2. **Inaccurate packet timing**

a. MCU timing controlled by internal oscillator (RTI) and Modem in off mode. Disadvantage of this scenario is the communication timing is not that accurate. The MCU timing is controlled by a RTI internal oscillator that does not provide precise timing (only 30 percent). Thus the receiving windows must be wider which also affects power consumption negatively. Single crystal is required.
Total consumption: 200 nA (Modem) + ~1 µA (MCU Sleep mode + RTI enabled)

b. MCU timing controlled by internal oscillator (RTI) and Modem in Hibernate mode is similar scenario to a. Difference is in shorter transition time to idle mode and Modem re-initialization is not needed.
Total consumption: 1 µA (Modem) + 1 µA (MCU Sleep mode + RTI enabled)

c. MCU clocked by internal oscillator running at the lowest possible frequency (4 kHz), Modem in off mode.
The MCU bus clock is driven by an internal clock reference. It is 2 percent accurate (after trimming) which may be sufficient for many applications. The Modem is in off mode, single crystal is required.
Total consumption: 200 nA (Modem) + 150 µA (MCU running at 4 kHz internal clock)

d. MCU clocked by internal oscillator running at the lowest possible frequency (4 kHz) and Modem in Hibernate mode. This scenario is similar to c; the Modem is in Hibernate mode.
Total consumption: 1 µA (Modem) + 150 µA (MCU running at 4 kHz internal clock)

Table 2

Mode	MCU Clock	Accuracy	Modem Mode	Consumption
1a	Xtal 32768 Hz	100 ppm	Off	95 µA
1b	Xtal 32768 Hz	100 ppm	Hibernate	96 µA
1c	Clocked from Modem	20 ppm	Doze	36 µA
2a	RTI clock	30%	Off	1.2 µA
2b	RTI clock	30%	Hibernate	2 µA
2c	Internal clock 4 kHz	2%	Off	150 µA
2d	Internal clock 4 kHz	2%	Hibernate	151 µA

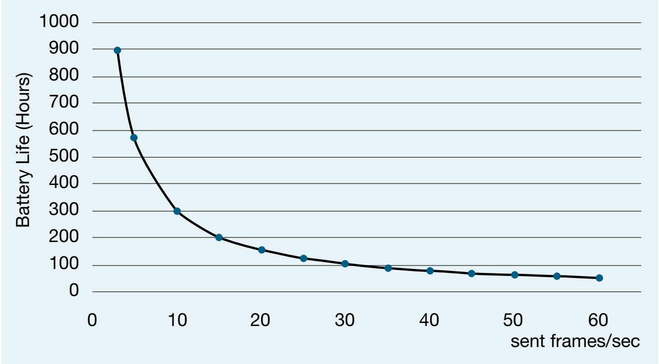
Another portion of total power consumption is drawn.

B. **Transmit and receive events**

As seen from the Modem transmit/receive operation mode supply current, Modem's transmitting and receiving state has a high influence on the power consumption. Therefore, the length of data packets needs to be highly optimized and communication rates must be selected carefully.

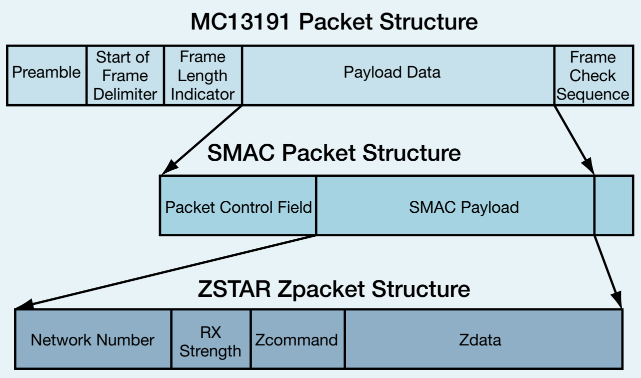
On the following charts you can see the influence of communication rates and data payload length versus the battery life. The first chart shows battery life in hours versus the number of frames sent per second. We had to compromise to 30 packet per second to gain sufficient sample rate together with reasonable battery life.

Figure 1: Battery Life Versus Packets/Sec.



Also the data packet length affects the battery life significantly. That is why we send only raw data from triaxial accelerometer (please see the packet format and packet length in Chapter 5.3 of ZSTARRM Reference Manual for further details).

Figure 2



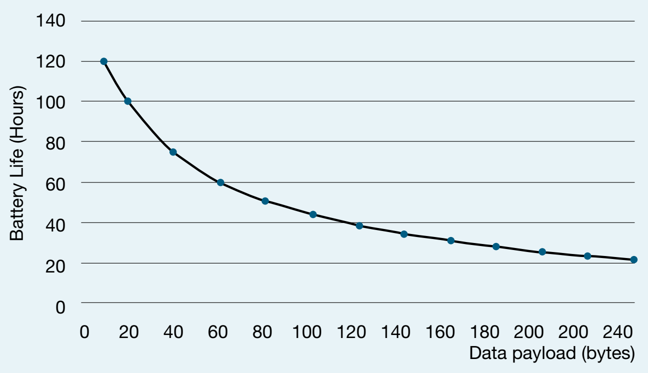
Standard IEEE 802.15.4-compliant packet has 4 bytes of preamble, 1 byte of SFD (Start of Frame Delimiter), 1 byte of FLI (Frame Length Indicator), 2 bytes of FCS (Frame Check Sequence) plus the payload data (125 bytes maximum). As a result, the overhead of a frame is 8 bytes or 8 x 32 = 256 µs, and the maximum payload transmit time is 125 x 32 = 4000 µs. The transmit time for a packet then is:

Total TX time (µs) = 256 + (payload bytes x 32)

Standard SMAC Packet for our sensor board has 17 bytes (including 2-byte packet IEEE 802.15.4 control field). There are also so called "warm up" (transition from Idle to Transmit) and "warm down" (Transmit to Idle) periods which take 144 µs and 10 µs respectively.

The following chart shows dependence of packet length versus battery life. It is obvious that packet length should be as small as possible.

Figure 3: Battery Life Versus Data Payload



This chart is also quite theoretical because there is another physical limitation in place. The power supply (Lithium battery) provides only very limited continuous current, so if a longer frame would be transmitted, the battery voltage will drop significantly. In the ZSTAR design, a large tantalum capacitor (470 µF) is designed to improve the response of the power supply to current peaks caused by reception or transmission. Even that, larger data frame (longer than ~50–100 bytes) would still cause a significant voltage drop.

Capacitor calculation: we specified allowable supply voltage drop 100 mV while transmit time (1 ms).

$$\Delta u = \frac{1}{C} * i * \Delta t$$

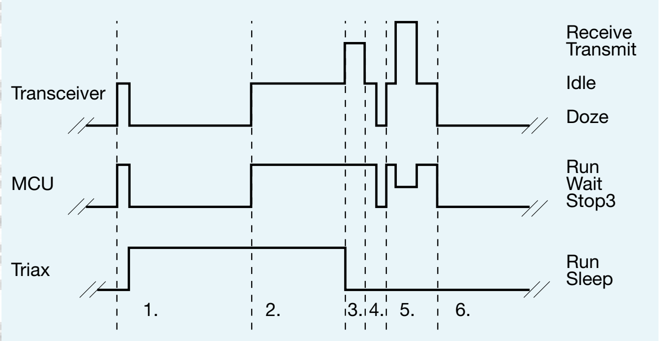
$$C_{min} = i\,\frac{\Delta t}{\Delta u} = 30\ mA * \frac{1\ ms}{100\ mV} = 300\ \mu F$$

So we finally chose 470 µF capacitor from the series in the same package.

How it Works Together

The sensor board was developed to be very small and cost-effective, which is why we used only one crystal at Modem. Also, we needed the accurate communication timing to use Modem Doze mode for communication timing, and the MCU uses internal clock (during its activity), otherwise the MCU is in the Stop3 mode. Communication between the sensor board and USB stick is described below.

Figure 4



Transmitting one data packet is processed.

1. At the beginning of the period, the Modem wakes up the MCU from Stop3 mode. The MCU awakes the Triax from sleep (asserting SLEEP pin of Triax). Then, the Triax outputs stabilize before a measurement can be done. During this time, the Modem is again turned into a timed Doze mode and the MCU also goes to the Stop3 mode.
2. When Triax is ready for measurement, Modem wakes the MCU up again and the MCU measures the Triax. After data is acquired by the MCU, the Triax is switched back to sleep mode, the data packet is formed and moved in to the Modem's RAM buffer.
3. The Modem transmits the packet over RF.
4. In order to detect the connection lost (between Sensor board and USB stick), the response from the USB stick is awaited. But because the USB stick needs some time to process received data, form a response packet, the MCU sets the Modem to the Doze mode for the short period again.
5. After that short period, the MCU wakes and response from the USB stick can be received within the specified receive window. Then, the Modem is set to Doze mode till the end of the cycle and the MCU goes to Stop3 mode.
6. Between two successive packets, MCU stays in Stop3 mode, Triax is switched to sleep mode and Modem is in Doze mode while its time base measures time to the next packet.

During the evaluation of this communication method we observed that the receiving after every transmission heavily increases the current consumption. Since this was not required or needed, the receive window for the response of the USB stick is only opened at every 8th opportunity. In all other instances, the sensor board goes to the Doze mode immediately after the transmission is finished (states 4 and 5 skipped).

Summary

The original goal of the ZSTAR project (to transfer real-time data using limited power supply) was evaluated. All aspects of the system design (both hardware architectures and also the software implementation) were explored and optimized for the best performance of the application. XYZ samples from the triaxial accelerometer, together with internal temperature and battery voltage level, are sent 30 times per second using a tiny lithium CR2032 battery for ~100 hours of run-time. The average current consumption of this system is estimated to be 2.5 mA and the measurements confirmed this expectation.

Overall, the ZSTAR design shows cost-effective implementation of the MC1319x Modem family with a cost effective low-pin 8-bit microcontroller (MC9S08QG family) for wireless sensing of 3-axis accelerometer (MMA7260QT).

Radomir Kozub is an application engineer at Freescale and has been with the company for almost four years. He has a master's degree in electrical engineering. Before joining Freescale, Kozub worked on development of various metering applications.

Pavel Lajsner is an application engineer at Freescale and has been with the company for more than eleven years. He holds a master's degree in electrical engineering.

David Baca

Efficient Control of JTAG TAP

JTAG TAP stepping made easy

Introduction

Many of Freescale's microprocessors feature a JTAG port–a port compatible with the IEEE® 1149.1 standard for performing boundary scans and debugging embedded applications. The development of the IEEE 1149.1 standard was started by the Joint Test Action Group in mid 80s, and the activity was later moved under the IEEE organization. As a result, a test access port compliant with the IEEE 1149.1 standard is usually called a JTAG port after the group that initiated the standardization process.

Today, the JTAG port is used for chip boundary scans and programming and debugging purposes. Any JTAG implementation compatible with the IEEE 1149.1 standard is not limited to the functions specified by the standard—it can freely extend functionality. For example, the IEEE-ISTO 5001 Forum™ standard defining a debug interface for embedded processors based its interface on JTAG.

The IEEE 1149.1 standard defines a general-purpose test access port (TAP) as a physical layer for accessing JTAG functions. Further, the standard specifies the TAP controller that controls the signals used for accessing JTAG functions and other possible incorporated circuitries such as a debugging module. The TAP controller is a synchronous finite state machine that responds to the changes at the TAP pins. To access a particular JTAG function, you have to step the TAP controller through its states. For learning the purpose of each TAP controller state and other JTAG specification details, refer to the IEEE 1149.1 standard.

This article presents a simple and efficient algorithm for stepping the TAP controller along the shortest path from the current TAP state to any other of its states. Programmers usually step through the TAP controller by writing sequences of instructions manipulating the TAP input pins. As a result, writing the stepping code for a comprehensive software module mediating JTAG's and associated functionalities can easily become a difficult, tedious and error-prone task. Our algorithm eliminates the sequences of TAP controlling instructions by a routine that navigates the TAP controller to the desired state. As a result, you just call the routine, for example `GoTo(dstate)` and the TAP controller is stepped to the desired state `dstate`.

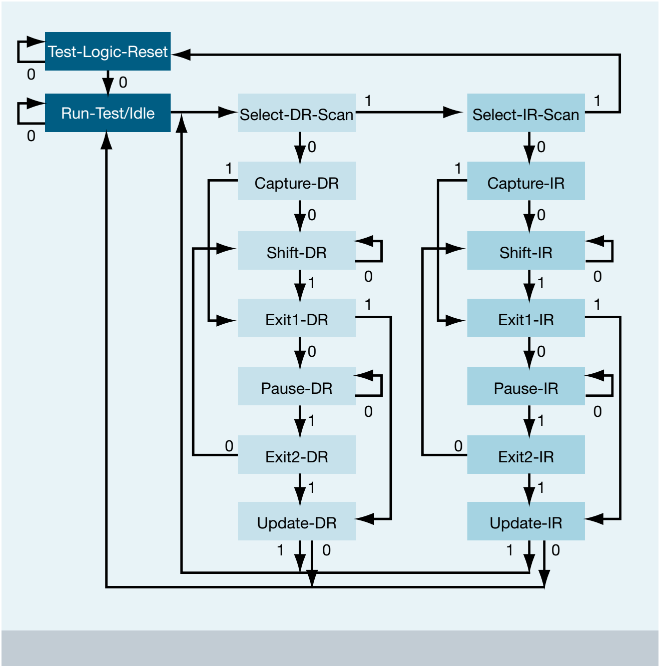
The Stepping Algorithm

In the TAP controller state machine, depicted in Figure 1, a transition is followed at the time of the clock rising edge (pin TCK) if it is enabled by the logical signal level (0 or 1) at the TAP TMS pin and if the TAP is in the transition's initial state. The arrow labels 0 and 1 in Figure 1 represent the enabling logical signal levels.

The diagram in Figure 1 can be viewed as graph consisting of vertices and oriented edges where a vertex is a TAP controller state and an oriented edge is a transition. Finding the shortest path between any two vertices is a classical, solved problem. In the TAP state machine, the shortest paths are quite obvious and can be found manually. Thus, we need an efficient mechanism for storing the shortest paths, so they can be followed without being discovered again.

For storing the shortest paths, we utilize the following corollary. Suppose there is the shortest path from a state s_1 to a state s_n and the path goes through a state s_k. Thus, the path is $s_1,\ldots,s_k,\ldots s_n$. Then, the path $s_k,\ldots,s_n$ is the shortest path between the states s_k and s_n. If this path wasn't the shortest,

Figure 1: TAP Controller State Diagram



then we could shorten the path between s_1 and s_n and the original path wouldn't be the shortest, which is a contradiction.

As a result, it is sufficient for any of the shortest paths starting at s_1 and ending at s_n to store the very next state following the state s_1 on the path. The repository that stores the very next state for each of the shortest paths is called a routing table, similarly defined in networks to route messages. Unlike networks where every node stores a vector of the very next nodes, our routing table is a matrix because the traversing is controlled via the TAP interface. The TAP routing table is depicted in Table 1.

Table 1: TAP Routing Table

States	Test-logic-reset	Run-test/idle	Select-DR-scan	Capture-DR	Shift-DR	Exit1-DR	Pause-DR	Exit2-DR	Update-DR	Select-IR-scan	Capture-IR	Shift-IR	Exit1-IR	Pause-IR	Exit2-IR	Update-IR
Test-logic-reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Run-test/idle	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Select-DR-scan	1	1	x	0	0	0	0	0	0	1	1	1	1	1	1	1
Capture-DR	1	1	1	x	0	1	1	1	1	1	1	1	1	1	1	1
Shift-DR	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
Exit1-DR	1	1	1	1	0	x	0	0	1	1	1	1	1	1	1	1
Pause-DR	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
Exit2-DR	1	1	1	1	0	0	0	x	1	1	1	1	1	1	1	1
Update-DR	1	0	1	1	1	1	1	1	x	1	1	1	1	1	1	1
Select-IR-scan	1	1	1	1	1	1	1	1	1	x	0	0	0	0	0	0
Capture-IR	1	1	1	1	1	1	1	1	1	1	x	0	1	1	1	1
Shift-IR	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Exit1-IR	1	1	1	1	1	1	1	1	1	1	1	0	x	0	0	1
Pause-IR	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
Exit2-IR	1	1	1	1	1	1	1	1	1	1	1	0	0	0	x	1
Update-IR	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	x

The algorithm that steps the TAP controller via the shortest path to the desired state is very simple. Suppose that the current TAP state is maintained in the variable `tap_state` and the two dimensional array `RoutingTable` holds the content of Table 1. The function `GoTo(dstate)` works as follows.

```
GoTo(dstate)
{
   while(tapstate != dstate)
      step(RoutingTable[tapstate][dstate]);
   return;
}
step(tms)
{
      set TMS pin to tms (typically on a parallel port);
   generate clock pulse on the TCK pin;
   tap_state = next_state(tap_state,tms); //
update the current state
}
```

The function `next_state(tap_state,tms)` uses another table that provides the very next successor for each state given the TMS pin value.

Implementation

We implemented the described algorithm in C++. A singleton class called `JTAG` controls the signals for the TAP controller, maintains the current state, and steps the TAP controller to the desired state. When the class is being instantiated, the routing table along with other tables is loaded in memory and the TAP controller is reset to its initial state test-logic-reset. The `JTAG` singelton interface provides a method called `AssertTAPState(tapstate)` to make sure the current TAP controller state is `tapstate`. If the current state is different, the method steps the TAP controller to the right state.

The `JTAG` singleton steps the TAP controller by manipulating the pins of the parallel port that is connected to the JTAG physical interface on the embedded processor. The methods that control the JTAG pins via the parallel port are implemented in another singleton called `JTAGPort`, which is accessed solely from `JTAG`.

Conclusion

We showed that writing the tedious sequence of instructions stepping the TAP controller can be eliminated by keeping the shortest paths between any two TAP controller states in a small routing table. The associated overhead caused by performing methods or function calls and by looking up the TMS pin value is negligible considering the current computer's speed and the maximal frequency achievable at the parallel port (around 1 MHz). Furthermore, the JTAG port communication speed is limited by the processor clock frequency.

David Baca is an application engineer at Freescale. He has been with the company for almost two years. He holds masters degrees in electrical engineering and business administration and is a PhD candidate in artificial intelligence. Before joining Freescale, Baca worked on R&D projects sponsored by agencies such as NASA and the Air Force.

Lech Olmedo and Daniel Torres

Accelerating Innovation
With ColdFire® MCF5223x embedded controllers

Introduction

In today's fast-paced world, the need to save time and have every part of our lives in control is becoming increasing important. Technology continues to play a significant role in the way our world is changing. At home, almost every single appliance is embedded with a technology that is constantly evolving. Connectivity is an example of how technology is always changing to provide new solutions to make our life easier, faster or more productive.

Design Challenge

Ethernet opened a new world of possibilities for human beings by allowing access to a worldwide network from any place on Earth that lets us be in touch with our daily activities and home duties. With Ethernet connectivity, a person can remotely control the temperature or lighting in their home from their computer while at the office.

The ability to automate houses and buildings has recently been exploited due to the introduction of low-cost Ethernet-enabled devices. High-end appliances such as refrigerators connected to Ethernet, alarm systems, cameras, door and window locks, irrigation systems and air-conditioning systems are just a few examples of how this enabling technology is making life easier.

Freescale Semiconductor is committed to making life easier and safer which is why many of the new low-end 32-bit ColdFire devices are being introduced to enable the remote control of home appliances, industrial process, safety and security systems. Freescale gives development engineers the flexibility to choose the right 32-bit microcontroller from a broad portfolio of ColdFire embedded controllers.

Freescale Solution

Many applications may be found for control-over-Ethernet, such as home appliances and industrial control. All of them use Ethernet as the main communication protocol and a controller capable of driving the application and connectivity tasks. The new low-cost ColdFire MCF5223x family devices from Freescale are well suited for such requirements of control and connectivity.

MCF5223x Overview

The ColdFire MCF5223x family is part of the new low-end side of ColdFire devices and is built upon the legacy of the popular 68 KB family. To help ensure secure connectivity, the MCF5223x family of ColdFire devices is a single-chip solution that provides 32-bit control with an Ethernet interface. It combines a 10/100 Fast Ethernet Controller (FEC), Ethernet Physical Layer (EPHY), CAN 2.0B and a Cryptographic Acceleration Unit (CAU) for secure hardware encryption with the Version 2 ColdFire core to provide exceptional performance at a reasonable cost (up to 57 Dhrystone 2.1 MIPS at 60 MHz). The MCF5223x embedded controller provides the designer with the right set of peripherals

Figure 1: MCF5223X Block Diagram

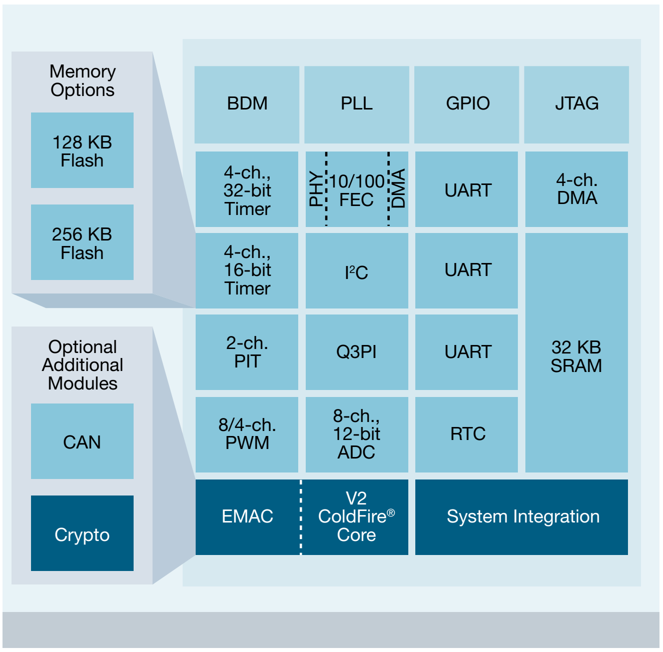
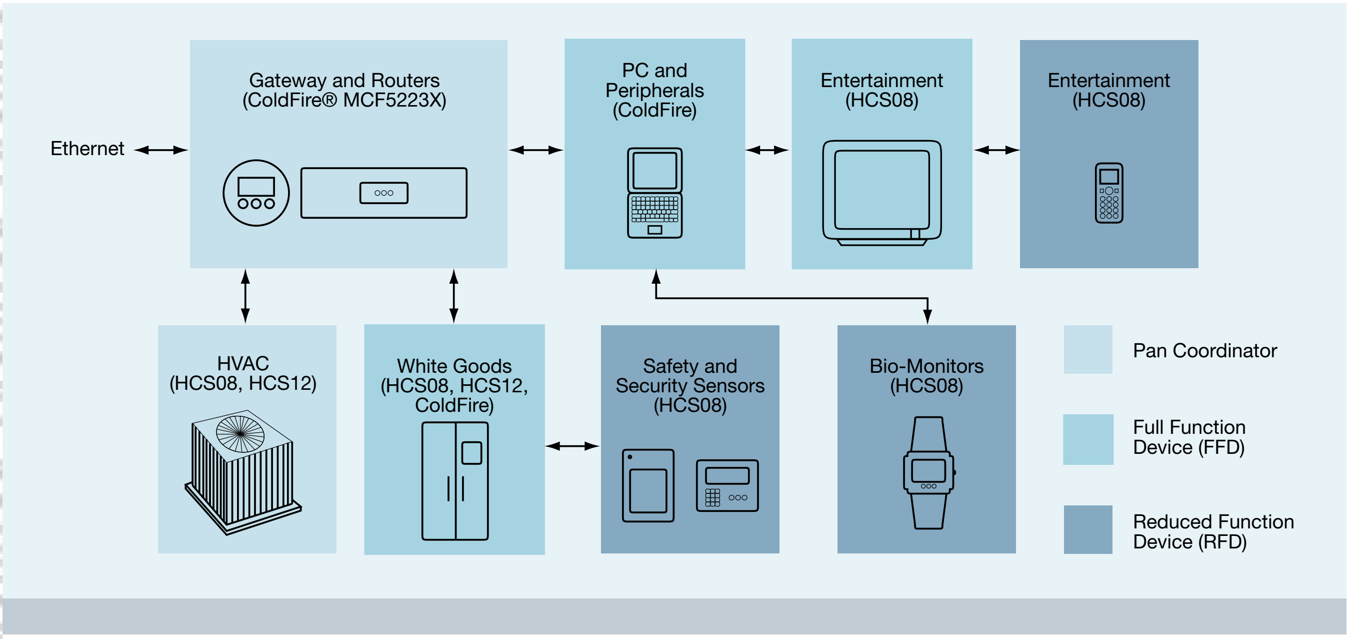


Figure 2: Typical ZigBee® Network


Ethernet
Gateway and Routers (ColdFire® MCF5223X)
PC and Peripherals (ColdFire)
Entertainment (HCS08)
Entertainment (HCS08)
HVAC (HCS08, HCS12)
White Goods (HCS08, HCS12, ColdFire)
Safety and Security Sensors (HCS08)
Bio-Monitors (HCS08)
Pan Coordinator
Full Function Device (FFD)
Reduced Function Device (RFD)

and memory sizes (up to 32 KB SRAM and up to 256 KB flash: 100 KB W/E cycles,10 years data retention) for a compact Ethernet-enabled platform that cuts development time and cost to help your products get to market quicker.

The MCF5223x family integrates standard ColdFire peripherals, including three universal asynchronous receiver/transmitters (UARTs) for medium and long distance connections, an inter-integrated circuit (I²C) and a queued serial peripheral interface (QSPI) for in-system communications to connected peripherals.

For digital control the MCF5223x family has the following peripherals: four 32-bit timer channels with DMA capability, 4-channel, 16-bit or 8-channel, 8-bit PWM generator, two periodic interrupt timers (PITs) for alarm and countdown timing, 4-channel DMA controller, 8-channel, 12-bit ADC, up to 73 general-purpose I/Os and system integration modules such as PLL and COP.

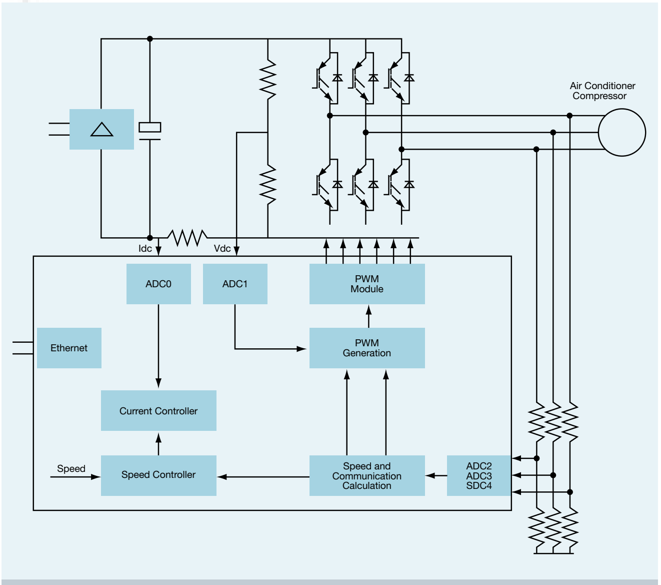
These devices widen the migration bridge for existing 8- and 16-bit users who need a low-cost, entry-level 32-bit solution that incorporates 32-bit performance, fast Ethernet features and comprehensive tools support. The Freescale ColdFire roadmap to high-end 32-bit performance and connectivity is designed to allow customers to migrate their applications easily and quickly.

ColdFire MCF5223x as Master of the ZigBee® Concert

Imagine being able to control your home lights or sensor and watch your locks from your PC. This kind of network control is not as difficult as it may seem—requirements include obtaining a wireless communication and a gateway to Ethernet. Freescale provides the wireless solution and the controller solution as well. The M13192 devices are transceivers at 2.4 GHz Band, -92 dBm RX sensitivity at 1 percent PER, power supply 2.0–3.6V w/ on-chip regulator, logic interface 1.7–3.3, Runs off a single Li or two alkaline cells, complete RF transceiver data modem—antenna in, fully packetized data out, data and control interface via standard SPI at four MHz minimum, 802.15.4 MAC is supplied with MC13192.

This transceiver can be easily attached to the MCUs by SPI. Attaching transceivers to sensors on lock, light interrupters or any device that wants to be monitored or controlled. Those devices should have a tag and be communicated as devices to a central transceiver that is connected to a MCF5223x controller. The MCF5223x device is able to be connected to the web allowing monitoring of every device and even proceeding with some actions when needed (use of the free stack from InterNiche). Figure 2 shows a typical ZigBee network at home.

Figure 3: Ethernet Enabled Air-Conditioner Controller



ColdFire MCF5223x as the Gateway to Monitor and Diagnostics

Monitoring any industrial line and showing the result into an inner net automatically is also a common task. The monitoring live system could improve times and systematically fill information in databases directly to systems without any human intervention. This will allow controlling process within the production line remotely, and if a problem is encountered, the technical support team could decide which corrective actions to perform.

ColdFire MCF5223x for Air Conditioner Control and Monitoring

For home appliances as well as industrial applications, the control and monitoring of an air conditioner could be a matter of comfort and productivity. The MCF5223x is ideal for these devices. Usually air conditioners are controlled by DSPs or DSCs that are oriented to motor control applications, however the MCF5223x offers the ability to handle the control of these devices and adds the Ethernet connection to a remote monitoring. The MCF5223x contains PWM modules, timers and a 12-bit ADC that are just the right peripherals to adjust motor control speed and torque. It also includes Ethernet control and contains the physical layer, plus a stack providing the full functionality to diagnose, monitor and remotely control this device.

Conclusion

Connecting 32-bit controlled applications in the industrial, commercial and even consumer markets is quickly becoming a necessity rather than an option. Freescale's ColdFire MCF5223x family helps ensure secure connectivity by providing a single-chip solution for 32-bit control with an Ethernet interface. These cost effective embedded controllers provide the right set of peripherals and memory sizes for a compact Ethernet-enabled platform that cuts development time and cost to help your products get to market faster.

Lech Olmedo is a go-to-market application engineer in Freescale's RTAC. He supports the ColdFire Family and is involved in Connectvity solutions for ColdFire and Linux (uClinux) applications. His collateral creation includes 8- to 16-bit automotive applications migration.

Daniel Torres is an applications engineer at Freescale Semiconductor; he is experienced in Digital Signal Controllers, ColdFire processors and 8-bit MCUs. He is focused on motor control and power management areas.

David Baca

Fast 32-bit Division on the DSP56800E
Minimized nonrestoring division algorithm

Introduction

The processor core DSP56800E features several 32-bit registers. It may appear that performing a 32-bit integer division should be a simple task, however, if you're not an expert on binary division, you may need more support. DSP56800E uses an instruction called DIV to perform a division. The instruction represents a divide iteration calculating 1 bit of the division result. The instruction description along with 16-bit division algorithms can be found in the DSP56800E Reference Manual.

DIV accepts a 32-bit dividend, a 16-bit divisor and produces a 16-bit quotient and a 16-bit remainder after 16 iterations. The iteration step operates on two 32-bit operands where the 16-bit divisor occupies the upper 16 bits of the operand. The algorithm used for calculating the quotient and the reminder is Nonrestoring Division Algorithm (NrDA).

If we were to implement NrDA for 32-bit division, we would need 64-bit operands. Implementing NrDA with 64-bit operands on a 16-bit processor such as the DSP56800E is cumbersome because it needs support for 64-bit arithmetic. Another option is to apply the well known Newton-Rhapson iterative method that requires only 32-bit arithmetic. Nevertheless, this method employs division and therefore requires an abundance of tweaking to achieve the desired accuracy. Furthermore, we can't be sure that all pairs (dividend, divisor) have the correct result calculated unless the implementation is tested for almost all possible input pairs.

Minimized nonrestoring division algorithm (MNrDA) efficiently implements 32-bit non-restoring algorithm on 32-bit arithmetic. Since the DIV instruction on the DSP56800E also accepts the divisor as a 32-bit number, we can implement the NrDA algorithm effectively.

This article will:

- Introduce the fundamental restoring division algorithm
- Explain the nonrestoring division from the restoring algorithm
- Describe how to eliminate unnecessary iterations from NrDA to arrive at the minimized version of NrDA
- Provide an example implementation of MNrDA for the processor core DSP56800E

The Nonrestoring Division Algorithms

Integer division notation

Integer division can be defined as follows:

$$N = Q \bullet D + R$$

Where

- N is the numerator (dividend)
- D is the denumerator (divisor)
- Q is the quotient
- R is the remainder

For the sake of simplicity, we assume that N, D, Q is positive. R is positive by definition and $0 \leq R < D$. Further, each operand can be dismantled into individual bits as follows $Q = \sum_{j=0}^{m-1} Q_j \cdot 2^j$.

Restoring Division Algorithm

Nonrestoring Division Algorithm (NrDA) comes from the restoring division. The restoring algorithm calculates the remainder by successively subtracting the shifted denominator from the numerator until the remainder is in the appropriate range. Now we can derive the algorithm for restoring division of *m*-bit integers.

The algebraic equation of the remainder calculation is:

$$R = N - (D \cdot Q)$$

We dismantle Q into bits.

$$\begin{aligned} R &= N - D \cdot \sum_{j=0}^{m-1} Q_j \cdot 2^j \\ &= N - D \cdot \sum_{j=0}^{m-1} Q_j \cdot D \cdot 2^j \end{aligned}$$

This can be viewed as subtracting denominator shifted by j bits left, $D \cdot 2^j$, from N for each $Q_j = 1$. The subtraction can only occur if it still yields a positive reminder ($Q_j = 1$) given the previous subtractions. As a result, we can formulate the restoring algorithm as follows. We set:

$R(m) = N$ where m is the size of integers in bits, for example 16, 32

We then calculate *R(j)* and *Qj* for *j* = *m* − 1 . . . 0 as follows:

$$
\begin{array}{ll}
Z = R(j+1) - D \cdot 2^j & \\
\text{if } Z \geq 0 & Q_j = 1,\ R(j) = Z \\
\text{else} & Q_j = 0,\ R(j) = R(j+1)
\end{array}
$$

After calculating *R*(0), *R*(0) is the actual remainder and *Q* contains the resulting quotient.

In order to carry out the restoring algorithm, we need either the temporary variable *Z* or we have to restore the remainder at each step the subtraction yields a negative result. This is because we are determining Q_j and *R(j)* calculation in one step. The nonrestoring algorithm eliminates this concurrency by determing *R(j)* calculation from Q_{j+1}.

Nonrestoring Division Algorithm

In the nonrestoring algorithm, we use the negative result for the next iteration. To formulate the algorithm we start from the restoring algorithm. Suppose that we obtain a negative remainder at the *k*-th step.

$$R(k) = R(k+1) - D \cdot 2^k$$

Obviously $Q_k = 0$ and therefore at the very next step we should compute:

$$R(k-1) = R(k+1) - D \cdot 2^{k-1}$$

By using the result *R(k)* and for $Q_k = 0$ we use addition.

$$
\begin{aligned}
R(k-1) &= R(k) + D \cdot 2^{k-1} \\
&= R(k+1) - D \cdot (2^k - 2^{k-1}) \\
&= R(k+1) - D \cdot 2^{k-1}
\end{aligned}
$$

Hence, we formulate the iteration in the non-restoring algorithm as follows:

$$
\begin{array}{ll}
R(j) = R(j+1) - D \cdot 2^j & \text{if } Q_{j+1} = 1 \\
R(j) = R(j+1) - D \cdot 2^j & \text{if } Q_{j+1} = 0 \\
Q(j) = 1 & \text{if } R(j) \text{ is positive} \\
Q(j) = 0 & \text{if } R(j) \text{ is negative}
\end{array}
$$

We combine the two equations above to express the iteration step algebraically as:

$$R(j) = R(j+1) + (1 - 2 \cdot Q_{j+1}) \cdot D \cdot 2^j$$

We start the algorithm by initializing *R(m)* = *N* and $Q^m = 1$. After performing *m* – 1 iterations we obtain *R*(0).

$$R(0) = N - \left(2^{m-1} - \sum_{j=0}^{m-1} (1 - 2 \cdot Q_{j+1}) \cdot 2^j\right) \cdot D$$

$$R(0) = N - \left(2^0 - \sum_{j=0}^{m-1} Qj \cdot 2^j\right) \cdot D$$

Clearly, *R*(0) is not the correct actual remainder. Therefore, at the end of the algorithm we need to perform one additional calculation to obtain the remainder.

$$R = R(0) + (1 - Q_0) \cdot D$$

Finally, we can summarize the non-restoring algorithm as follows:

```
              $R(m) = N,\ Q_m = 1$ (initial state)
for $j = m - 1$ to $0$
              $R(j) = R(j+1) - D \cdot 2^j$ if $Q_{j+1} = 1$
              $R(j) = R(j+1) + D \cdot 2^j$ if $Q_{j+1} = 0$
              $Q_j = 1$ if $R(j) \geq 0$
              $Q_j = 0$ if $R(j) < 0$
         end for
                $R = R(0) + \neg Q_0 \cdot D$
```

The nonrestoring algorithm requires shifting *D m* – 1 times. In particular for 32-bit division, *D* has to be shifted 31 bits left. So, 64-bit arithmetic is needed to perform NrDA. In general, one needs (2**m*)-bit arithmetic to execute NrDA for *m*-bit division.

Minimized Nonrestoring Division Algorithm

Minimized Nonrestoring Division Algorithm (MNrDA) is designed to execute NrDA for *m*-bit numbers on m-bit arithmetic.

Proposition 1 If *n*, *d* and *q* are the indices of the first significant bits of *N*, *D* and Q respectively, then *n* – *d* ≥ *q*,

Proof of proposition 1: For *m*-bit integers we have:

$$N = Q \cdot D + R$$

$$\sum_{j=0}^{m-1} N_j \cdot 2^j = \sum_{j=0}^{m-1} D_j \cdot 2^j \cdot \sum_{j=0}^{m-1} Q_j \cdot 2^j + R$$

$$2^n + \sum_{j=0}^{n-1} N_j \cdot 2^j = \left(2^d + \sum_{j=0}^{d-1} D_j \cdot 2^j\right) \cdot \left(2^q + \sum_{j=0}^{q-1} Q_j \cdot 2^j\right) + R$$

The condition *n* – *d* ≥ *q* immediately follows.

Corollary 1 Let *N* be the numerator and *D* be the denominator of an integer division. Let *n* and *d* be the indices of the first significant bits of *N* and *D*, respectively. Then $Q_j = 0$ for all *j* > *n* – *d*.

Proof of corollary 1: The corollary is a direct result of proposition 1

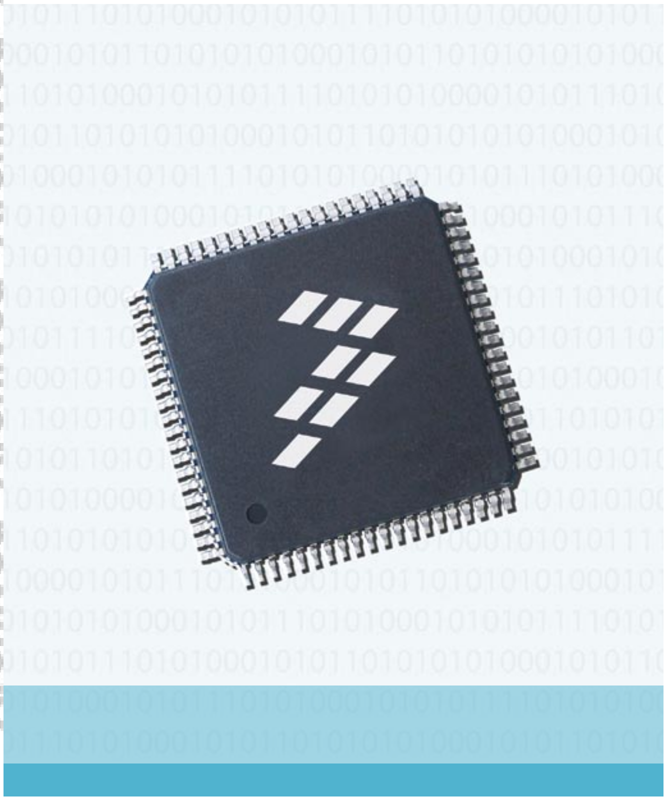
If we apply corollary 1 to NrDA, we can jump in the algorithm to $R(n-d)$ because previous iterations resulted in the quotient bits of the value 0.

$$R(n-d) = N - D \cdot 2^{m-1} + \sum_{j=n-d}^{m-2} D \cdot 2^j$$
$$= N - D \cdot 2^{n-d}$$
$$= N - 2^{n-d} \cdot (2^d + \sum_{j=0} D_j \cdot 2^j)$$
$$= N - (2^n + \sum_{j=0}^{d-1} D_j \cdot 2^{j+n-d})$$

Since the first calculated partial remainder is $R(n-d)$, we only need to shift D $n-d$ times effectively moving the first significant bit of D to position n. Hence, since $m-1 \geq n$, we need only m bits to store shifted D.

MNrDA doesn't decrease the number of iterations for the worst case scenario ($n = m - 1$, $d = 0$). Since timing analyses are typically based on the worst case execution times, we can simply assume $n = m - 1$. Thus, MNrDA is performed as follows:

1. Calculate the leading zeroes l_0 of D.
2. Shift D left by l_0 bits.
3. Execute Nonrestoring Division Algorithm starting from R(l_0) and setting $Q_j = 0$, $j = l_0 + 1 \ldots m - 1$



MNrDA Implementation for Signed Division on the DSP56800E

The DIV instruction on the DSP56800E represents the iteration of 16-bit restoring division. The instruction accepts 32-bit dividend and 16-bit divisor. The divisor is applied as shifted 16 bits left and instead of shifting the divisor right in each iteration, the dividend is shifted left. The resulting bits Q_j are shifted in during each iteration. As a result, after performing 16 iterations, the remainder $R(0)$ is contained in the upper 16 bits of the dividend. This can be algebraically illustrated as follows. One iteration step is:

$$\begin{aligned} R' &= 2 \cdot R'(j+1) \pm D \cdot 2^n \\ &= 2^{n-j} \cdot R(j+1) \pm D \cdot 2^n \\ &= 2^{n-j} \cdot (R(j+1) \pm D \cdot 2^n) \\ &= 2^{n-j} \cdot R(j) \end{aligned}$$

This ultimately leads to:

$$R'(0) = 2^n \cdot R(0)$$

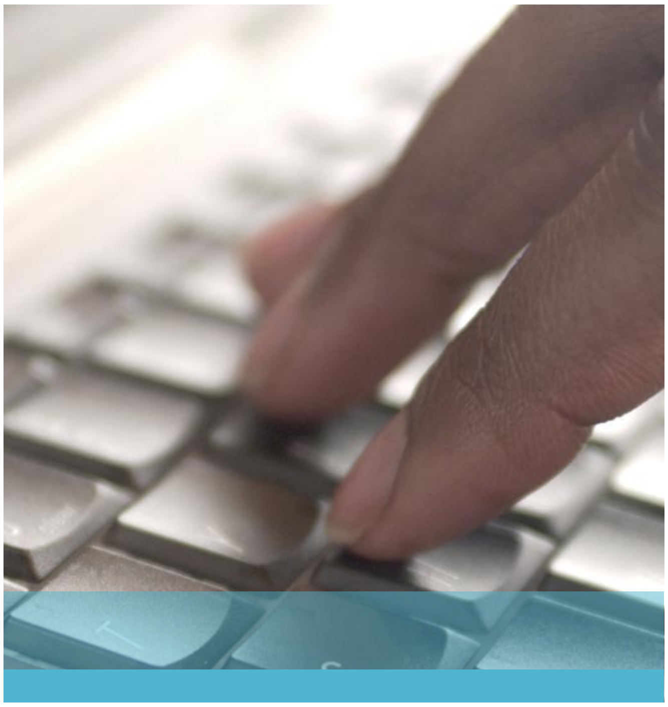
In our case $n = 16$ and therefore $R(0)$ is contained in the upper 16 bits. The quotient occupies the lower 16 bits.

In order to perform full 32-bit division, we supply the DIV instruction with a 32-bit divisor shifted $n - d$ bits left where n and d are the most significant bits of the dividend and the divisor respectively.

$$\begin{aligned} R'(n-d) &= R(n-d) \\ &= N - D \cdot 2^{n-d} \\ R'(n-d-1) &= 2 \cdot R(n-d) - D \cdot 2^{n-d} \\ &= 2 \cdot (R(n-d) - D \cdot 2^{n-d-1}) \\ &= 2 \cdot R(n-d-1) \\ &\vdots \\ R'(n-d-j) &= 2^j \cdot R(n-d-j) \\ &\vdots \\ R'(0) &= 2^{n-d} \cdot R(0) \end{aligned}$$

For this reason, $R(0)$ is contained in the upper $m - 1 - (n - d)$ bits and the quotient in the lower $n - d$ bits. If we put $n = m - 1$, then $R(0)$ resides in the d upper bits with its LSB at the (n-d)th bit. We apply mask to the DIV destination operand in order to retrieve the quotient. In our case, $m = 31$ as we assume positive dividend and divisor for the actual division. We present four-quadrant division employing MNrDA that produces the quotient. Since the complexity of retrieving the remainder from MNrDA is more or less equivalent to multiplying the quotient and subtracting it from the dividend, we don't produce the remainder. The worst case time execution of this algorithm is 60 cycles.

```
;  A = signed dividend (numerator)
;  B = signed devisor (denominator)
;  A1:A0 / B1:B0
;  The routine uses the following registers : A, B, D, X0, Y

   tfr     B,Y        ;  Y=den

   ;  the quotient sign
   eor.l   A,B        ;  MSB of B holds the sign bit

   ;  prepare R(q) and den
   abs     A          ;  abs (num)
   abs     Y          ;  abs (den) , so we can calculate R(q)
                      ;  DIV accepts negative S but for
                      ;  R(q) calculation we hardcode subtraction

   ;  shifted denominator and the first bit
   clb     Y,X0       ;  calculate leading zeroes
   asll.l  X0,Y       ;  Y=den*2^q
   sub     Y,A        ;  A=R(q), Carry=1 if Y>A

   ;  calculate the bit q; 1 if R(q)>=0
   bftstl  #$8,A2     ;  set Carry if positive

   ;  divide loop
   rep     X0         ;  calculate the bits q–1. . . 0
   div     Y,A        ;  Carry has the least significant bit

   rol.l   A          ;  last Carry has to be rolled in
                      ;  DIV would destroy the highest bit

   ;  mask off bits 0. . . q–1
   eor.l   Y,Y        ;  clear Y
   add.l   #2,Y       ;  for X0=0, we calculate the bit 0
   asll.l  X0,Y
   dec.l   Y          ;  Y contains the mask
   and.l   Y,A        ;  apply the mask

   ;  apply the sign bit
   brclr   #$8000, B1, Qpositive
   nop
   neg     A
Qpositive:
```



Conclusion

In this article, we showed that Freescale processors with the DSP56800E core are capable of performing a full precision 32-bit division. Since the DSP56800E DIV instruction can accept 32-bit number as the divisor, we can implement the minimized version of the nonrestoring division algorithm (MNrDA) to carry out 32-bit division. The DIV instruction was originally designed for 16-bit non-restoring division where the divisor initial shift is automated to full 16 bits.

Unlike with 32-bit division, applying MNrDA to 16-bit division would rather increase the worst case execution time as the reduction of the number of iteration is more or less consumed by calculating and performing the initial divisor shift. On the other hand, for 32-bit division we are running out of the register sizes on a 16-bit processor; therefore MNrDA is the best option. Another benefit of this option is that the effect of the reduction of iterations is leveraged by the fact that 32 iterations are needed for the pure NrDA.

David Baca is an application engineer at Freescale. He has been with the company for almost two years. He holds masters degrees in electrical engineering and business administration and is a PhD candidate in artificial intelligence. Before joining Freescale, Baca worked on R&D projects sponsored by agencies such as NASA and the Air Force.

Daniel Torres

Motor Control and Power Supply Design Using BLDC, PMSM and SMPS

Introduction

Nowadays, most of the devices that help make life easier and comfortable—things like washing machines, refrigerators, fans, air conditioners, power tools, blenders—require motion control.

All of these devices consume energy to produce motion. The way this energy is efficiently used depends on the control systems, electric machine design, control algorithms, etc. One of the biggest challenges that we face as a human race is the efficient use of energy and most of the efforts to improve this challenge are targeted at motion control systems. Hence, many energy-saving advances are coming from improved motor control techniques, frame design, materials and manufacturing precision.

More efficient control techniques were developed years ago, but the required CPUs to perform such complex algorithms and computations were too expensive for cost-sensitive markets like the appliance market. This situation has changed in recent years; high-performance digital signal controllers have been developed at lower costs with all the required features to perform these complex control algorithms.

Another area where energy-saving advances are being developed is power conversion. Power conversion systems are used to convert electric power from one form to another. During this process a certain amount of energy is lost due to the inherent power consumption of the system, efficiency of the topologies, the control techniques and the electronic devices used. Most of the power conversion controls are performed by analog circuitry, but the requirements, of the new energy-saving regulations have increased and frequently have become harder to meet with analog control systems.

The use of microcontrollers (MCU) and DSCs has opened new frontiers in this regard. Today, it is feasible to achieve 98-percent efficiency in power conversion systems through the use of digital control techniques and complex mathematical computations executed by high-performance low-cost digital signal controllers.

Motion Control Design Challenges

Several kinds of motors are used in motion control, including brushless DC motors (BLDC), brush-commutator permanent magnet DC motors, linear motors and stepper motors.

System engineers must not only choose the right kind of motor for the mechanical task, but they must also choose the appropriate control loop scheme for encompassing both the mechanical and electrical time-variant responses of the system. Tuning this control loop is often done in the design stage of the drive electronics.

Developers face a number of design variables because each type of motor has a unique set of requirements for the drive electronics. Designing drive electronics is further complicated by the electric motors themselves, which, by their inductive nature, are prone to producing electromagnetic interference (EMI), radio frequency interference (RFI) and destructive high-energy transients. Drive electronics design must prevent EMI and RFI, while still withstanding transient over-voltage and over-current conditions.

BLDC motors are very popular for many applications. A BLDC motor does not have a commutator and is more reliable than a DC motor. A BLDC motor has other advantages over an AC induction motor. BLDC motors achieve higher efficiency by generating the rotor magnetic flux with rotor magnets. They are used in high-end home appliances (such as refrigerators, washing machines and dishwashers), high-end pumps, fans and other devices that require high reliability and efficiency.

BLDC motors are widely used in pump, fan and compressor applications because of their robust structure. A common feature of these applications is that they do not require position information; only speed information is required, and only to perform control. BLDC motors can be used without complex control algorithms.

In the BLDC motor, the rotor position must be known to energize the phase pair and control the phase voltage. If sensors are used to detect rotor position, then sensed information must be transferred to a control unit.

This requires additional connections to the motor, which may not be acceptable in some applications. It may be physically impossible to make the required connections to the position sensors. Also, the additional cost of the position sensors and the wiring may be unacceptable. The physical connection problem could be solved by incorporating the driver in the motor body; however, a significant number of applications do require a sensorless solution due to their low-cost nature.

A permanent magnet synchronous motor (PMSM) provides rotation at a fixed speed in synchronization with the frequency of the power source, regardless of the fluctuation of the load or line voltage. The motor runs at a fixed speed synchronous with mains frequency, at any torque up to the motor's operating limit. PMSMs are therefore ideal for high-accuracy fixed-speed drives.

A three-phase PMSM is a permanently excited motor. Boasting very high-power density, very high efficiency and high response, the motor is suitable for most sophisticated applications in mechanical engineering. It also has a high overload capability. A PMSM is largely maintenance-free, which ensures the most efficient operation.

Precise speed regulation makes a PMSM an ideal choice for certain industrial processes. PMSMs have speed/torque characteristics ideally suited for direct drive of large-horsepower, low-RPM loads.

Synchronous motors operate at an improved power factor, thereby improving the overall system power factor and eliminating or reducing utility power factor penalties. An improved power factor also reduces the system's voltage drop and the voltage drop at the motor terminals.

A PMSM abandons the excitation winding and the rotor turns at the same speed as the stator field. The PMSM's design eliminates the rotor copper losses, giving very high peak efficiency compared with a traditional induction motor. The power-to-weight ratio of a PMSM is also higher than induction machines.

Progress in the field of power electronics and microelectronics enables the application of PMSMs for high-performance drives, where, traditionally, only DC motors were applied.

Power Conversion Design Challenges

The main purpose of a power supply is to provide regulated, stable power to a load, regardless of power grid conditions. The switched-mode power supply (SMPS) is one type of power supply that has been widely used in office equipment, computers, communication systems and other applications because of its high efficiency and high energy density.

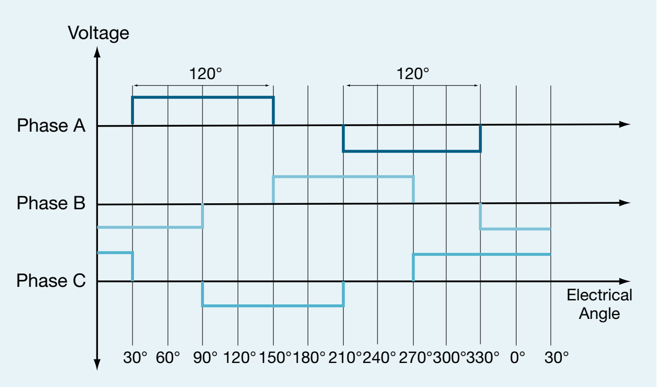
An SMPS fully digitally controlled by software running on a digital signal controller (DSC) has many advantages over mixed-analog and processor-controlled implementations. These include programmability, adaptability, reduced component count, design reusability, process independence, advanced calibration ability and better performance.

By using full digital control, an SMPS system becomes flexible and can also realize complex control arithmetic that improves efficiency and lowers cost. A controller-based SMPS system integrates high-performance digital signal processing with power electronics, providing a new method for design of power electronics and the typical high-level control and communication capability an SMPS system requires.

Brushless DC Motor Control Theory

The BLDC motor is a rotating electric machine with a classic three-phase stator like that of an induction motor; the rotor has surface-mounted permanent magnets. It is also referred to as an electronically commuted motor. There are no brushes on the rotor, and the commutation is performed electronically at certain rotor positions. The stator is usually made from magnetic steel sheets. The stator phase windings are inserted in the slots (distributed winding) or can be wound as one coil on the magnetic pole. The air gap magnetic field is produced by permanent magnets, the rotor magnetic field is constant. The magnetization of the permanent magnets and their displacement on the rotor is chosen so that the shape of the back-EMF (the voltage induced in the stator winding due to rotor movement) is trapezoidal. This allows a DC voltage, with a rectangular shape, to be used to create a rotational field with low torque ripples.

Figure 1: Three-Phase Voltage System for a BLDC Motor



The motor can have more then one pole-pair per phase. The number of pole-pairs per phase defines the ratio between the electrical revolution and the mechanical revolution. For example, the BLDC motor shown has three pole-pairs per phase, which represents three electrical revolutions per mechanical revolution.

The rectangular, easy to create, shape of the applied voltage makes controlling and driving the motor simple. However, the rotor position must be known at certain angles to be able to align the applied voltage with the back-EMF. Alignment of the back-EMF with commutation events is very important; when this is achieved, the motor behaves as a DC motor and runs at maximum efficiency. Thus, simplicity of control and performance makes the BLDC motor the best choice for low-cost and high-efficiency applications. Figure 1 shows the waveforms applied to a three-phase BLDC motor.

Permanent Magnet Synchronous Motor Control Theory

Thanks to sophisticated control methods such as vector control, a PMSM offers the same control capabilities as high-performance four-quadrant DC drives.

Vector control is an elegant control method of a PM-synchronous motor, where field-oriented theory is used to control space vectors of magnetic flux, current and voltage. It is possible to set up the coordinate system to decompose the vectors into a magnetic-field-generating part and a torque-generating part. Then the structure of the motor controller (Vector Control controller) is almost the same as for a separately excited DC motor, which simplifies the control of permanent magnet synchronous motor. This vector control technique was developed in the past specifically to achieve a similarly good dynamic performance in PM-synchronous motors.

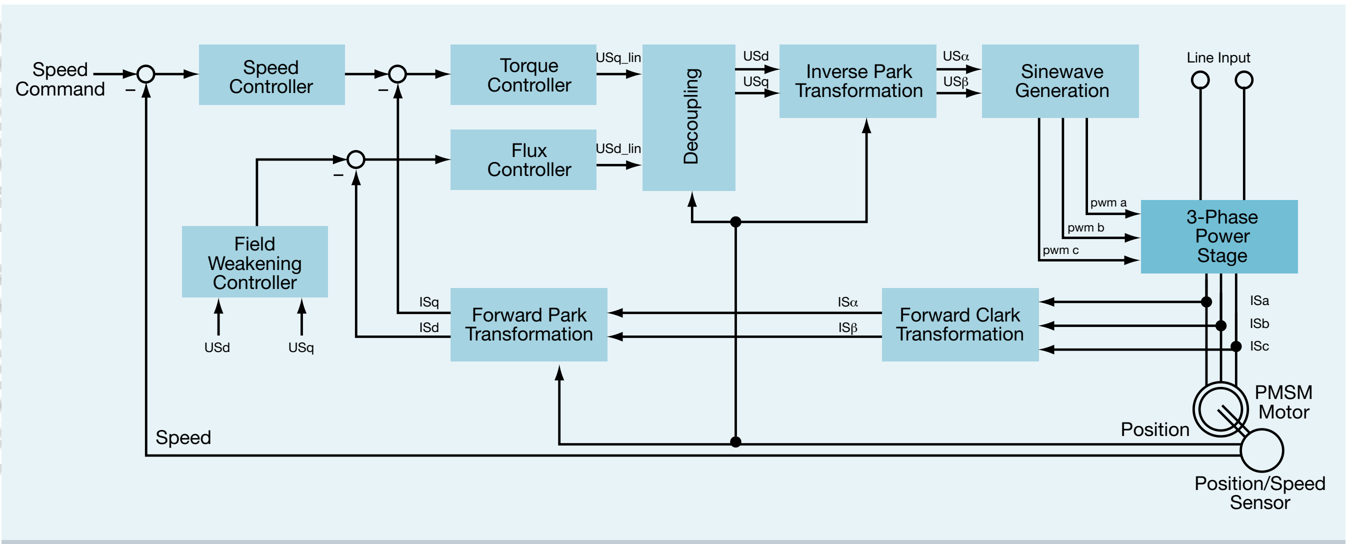
In this method, it is necessary to break down the field-generating part and the torque-generating part of the stator current to separately control the magnetic flux and the torque. To do so, you must set up the rotary coordinate system connected to the rotor magnetic field. This coordinate system is generally called “d,q-coordinate system.” Very high CPU performance is needed to perform the transformation from rotary to stationary coordinate systems.

Figure 2 shows the block diagram of the required task a CPU must perform in a vector control technique

To perform vector control:

- Measure the motor quantities (phase voltages and currents)
- Transform them into the two-phase system (α, β) using Clarke transformation
- Calculate the rotor flux space-vector magnitude and position angle
- Transform stator currents into the d,q-coordinate system using Park transformation
- The stator current torque (isq) and flux (isd) producing components are controlled separately by the controllers
- The output stator voltage space-vector is calculated using the decoupling block
- The stator voltage space-vector is transformed back from the d,q-coordinate system into the two-phase system fixed with the stator by inverse Park transformation
- Using the sinewave modulation, the output three phase voltage is generated

Figure 2: Permanent Magnet Synchronous Motor Vector Control Block Diagram



Benefits and Features of the 56F8013 DSC

The Freescale MC56F801x DSC family is well-suited to digital motor control, combining a DSP's calculation capability with an MCU's controller features on a single chip. These hybrid controllers offer many dedicated peripherals, such as pulse width modulation (PWM) module(s), an analog-to-digital converter (ADC), timers, communication peripherals (SCI, SPI, I^{2}C), on-board flash and RAM.

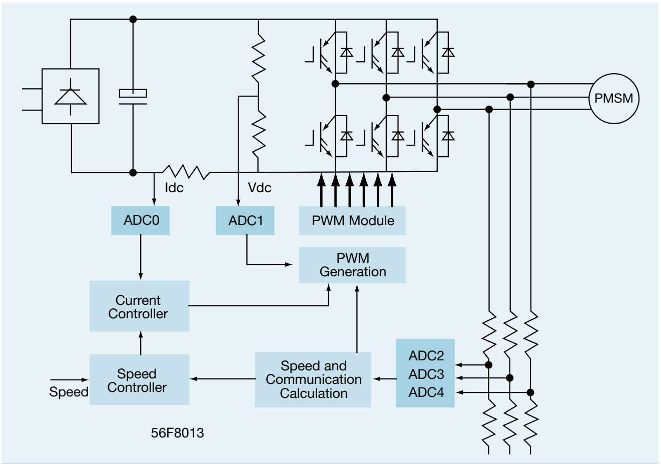
- Up to 32 MIPS at 32 MHz execution frequency
- On-chip memory includes high-speed volatile and non-volatile components
- 16 KB of program flash
- 4 KB of unified data/program RAM
- Flash security feature prevents unauthorized accesses to its content
- Flash protection prevents accidental modifications
- JTAG/Enhanced On-Chip Emulation (EOnCE) for unobtrusive, real-time debugging
- Four 36-bit accumulators
- 16- and 32-bit bidirectional barrel shifter
- Parallel instruction set with unique addressing modes
- Hardware DO and REP loops available
- Three internal address buses
- Four internal data buses
- MCU-style software stack support
- Controller-style addressing modes and instructions
- Single-cycle 16 x 16-bit parallel multiplier-accumulator (MAC)
- Proven to deliver more control functionality with a smaller memory footprint than competing architectures

Motion Control Proposed Solution

BLDC systems combine the positive attributes of AC and DC systems. In contrast to a brush DC motor, BLDC systems use a type of permanent magnet AC synchronous motors with a trapezoidal back-EMF waveform shape, and electronic commutation replaces the mechanical brushes in the DC motor. Although this control method will generate torque glitches during phase commutation, it satisfies most applications in which rotor speed is the control target.

PMSM motors with a sinusoidal back-EMF waveform shape can also be used in a BLDC system. But the phasor angle between stator flux and rotor flux is maintained between 60° electrical and 120° electrical. Torque ripples will occur during operation, but average torque will remain constant, meeting the requirements of most low-end applications. Figure 3 shows a block diagram that can be used to implement either PMSM vector control or a BLDC motor control.

Figure 3: Generic PMSM/BLDC Motor Control Solution Block Diagram



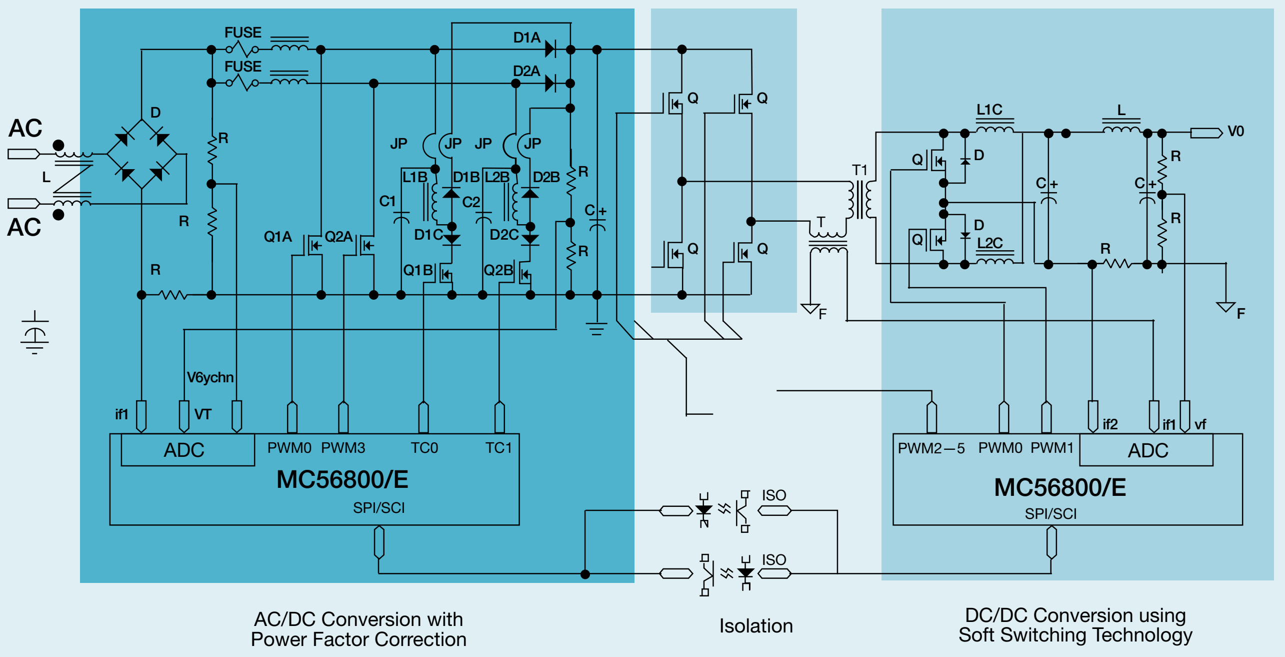
The information from back-EMF zero crossing can be used to determine the rotor position for proper commutation and to determine which power transistors to turn on to obtain maximum motor torque. The cheapest and the most reliable method to sample back-EMF zero crossing information is to feed the resistor network samples' back-EMF signal into ADC inputs or GPIOs. In sensored control structure, phases are commutated once every 60° electrical rotation of the rotor. This implies that only six commutation signals are sufficient to drive a BLDC motor. Furthermore, efficient control implies synchronization between the phase Bemf and the phase supply so that the Bemf crosses zero once during the non-fed 60° sector.

As only two currents flow in the stator windings at any one time, two phase currents are opposite and the third phase is equal to zero. Knowing that the sum of the three stator currents is equal to zero (star-wound stator), the anticipated instantaneous Bemf waveforms can be calculated. The sum of the three stator terminal voltages is equal to three times the neutral point voltage (Vn). Each of the Bemfs crosses zero twice per mechanical revolution, and as the Bemfs are numerically easy to compute, thanks to the signal processing capability of the 56F8013, it is possible to get the six required items of information regarding the commutation.

Switched-Mode Power Supply Proposed Solution

A generic SMPS AC/DC system is comprised of two parts: the primary side is the AC/DC converter with power factor correction (PFC); the secondary side is a full-bridge DC/DC converter. The AC/DC system uses an interleaved PFC boost control structure, which includes a full-bridge rectifier, two interleaved parallel BOOST PFC circuits, and two assistant

Figure 4: AC/DC Block Diagram



switches to realize the zero voltage switch (ZVS) of the main switches. Implementing a ZVS algorithm reduces the components' stress and improves efficiency, which allows the design to eliminate the reverse recovery output diodes. The DC/DC converter uses a ZVS phase-shifted full-bridge control structure implemented in software with a current doubler rectifier. This reduces the size of the filter inductor and improves efficiency.

A circuit diagram for a 56800/E-based switched-mode power supply (SMPS) is shown in Figure 4. The entire system is controlled by two 56800/E devices. The primary-side device accomplishes all control of the PFC system, which includes the two main switches and two ZVS switches. The secondary-side device accomplishes all control of the DC/DC phase-shifted full-bridge converter, which includes four main switches and two synchronous rectifiers. The functions performed in software for the PFC and DC/DC converter include: two digital PI regulators in the power system, control of all switches, soft start, digital generation of sine reference for the primary PFC, communication, power supply protection and supervisor functions.

References

www.freescale.com

Application Note AN1931
"Three-Phase PM Synchronous Motor Vector Control Using DSP56F80x" Freescale Semiconductor

Application Note AN3115
"Implementing a Digital AC/DC Switched-Mode Power Supply using a 56F8300 Digital Signal Controller" Freescale Semiconductor

Design Reference Manual DRM070
"Three-Phase BLDC Motor Sensorless Control Using MC56F8013" Freescale Semiconductor

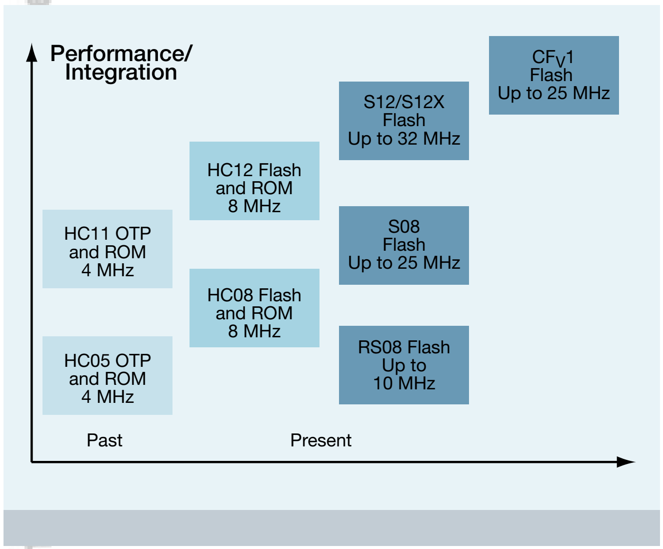
Design Reference Manual DRM077
"PMSM and BLDC Sensorless Motor Control Using the 56F8013 Device" Freescale Semiconductor

Daniel Torres is an applications engineer at Freescale Semiconductor, experienced in digital signal controllers, ColdFire controllers and 8-bit MCUs. He is focused on motor control and power management.

Inga Harris

The Newest S08/RS08 Tool
New breed of SPYDER discovered

Figre 1: Core Roadmap



Freescale Semiconductor is a global leader in the design and manufacture of embedded semiconductors for the automotive, consumer, industrial, networking and wireless markets. Over the last couple of years, we have put more and more of a focus on providing tools, both for the mass and hobbyist markets. With the recent addition of our e-commerce site, complimentary samples, free-of-charge compilers and debuggers and cost-effective hardware tools, Freescale's microcontroller families are now easily accessible to designers. The latest tool, the USBSPYDER08, manufactured by SofTec Microsystems™ with a 2007 price of $29 USD confirms our commitment to providing small, fast and cost-effective hardware tools to get you started with your design.

HC05 and HC11 8-bit cores, introduced in the 1980's, were widely used by major segments of the market. In the late 1990's the HC08 (8-bit) and HC12 (16-bit) cores were introduced and are now more popular with customers within niche consumer and industrial applications. In early 2000, the HCS12 16-bit core and the HCS08 (S08) 8-bit core were introduced with a key feature—Background Debug Module (BDM), which makes the development process easier. Figure 1 shows a section of Freescale's core roadmap.

The recently introduced RS08 (a reduced HCS08 core made physically smaller by removing instructions and other cost saving methods) and the 8-bit S08 microcontrollers contain a single-wire background debug interface supporting in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. It is this module that enables development of cost-effective, easy-to-use tools. The same BDM connection is also present on the 32-bit microcontroller ColdFire® V1 core products currently being introduced (see www.freescale.com/flexis for more information).

In 2005, freegeeks.net provided the HCS12 microcontroller community with an open source tool named TBDML (Turbo BDM Light). With 1,454 downloads in 12 months, we realized engineers found it very valuable. Now an equivalent tool for our 8-bit MCUs, BDM-enabled controllers are available in two forms:

- Open source BDM (OSBDM) for S08s—details of which can be found on Freescale Forums (www.freescale.net/forums). This self-build tool has a bill of materials (BOM) of less than $10 dollars and can be bought pre-fabricated from small companies.
- USBSPYDER08 Development Kit supporting MC9S08QG, MC9S08QD and MC9RS08KA 8-bit microcontrollers. This kit is designed to support future Freescale 8-bit, 8-pin MCUs.

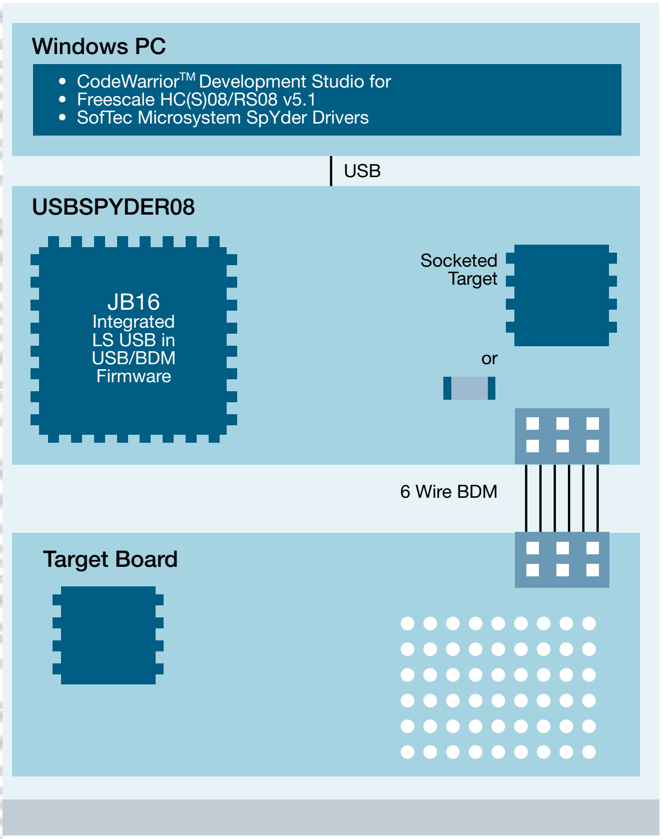
OSBDM and USBSPYDER08 essentially do the same thing. They interface between your development environment (Windows® PC based) and target MCU as shown in Figure 2. The main objective of these tools is to provide a less expensive and easy-to-design with option for the enthusiasts to use.

USBSPYDER08 is Freescale's new USB-to-BDM development tool. If you are not familiar with BDM, it is similar to debugWIRE or JTAG, used on our recently launched 8- and 16-bit MCUs.

The background (BKGD) pin on these devices provides a single-wire background debug interface to the on-chip debug modules. See the Development Tools chapter of any S08 or RS08 data sheet for more information about these debug modules and how to use them. While the interface is a single wire, typically a 6-pin connector, known as a BDM port, is used to interface with the target, as shown in Figure 3.

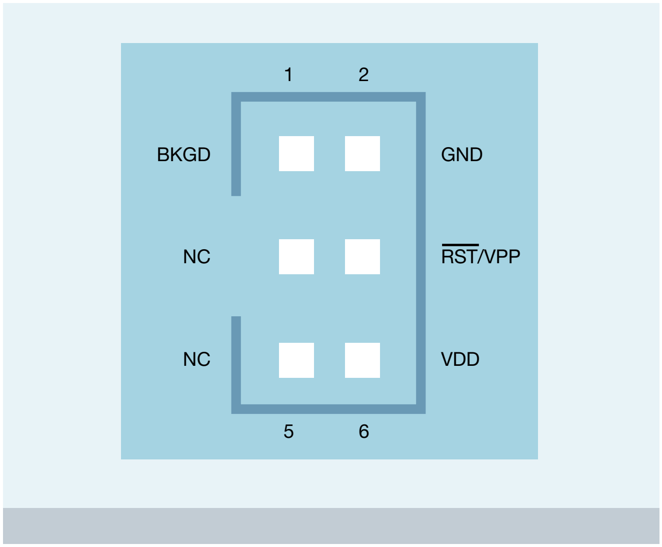
The primary function of this BKGD pin is for bidirectional serial communication of active background mode commands and data transfer. During reset, this pin is used to select between starting in active background mode and starting the user's

Figure 2: Development Environment



Windows PC
CodeWarrior™ Development Studio for
Freescale HC(S)08/RS08 v5.1
SofTec Microsystem SpYder Drivers
USB
USBSPYDER08
JB16 Integrated LS USB in USB/BDM Firmware
Socketed Target
or
6 Wire BDM
Target Board

Figure 3: BDM Connector



application program. Additionally, this pin requests a timed sync response pulse, allowing a host development tool to determine the correct clock frequency for background debug serial communications. Background Debug Controller (BDC) commands are sent serially from a host computer to the BKGD pin of the target S08 or RS08 MCU. The commands and data are sent MSB-first using a custom BDC communications protocol. With a single-wire background debug interface, it is possible to use a relatively simple interface pod to translate commands from a host computer into commands for the BDC.

In the case of the USBSPYDER08, a low-speed universal serial bus (USB) interface is used. Conveniently, the tool takes the form of a USB flash memory stick.

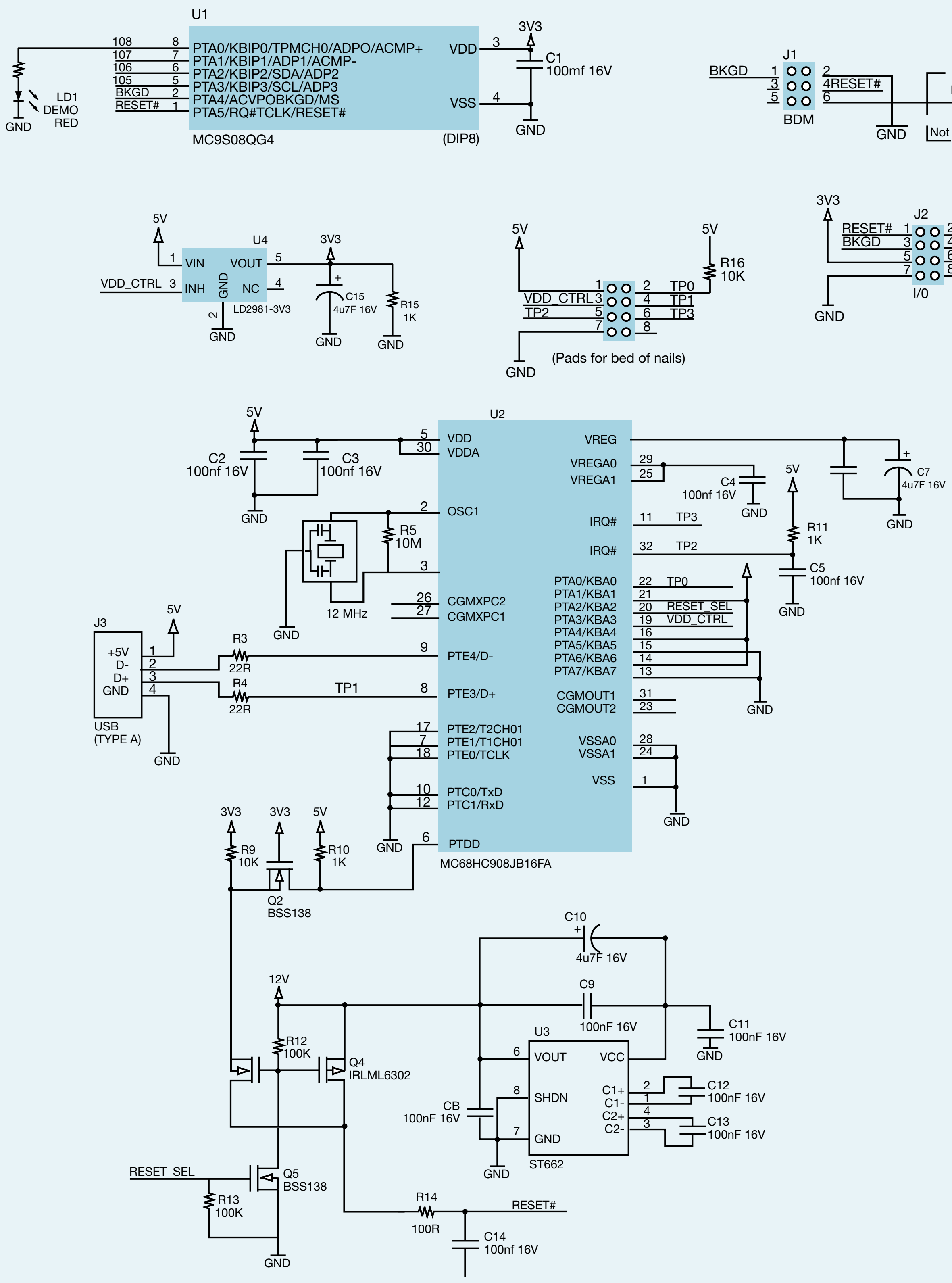
The USBSPYDER08 is a cost effective tool built by SofTec Microsystems with full collaboration from Freescale Semiconductor. Together with the award winning CodeWarrior® Development Tools, the USBSPYDER08 provides the essential tools you need to write, compile, download, in-circuit emulate and debug code. Full-speed program execution allows developers to perform hardware and software testing in real time. The tool works up to bus speeds of 10 MHz, supporting the 3.3V operation range of the MCUs. It has an on-board socketed target MCU that can be replaced with other supported PDIP packaged parts available in small sample quantities from www.freescale.com. To increase the flexibility of the tool, it has a BDM connector for off-board debugging of the supported products in other packages or if the development requires the incorporation of other board components.

How USBSPYDER08 Works

By relying on the S08 and RS08 background debug controller (BDC) module, the USBSPYDER08 enables a fast and easy way to program the on-chip flash and any other memories. It is the primary debug interface for development which allows non-intrusive access to memory data and traditional debug features such as CPU register modify breakpoint and single-instruction trace commands.

This tool uses the USB interface to communicate to the PC, and the USB bus voltage to power the tool and microcontroller without the bulky wall adapters of old, making it truly portable. The USB power supply can also power the target board, provided not too many motors are running off it (up to 100mA). The MC9S08JB16 (JB16) microcontroller is at the heart of the USBSPYDER08 tool. The MCU has a USB (2.0 low speed) interface and operates from the 5V supplied by USB. When the tool supports RS08 MCUs, 12V is needed for programming. The JB16 recognizes the target and via control of PTD0 can enable the 12V signal using the DC-DC converter chip.

Figure 4: SpYder Schematic



Step 1—USBSPYDER08 Board Assembly

The essential parts of the USBSPYDER08 tool are already populated on the product, allowing you to use it as a stand-alone tool with the socketed microcontroller. If there is a requirement to use another package type or a separate target board, it is also supported. By adding a 0 ohm resistor or a short circuit on the space next to the bed of nails labeled R2, the power supply is connected to the BDM socket so you can use another target board. The net result is the USBSPYDER08 Discover Kit turns into a BDM pod.

Step 2—Debugger Installation

CodeWarrior® Development Studio for Freescale HC(S)08/RS08 v5.1 Special Edition is available free of charge on the USBSPYDER08 CD. More information on the features of this tool is available at www.freescale.com/codewarrior. Without a license key, the product will run in a 1 KB code-size in limited demonstration mode. To break the 1 KB limit, there are two options:

- Contact Freescale to request an unlimited period, free license key to increase the code size limit to 16 KB
- Contact Freescale to request a 30-day limited, free license key to run the compiler without code size limitations.

Step 3—Drivers

Once the board is ready for development, the CodeWarrior tool is installed, and the next step is to make the USBSPYDER08 communicate with the debugger. After the drivers have been installed from the accompanying CD and the tool is connected to the PC for the first time Windows will recognize a new USB device; the “Windows New Hardware Wizard” dialog box will be opened and asked for the required driver. To complete the installation the developer must simply choose the “Install Automatically” option.

Step 4—Ready to Discover

It is important to appreciate that USBSPYDER08 uses the target MCU to execute the in-circuit debugging, not an emulator, so the microcontroller’s peripherals (e.g. timers, A/D converters, Serial Communication Modules) are not reconstructed by software or an external device.

To create a new project with the CodeWarrior tool for any of the devices supported by USBSPYDER08 the first place to start is with the CodeWarrior tool. From the main menu, select “File > New Project…” or from the startup box click “Create New Project”. A dialog box will then appear prompting to select the target device from a pull down menu. Select “SofTec HCS08” as the connection type. Then set the code type, project name and location. If there are no files to add and you do not want to use the device initialization tools, then press “Finish”. CodeWarrior will create the project, set up the framework for the code and include the header file with all the register definitions. Once you have written code and are ready to start debugging, make the file and then compile. There are a few ways to do this by either using the shortcut buttons on the window, using the drop down menu “Project” or by pressing F5 then F7.

The first time the debugging session is entered an “MCU Configuration” dialog box will open, prompting to select the debugging hardware connection to your PC. Make sure that the connection type “USBSPYDER08” is selected. At this point the CodeWarrior tool has erased and reprogrammed the memory and trimmed the oscillator if the feature is available on the selected MCU.

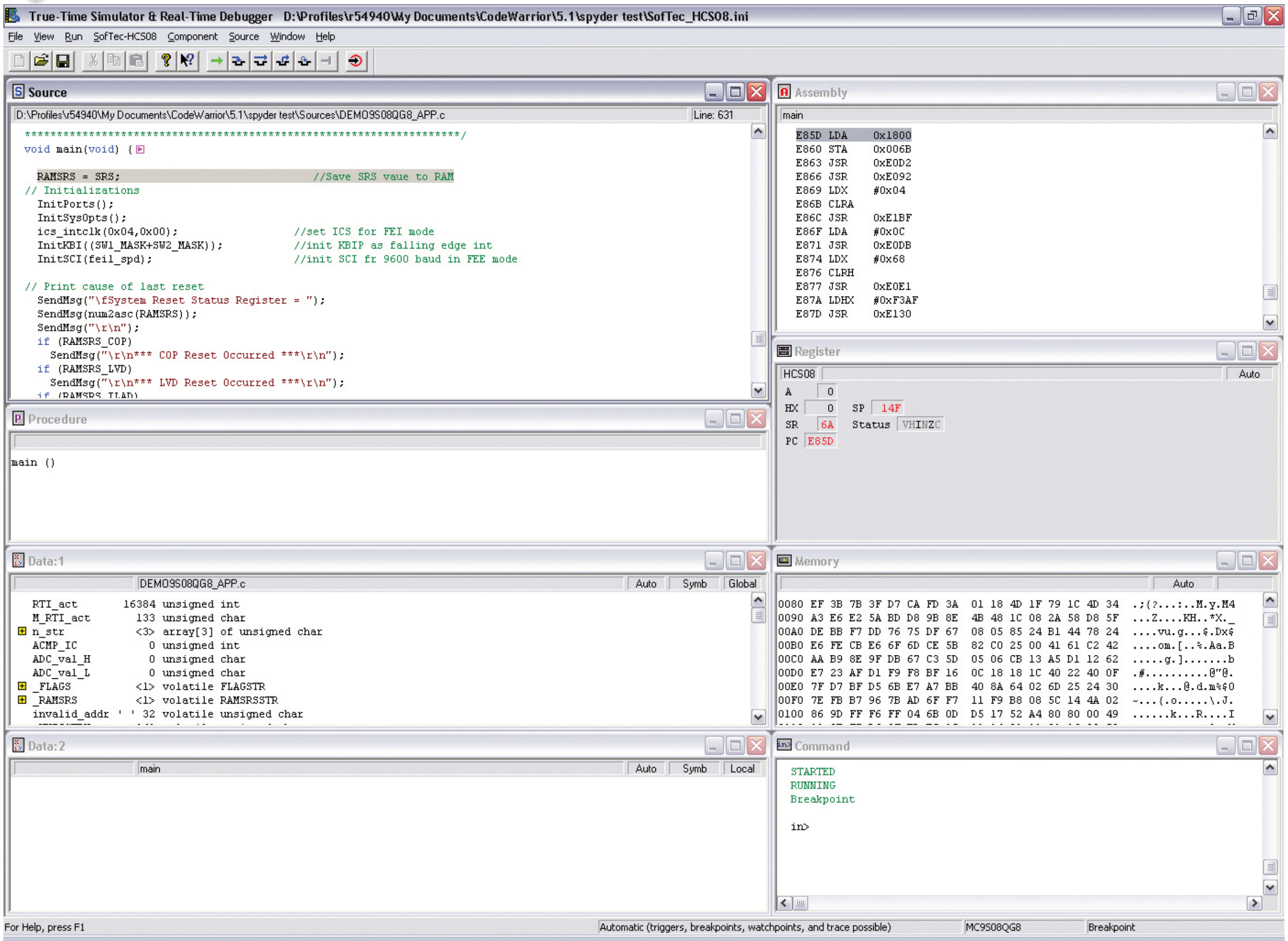
At this stage you have the tools you need to start debugging code. The CodeWarrior tool suite enables a variety of ways to analyze program flow via breakpoints, watchpoints and a trace buffer. All these features are implemented by taking advantage of the target microcontroller’s debug peripheral. Figure 5 shows the CodeWarrior window which will consist of a:

- Source window with your code displayed
- Assembly window where you can see what the compiler has created from your source code
- Registers window where the CPU registers are visible
- Memory window where you can watch any location or force bytes to another value
- Other windows with data, procedures and commands

The format of the data and the refresh rates of the data can be changed by right clicking on the window and changing the mode or format. Preferred settings can be saved by going to “File > Save Configuration.” Users new to the CodeWarrior tool suite should take a few minutes to familiarize themselves with the Start/Continue, Single Step, Step Over, Step Out, Assembly Step, Halt and Reset Target buttons. Application notes AN3335—Introduction to HCS08 Background Debug Mode and AN2616—Getting Started with HCS08 and CodeWarrior Using C are good resources when learning about the debugging environment.

The bed of nails next to the MCU can be connected to a scope to allow monitoring of the pins outside the debugger, in real time.

Figure 5: CodeWarrior Screenshot



True-Time Simulator & Real-Time Debugger D:\Profiles\r54940\My Documents\CodeWarrior\5.1\spyder test\SofTec_HCS08.ini
File View Run SofTec-HCS08 Component Source Window Help
Source
D:\Profiles\r54940\My Documents\CodeWarrior\5.1\spyder test\Sources\DEMO9S08QG8_APP.c
Line: 631

```
*********************************************************************/
void main(void) {

  RAMSRS = SRS;                               //Save SRS vaue to RAM
// Initializations
  InitPorts();
  InitSysOpts();
  ics_intclk(0x04,0x00);                   //set ICS for FEI mode
  InitKBI((SW1_MASK+SW2_MASK));            //init KBIP as falling edge int
  InitSCI(feil_spd);                       //init SCI fr 9600 baud in FEE mode

// Print cause of last reset
  SendMsg("\fSystem Reset Status Register = ");
  SendMsg(num2asc(RAMSRS));
  SendMsg("\r\n");
  if (RAMSRS_COP)
    SendMsg("\r\n*** COP Reset Occurred ***\r\n");
  if (RAMSRS_LVD)
    SendMsg("\r\n*** LVD Reset Occurred ***\r\n");
  if (RAMSRS_ILAD)
```

```
E85D LDA    0x1800
E860 STA    0x006B
E863 JSR    0xE0D2
E866 JSR    0xE092
E869 LDX    #0x04
E86B CLRA
E86C JSR    0xE1BF
E86F LDA    #0x0C
E871 JSR    0xE0DB
E874 LDX    #0x68
E876 CLRH
E877 JSR    0xE0E1
E87A LDHX   #0xF3AF
E87D JSR    0xE130
```

```
A      0
HX     0   SP  14F
SR    6A   Status VHINZC
PC  E85D
```

```
main ()
```

```
RTI_act       16384 unsigned int
M_RTI_act       133 unsigned char
⊞ n_str         <3> array[3] of unsigned char
ACMP_IC           0 unsigned int
ADC_val_H         0 unsigned char
ADC_val_L         0 unsigned char
⊞ _FLAGS        <1> volatile FLAGSTR
⊞ _RAMSRS       <1> volatile RAMSRSSTR
invalid_addr ' ' 32 volatile unsigned char
```

```
0080 EF 3B 7B 3F D7 CA FD 3A  01 18 4D 1F 79 1C 4D 34  .;{?...:..M.y.M4
0090 A3 E6 E2 5A BD D8 9B 8E  4B 48 1C 08 2A 58 D8 5F  ...Z....KH..*X._
00A0 DE BB F7 DD 76 75 DF 67  08 05 85 24 B1 44 78 24  ....vu.g...$.Dx$
00B0 E6 FE CB E6 6F 6D CE 5B  82 C0 25 00 41 61 C2 42  ....om.[..%.Aa.B
00C0 AA B9 8E 9F DB 67 C3 5D  05 06 CB 13 A5 D1 12 62  .....g.]......b
00D0 E7 23 AF D1 F9 F8 BF 16  0C 18 18 1C 40 22 40 0F  .#..........@"@.
00E0 7F D7 BF D5 6B E7 A7 BB  40 8A 64 02 6D 25 24 30  ....k...@.d.m%$0
00F0 7E FB B7 96 7B AD 6F F7  11 F9 B8 08 5C 14 4A 02  ~...{.o.....\.J.
0100 86 9D FF F6 FF 04 6B 0D  D5 17 52 A4 80 80 00 49  ......k...R....I
```

```
STARTED
RUNNING
Breakpoint

in>
```

```
For Help, press F1    Automatic (triggers, breakpoints, watchpoints, and trace possible)    MC9S08QG8    Breakpoint
```

Summary

For its size and simplicity, the USBSPYDER08 Discovery Kit is a surprisingly flexible tool that meets low budget developments requirements. It supports Freescale's 8-pin S08 devices with the ability to extend pin count using the off-chip target option. When combined with the power of CodeWarrior tool suite offerings—IDE, compiler, debugger, editor, linker, assembler, run control devices—Freescale provides the key tools necessary to build platforms and applications for the mass market and hobbyists.

Happy Developing!

Further Reading Materials

AN3335
Introduction to HCS08 Background Debug Mode

AN2616
Getting Started with HCS08 and CodeWarrior Using C

BR8BITLOWEND
A Little 8-bit Goes a Long Way

HCS08QRUG
HCS08 Peripheral Module Quick Reference

RS08 Peripheral Module Quick Reference
A Compilation of Demonstration Software for RS08 Modules

Inga Harris graduated from the University of Strathclyde in 2000 with an honors degree in electronics and electrical engineering. Her Technical Marketer and Applications Engineering career at Freescale has predominantly been with the 8-bit MCU families with a strong focus on Consumer and Industrial markets.

Pattye Brown

ZigBee® and Low-Cost Wireless Networks
For sensing and control applications

Figure 1: Example, stand alone transceiver for use with various microcontrollers (Freescale Semiconductor MC1320X)

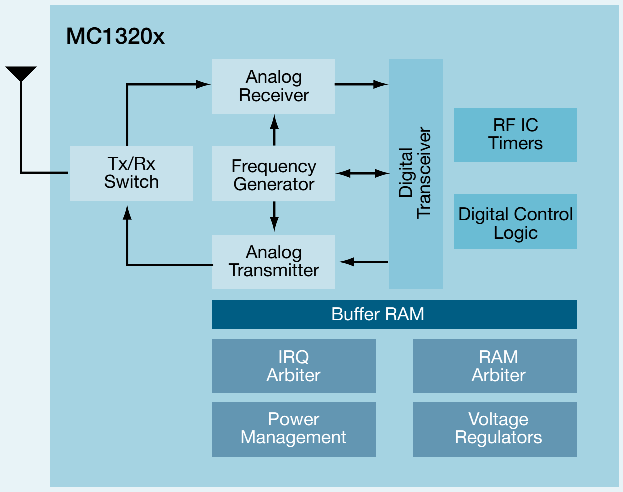
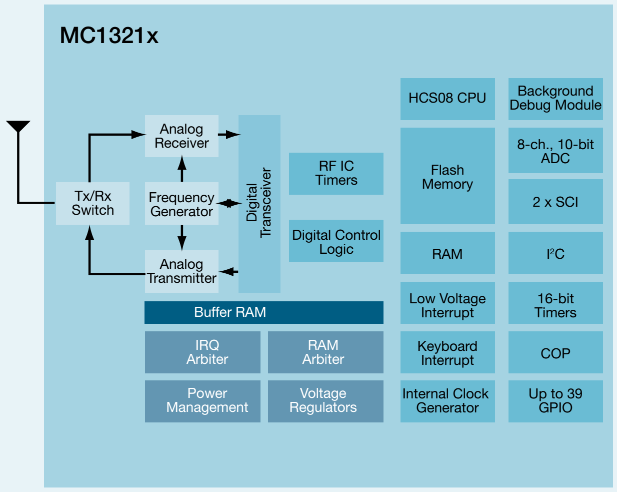


Figure 2: Example, integrated system in package (SiP), transceiver and microcontroller (Freescale Semiconductor MC1321X)



Many sensing, monitoring and control applications offer a unique potential to incorporate low-cost wireless networking functionality. Low-cost wireless networking solutions often require ranges of 30-70 meters or less, data rates of 250 kbps or less and, in many cases, the capability to achieve optimum battery life, particularly for end node functions. The wireless networking implementation can be enhanced with proactive analysis of several key factors prior to design start. A matrix of these key factors will help you choose your components and solutions. Reference design schematics are also provided as a baseline for design initiation. Consider reviewing the following areas when determining your design requirements:

- Integration
- Wireless networking topologies (Figure 3)
  - Radio (RF modem or transceiver)
  - Performance
  - Operating voltage
  - Data rates
  - Range
  - Channel flexibility
  - Output power
  - Sensitivity
  - Power management
  - Peripherals
  - Clocking
  - Multi-tier software
  - Ease of hardware and software design
  - Antenna design
  - Packaging
- Microcontroller (MCU)
  - CPU features
  - Performance
  - Memory options
  - Power management
  - Clock source options
  - Analog to digital conversion
  - Peripherals
  - Packaging
  - In-circuit debug and programming
  - Ease of software and hardware design

Such analysis will provide an organized perspective for engineering decisions, an avenue toward design success, a fast time to market, and an easier implementation of the low-cost wireless networking.

A variety of implementation alternatives for low-cost wireless networking can give you a high level of flexibility in the design process. As one alternative, consider solutions from providers that offer various configurations of stand-alone transceivers to be used in conjunction with a wide selection of MCUs (Figure 1). As a second and equally effective alternative, consider the newest solutions which offer integrated transceiver/MCU products (Figure 2). Reuse of design components and engineering investment may be important as you work on multiple, yet similar, end products. Therefore, a structured evaluation of solution options can be both cost and resource efficient. Well thought out research may provide a basis for several end products to be designed from a single foundation.

Wireless Networking Technologies

The 2.4 GHz industrial, scientific and medical (ISM) band supports multiple short range wireless networking technologies. Each alternative has been developed to optimally serve specific applications or functions. The networking topologies most commonly associated with the 2.4 GHz frequency range are Bluetooth™, WiFi™ and ZigBee® as well as other proprietary solutions. Non-standards-based proprietary solutions offer some risk as they are vendor dependent and thus subject to change.

ZigBee, an IEEE® 802.15.4 standards-based solution, as defined by the ZigBee Alliance, was developed specifically to support sensing, monitoring and control applications. The ZigBee solution offers significant benefits, such as low power, robust communication and a self-healing mesh network. The ZigBee solution frequencies are typically in the 868/915 MHz or 2.4 GHz spectrums.

The ZigBee data rate for technology solutions is 250 Kbps. Power consumption must be extremely low to allow battery life that is measure in years (equivalent to the shelf life of the battery) using alkaline or lithium cells. ZigBee technology theoretically supports up to 65,000 nodes. Common applications in sensing, monitoring and control, which are best supported by a ZigBee technology solution include:

- Personal and medical monitoring
- Security, access control and safety monitoring
- Process sensing and control
- Heating, ventilation and air conditioning (HVAC) sensing and control
- Home, building and industrial automation
- Asset management, status and tracking
- Fitness monitoring
- Energy management

Figure 3: Wireless Networking Technologies

	ZigBee®	Bluetooth®	UWB™	Wi-Fi™	LonWorks®	Proprietary
Standard	IEEE® 802.15.4	IEEE 802.15.1	IEEE 802.15.3a (to be ratified)	IEEE 802.11 a, b, g (n, to be ratified)	EIA 709.1, 2, 3	Proprietary
Industry Organizations	ZigBee Alliance	Bluetooth SIG	UWB Forum and WiMedia™ Alliance	WiFe Alliance	LonMark Interoperability Association	N/A
Topology Mesh, Star, Tree	Star	Star	Star	Medium-dependent	P2P, Star, Mesh	
RF Frequency	868/915 MHz 2.5 GHz	2.4 GHz	3.1–10.6 GHz (U.S.)	2.4 GHz 5.8 GHz	N/A (wired technology)	433/868/900 MHz 2.4 GHz
Data Rate	250 Kbps	723 Kbps	110 Mbps–1.6 Gbps	10–105 Mbps	15 Kbps–10 Mbps	10–250 Kbps
Range	10–300m	10m	4–20m	10–100m	Medium-dependent	10–70m
Power	Very low	Low	Low	High	Wired	Very Low–Low
Battery Operation (Life)	Alkaline (Months–Years)	Rechargeable (Hours–Days)	Rechargeable (Hours)	N/A	Alkaline (Months–Years)	
Nodes	65,000	8	128	32	32,000	100–1,000

RF Modem or Transceiver (Radio)

Several radio frequency (RF) modem features should be considered for implementing low-cost wireless networking systems. Most low-cost personal area network (PAN) RF modem solutions recommend power supplies from 2.0–3.6V.

For lightweight wireless networks, low data rates are adequate to support monitoring, sensing and control functions and also help manage system power consumption. 250 kbps offset quadrature phase-shift keying (O-QPSK) data in 2 MHz channels with 5 MHz spacing between channels with full spread-spectrum encode and decode is most often selected for these application types. In these environments, the transceiver wakes up, listens for an open channel and transmits small packets of data at lower data rates. Then it shuts down until the next event is indicated. The sequencing, fast power on latency, lower data rates and small data packets allow an 802.15.4 transceiver to select time increments where the data transmission will be most effective.

As mentioned previously, for sensing and control subsystems, data transmission range and power requirements are best supported with ZigBee technology solutions. The typical range defined by the ZigBee Alliance specification is 10–70m, however, many solutions offer line-of-sight ranges well beyond this.

It is important to review the number and types of transceiver channels available in relation to the planned design. Selectable transceiver channels offer the designer the option to take advantage of channels which minimize noise, particularly staying away from the more crowded 2.4GHz WiFi channels.

You should look for typical transmit output power in the 0 dBm up to +4 dBm range. Receive sensitivity typically in the -90 dBm range will offer adequate capabilities for sensing, monitoring and control functions. Buffered transmit and receive data packets simplify management of low-cost microcontrollers that will be used with the transceiver. The radio or transceiver should also offer link quality and energy detect functions for network performance evaluation.

Multiple power-down modes offer power saving features to minimize system power consumption. These typically include off current, hibernate current and doze currents in the single digit microamp (µA) ranges. Programmable output power also allows the designer to reduce power consumption where range or environment require less power to achieve transmit and receive objectives. Ensuring these functions are offered in the selected solution will aid in maximizing battery life in battery operated full-function/coordinator or end node devices, often to the full shelf life of the battery.

Look for additional essential peripherals, such as internal timer comparators, which are available to reduce MCU resource requirements. General purpose input/output ports (GPIO) are available in various different configurations and counts. GPIO is heavily dependent on interface requirements with other devices within the application. In solutions which offer the flexibility of a transceiver with separate MCU, the communications is handled through the serial peripheral interface (SPI) port. As would be expected, when the radio and MCU are integrated into a single package or chip, the transceiver communicates to the MCU through the onboard or internal SPI command channel. Also, integrated solutions which include low noise amplifiers (LNA), power amplifiers (PA) with internal voltage controlled oscillator (VCO), integrated transmit/receive switch, on-board power supply regulation and full spread-spectrum encoding and decoding reduce the need for external components in the system and lower overall system cost.

A wide array of system clock configurations gives you flexibility in end system design. Options which allow either an external clock source or crystal oscillator for CPU timing are most suitable. A 16 MHz external crystal is typically required for the modem clocking. The ability to trim the modem crystal oscillator frequency helps to maintain the tight standards required by the IEEE® 802.15.4 specification.

Depending on the complexity and requirements of the end design, you are best served by vendors who offer multiple network software topology alternatives. These may include a simple media access controller (MAC) configuration which utilizes MCU flash memory sizes from 4 KB and up and supports point-to-point or simple star networks. Fully 802.15.4 compliant MAC and full ZigBee compatible topologies, while requiring more memory, provide the added support of mesh and cluster tree networks.

Ease the design process by using vendor provided reference designs, hardware development tools and software development tools. For hardware development tools, simple getting started guides, essential boards with incorporated LED and LCD for a visual monitor plus cables and batteries provide an easy out-of-the-box experience. These tools help you set up a network within minutes and actually evaluate network and solution performance. In the past some software design tools, specifically those which support fully ZigBee compliant networks, have been extremely difficult to use. To reduce the complexity of RF modem preparation, look for vendors that offer graphical users interface (GUI)-based software design tools that walk the designer through a step-by-step transceiver set-up.

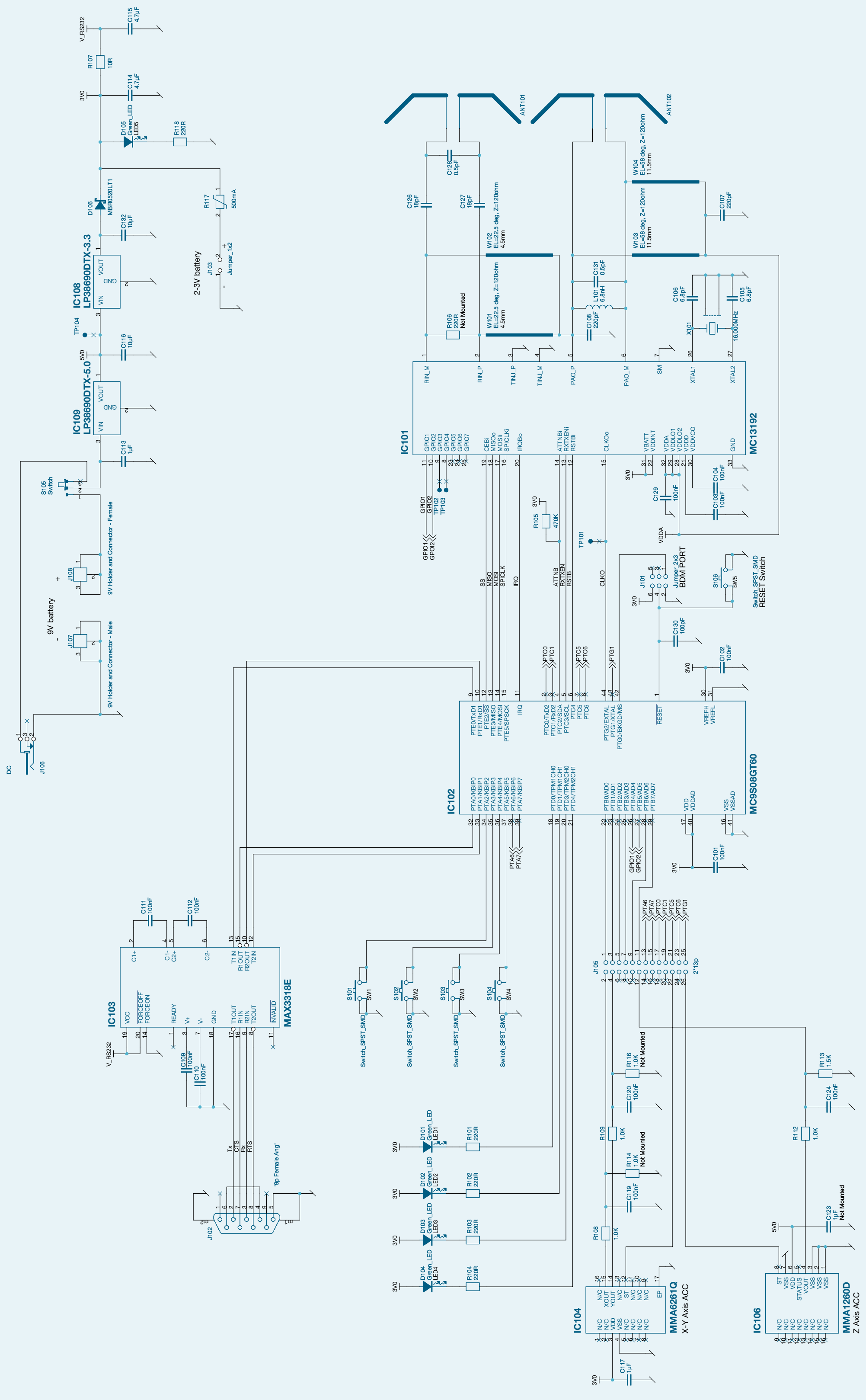


Figure 4: 13192 SARD Reference Block Diagram

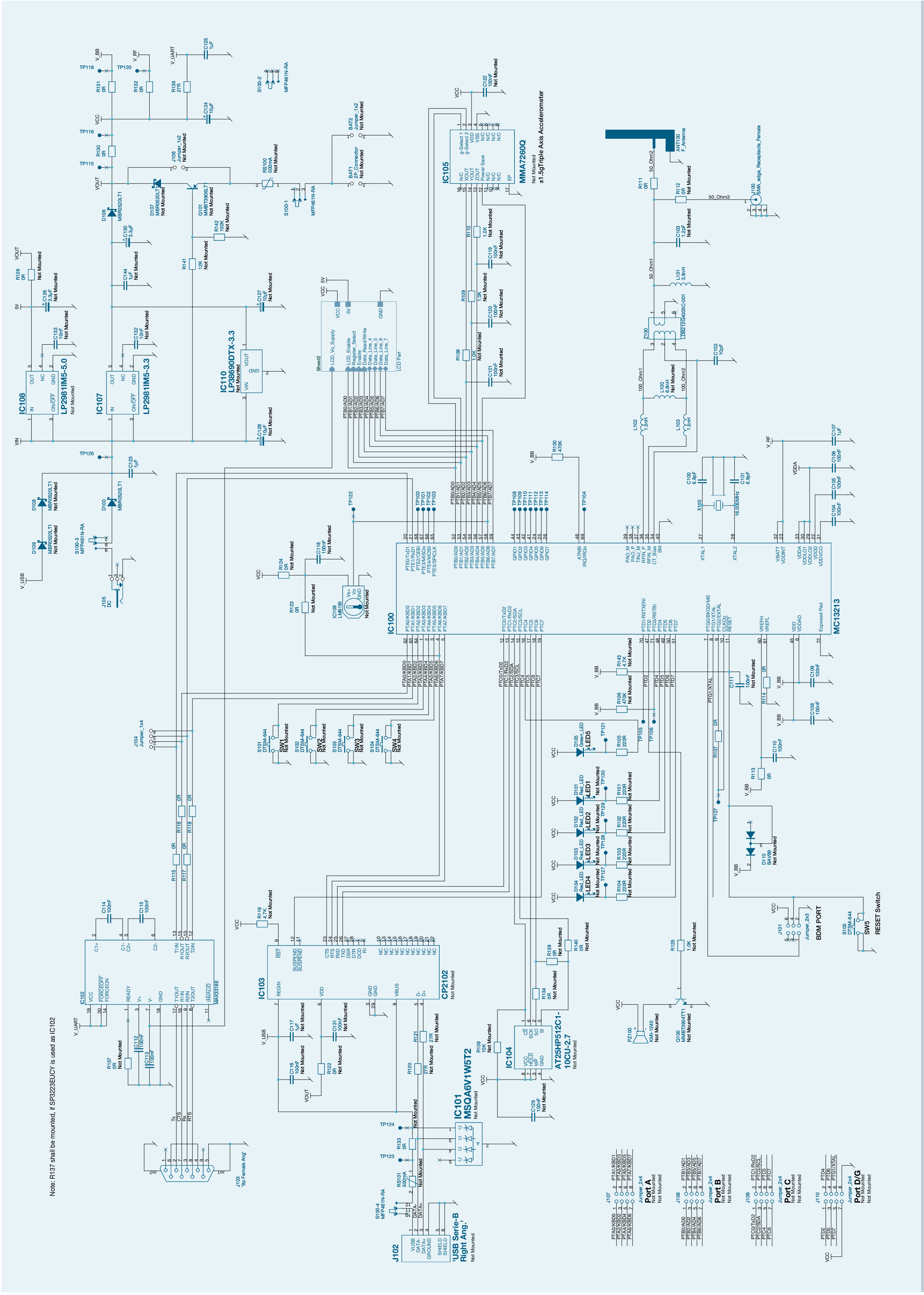


Figure 5: 13213 SED Reference Design Schematic

Antenna design can be a complex issue, particularly for digital designers who have limited to no experience in RF design. Typically, designers will take into account such factors as selecting the correct antenna, antenna tuning, matching, gain/loss and knowing the required radiation pattern. It is advisable to gain a basic knowledge of antenna factors through application notes provided by the transceiver vendor. However, most digital engineers prefer to consider working with a vendor solution where antenna design is provided. This allows them to focus on the application design. Look for antenna solutions where the antenna design is offered in completed Gerber files, which can be provided directly to the printed circuit board manufacturer for implementation. A vendor who provides such antenna design solutions eliminates the issues associated with good antenna design, good range and stable throughputs in wireless applications.

The quad flat no-lead package (QFN) is the optimum small footprint packaging solution for the transceiver portion of a low cost wireless networking subsystem. The packaging takes into consideration the board space limitations often driven by sensing and control solutions. Size is particularly important in the case of end nodes that are often battery operated with limited implementation space.

Microcontroller

Multiple alternatives exist in selecting a sensing and control implementation scheme. Some designers select a system in package (SiP) or platform in package™ (PiP) which includes transceiver and MCU functionality in a single package. However, should you opt for a stand-alone transceiver and microcontroller configuration, you gain the flexibility to choose from a variety of MCUs to mix and match for multiple end product configurations.

When choosing the latter implementation scheme, appropriate MCU selection requires thorough research. MCU selection depends upon matching the complexity of the sensing and control application with suitable performance factors, memory configurations and peripheral modules. Often for low cost wireless sensing systems, 8-bit MCUs in the 20 MHz CPU operating frequency (10 MHz bus clock) range offer an easy-to-implement, low-cost alternative which best suits these applications. Background debugging and breakpoint capability to support single breakpoint (tag and force options) setting during in-circuit debug (plus two breakpoints in on-chip debug module) offer the preferred debugging environment. Many MCU solutions provide support for up to 32 interrupt/reset sources.

Memory requirements for sensing and control applications are typically 8 KB of flash with 512B of RAM or as low as 4 KB of flash with 256B of RAM. Flash read, program or erase over the full operating voltage and temperature are essential.

A variety of operation modes provides precise control over power consumption, a key feature for extending battery operated solutions. Look for MCUs that support normal operating (run) mode, active background mode for on-chip debug, a variety of stop modes (bus and CPU clocks are halted) and wait mode alternatives.

Consider a microcontroller with an internal clock source module containing a frequency-locked-loop (FLL) controlled by an internal or external reference with precision trimming of internal reference that allows 0.2 percent resolution and 2 percent deviation over temperature and voltage. The internal clock source module should support bus frequencies from 1 MHz to 10 MHz. MCUs with selectable clock inputs for key modules provide control over the clock to drive the module function. As well, look for MCUs with a low-power oscillator module with software selectable crystal or ceramic resonator in the range of 31.25 KHz to 38.4 KHz or 1 MHz to 16 MHz that supports external clock source input up to 20 MHz.

It is essential that the chosen MCU offer system protection, including such options as watchdog computer operating properly (COP) reset with an alternative to run from a dedicated 1 KHz internal clock source or bus. Other must-have system protection features include low-voltage detection with reset or interrupt, illegal opcode detection with reset, illegal address detection with reset and flash block protection.

A variety of embedded peripherals will ease the implementation of your application. An 8-channel, 10-bit analog-to-digital (ADC) converter is recommended for accurate successive approximation. Specific functions should include automatic compare, asynchronous clock source, temperature sensor, internal bandgap reference channel and an ADC that is hardware triggerable using the real-time interrupt (RTI) counter.

Other essential peripherals for sensing and control applications include: an analog comparator module (ACMP) with an option to compare internal reference; serial communications interface module (SCI); serial peripheral interface module (SPI); inter-integrated circuit (I^2C) bus module; 2-channel timer/pulse-width modulator for input capture; output compare; buffered edge-aligned PWM or buffered center-aligned PWM; 8-bit modulo timer module with prescaler and 8-pin keyboard interrupt module with software selectable polarity on edge or edge/level modes.

There are multiple small foot print MCU packaging options that satisfy sensing and control design requirements. These help optimize limited board space, particularly in end node, battery operated functions. A few of the MCU packages that meet these considerations are low pin-count plastic dual in-line (PDIP), quad flat no-lead (QFN), thin shrink small outline (TSSOP), dual flat no-lead (DFN) and narrow body, small outline (NB SOIC) packages.

It is also prudent to consider as part of the MCU selection hardware and software design tool ease-of-use, documentation clarity, reference design and application code availability and other design support offerings. Similarly, on the RF or modem side of the design, an effective integrated development environment (IDE) should include GUI-driven tools with built-in features and utilities that simplify coding and project file management to expedite the design process. Expert tools that abstract the hardware layer and generate optimized, MCU-specific C code tailored to the application allow you to concentrate on application concepts. Fast and easy debug as well as a flash programming capability need to be considered. It is also helpful to have access to features that allow the designer to create reusable software components for reuse between projects.

Schematics for Sensing, Monitoring and Control Subsystems

A reference design schematic for sensing, monitoring and control subsystems is often of value as an application baseline from which to evolve design specific requirements. As an example of a design using a system in package (SiP) solution, shown below is the schematic for the Freescale 1321x-SRB sensor reference board, which is included in the 1321XDSK (developer's starter kit), 1321xNSK (network starters kit) and 1321xEVK (ZigBee evaluation kit). The 1321x-SRB includes the Freescale MMA7260Q tri-axis acceleration sensor and, used with the starter kits, helps you set up working networks within a matter of minutes. The board offers a reference design application that is a starting point for sensing application development (Figure 5).

You also have the option to use a stand alone transceiver and MCU alternative to provide additional design flexibility. For example, the block diagram below is Freescale's 13192 sensor application reference design for sensing, monitoring and control subsystems. Completed boards based on this design are included in our 13192DSK-A0E (developer's starter kit) and 13192EVK-SFTE (evaluation kit) development kits (Figure 4).

Supporting files for the above reference designs are available via web download. The download folder includes schematics, bill of materials, gerber files and other necessary documentation for complete reference design implementation.

Freescale's Wireless Networking Solutions for Sensing, Monitoring and Control Applications

We offers an exceptional set of solutions for the digital engineer who wants to establish a new paradigm in end products. The broad portfolio meets the design requirements noted above that are essential to low cost, wireless networking implementations. Sample products include:

- RF modem or transceiver solutions (Simple MAC, 802.15.4 MAC and ZigBee), which can be used with a variety of MCUs (MC1319x and MC1320x transceiver families are used with the HCS08 and ColdFire MCU families)
- Integrated transceiver solutions (Simple MAC, 802.15.4 MAC and ZigBee) with MCU system in package (SiP) solutions (MC1321x SiP family)
- Analog components
- Sensor components

For easy-to-use implementations of ZigBee and other low-cost, low-power wireless networks, Freescale is your resource for reference designs, application notes, hardware development tools and software design tools. For more detailed product information visit us at www.freescale.com/zigbee.

Pattye Brown is a technical marketing engineer and channel marketing liaison at Freescale Semiconductor; she has been with the company for 23 years. She holds a bachelor's degree in industrial engineering and a masters degree in business administration and marketing.

Dr. Martin Mienkina

Microcontrollers
For cost-sensitive applications

When you take a close look around us, humans are surrounded by a lot of electronic equipment every step of the day. Washing machines, refrigerators, dishwashers, vacuum cleaners, power tools and heating control in homes, automobile seat positioning, power steering and braking, air conditioners, ceiling fans, computers and many more devices in the workplace have electronic components.

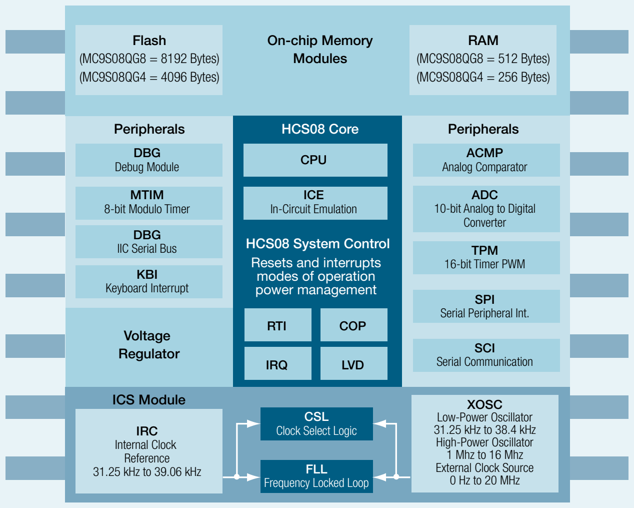
Each device contains an electrical drive and auxiliary electronics consuming energy at various rates of efficiency. Recent global interest towards environmental friendliness and energy savings puts pressure on how engineers design to keep these efficiency rates as high as possible, while achieving low-cost, compact and reliable solutions. There are several factors positively influencing energy efficiency; however some of the most effective ones involve the use of modern motors and power devices, exploiting novel motor control techniques and utilizing special stand-by and wake-up conditions of driving electronics. To successfully materialize such factors and meet multinational standards, smart microcontrollers, managing motors and auxiliary electronics, have to be employed.

Building off the popular series of high pin count HCS08 microcontrollers, Freescale is now introducing an advanced HCS08 family of low pin count devices called the MC9S08QG family.

Overview

The MC9S08QGx (QG) microcontrollers provide a rich feature set, balanced with a range of small packages which make it an outstanding choice for designers who need to design modern electronic equipment while taking into consideration primarily low power consumption, board space and system cost. The QG family provides high performance at low power consumption rates, robust flash memory, greater analog resolution, various serial communications and increased motion control capabilities. See Figure 1.

Figure 1: MC9S08QG8 Modules and Capabilities



HCS08 Processing Core

These microcontrollers are created by an advanced fabrication process. As a result, they run at up to 20 MHz (10 MHz bus) frequency at >2.1V operation and 16 MHz (8 MHz bus) frequency at <2.1V, providing high performance at low voltage levels. The HCS08 core uses the same instruction set as the HC08 core, with added support for entry into background debug mode (BGND instruction), thus taking full advantage of the integrated on-chip Background Debug Controller (BDC) and In-Circuit Emulation (ICE).

The BDC contains a set of non-intrusive commands allowing reads and writes of on-chip memory without CPU interruption. The ICE contains powerful division and trace units allowing setting of breakpoints, trigger options and capturing changes in software flow and execution.

The debugging cycle is carried out through a single wire communication interface employing only one BKGD pin of the device. After finishing a debug cycle, the BKGD pin can be used by the user applications as an additional general I/O pin. This enables users to eliminate expensive emulation tools by using the MC9S08QG on-chip debug capabilities without sacrificing effective pin count, thereby reducing overall development cost.

In the domain of battery-powered applications, the key requirement is minimizing the power consumption of the microcontroller while maintaining the processing needed by the application. To meet such requirements on the MC9S08QG devices, a special Wait and three additional Stop low-power modes (Stop3, Stop2 and Stop1) have been carefully implemented. Such listed low-power modes will be referred in further sections as “power saving” modes.

Clock Source Circuitry

The clock source distribution is a critical part of any system design. The MCS08QG family internal clock source (ICS) module provides a variety of clock source choices and forms a very flexible and cost-effective solution especially for low-pin count devices. It comprises a Frequency-Locked Loop (FLL), Internal Reference Clock (IRC), External Oscillator (XOSC) and Clock Selection Logic (CSL) modules.

The FLL increases the reference frequency by 512 while exploiting only simple digital logic techniques, eliminating the need for additional external components typically required for PLLs.

The IRC is fully trimmable and can be used either as a reference for the FLL or directly as the source for the CPU and bus clock. Its trimming is realized with a typical resolution of 0.1 percent. It can be trimmed to a frequency range of 31.25 kHz to 39.06 kHz.

The XOSC reference can be configured into three modes. The first, a low-frequency oscillator mode, is intended for use with any 32 kHz to 38.4 kHz crystals and resonators. The second, a high-frequency oscillator mode, can be used in connection with 1 MHz to 16 MHz crystals and resonators. The third oscillator mode is specifically tailored for use with an external active clock with a frequency of up to 20 MHz.

The CSL selects and divides clock sources for use by the CPU and additional microcontroller blocks. The FLL, IRC or XOSC sources can be selected to drive the CPU and bus. The output frequency can be divided by two, four or eight at any given time in order to slow down the CPU and peripheral execution, hence extending the battery life without switching the microcontroller into power saving modes.

On-Chip Memories

The family offers two flash/RAM memory configurations. Designers can either select the MC9S08QG8 with 8 KB flash and 512B of RAM or the MC9S08QG4 with 4 KB flash and 256B of RAM.

The flash memory is based on Freescale 3rd generation (0.25µ) technology allowing programmability down to 1.8V throughout the whole temperature range (-40°C to +85°C). The flash memory can be programmed byte-wise and gets erased page-wise (512 bytes)—up to 100,000 flash program/erase cycles are possible at room temperature, down to 10,000 cycles guaranteed for the entire temperature and voltage operating conditions. The flash data retention is engineered for a minimum of 15 years and beyond depending on system conditions. The proper flash partitioning and ultra-fast programming enable users to implement EEPROM emulation.

The MC9S08QG includes circuitry designed to prevent unauthorized access into flash and RAM memories, and thus prevents any potential theft of application code. The implemented mechanism yields a high level of security. Once a device is secured by this mechanism, it is engineered to only be unlocked by entering the proper 8-byte back-door comparison keys. Important non-volatile data and code, such as application constants and bootloader, can also be optionally protected against unintentional write and erase cycles.

Advanced Timers

There are two independent timer modules available on the MC9S08QG family. These modules can be used for timing purposes, motion control and/or software task scheduling.

The first is a two-channel programmable 16-bit Timer/PWM Module (TPM). Each channel of the TPM can be independently configured for input capture, output compare, buffered edge-aligned or centre-aligned Pulse Width Modulation (PWM). Both supported PWM modes, particularly the centre-aligned one, brings into system powerful motion control capabilities. In addition, the timer registers are handled in a way that achieves consistent 16-bit duty cycle updates and thus avoids generating unexpected pulse widths.

The second timer module is an 8-bit Modulo Timer Interrupt Module (MTIM) with several possible clock input sources and a programmable interrupt capability. The timer can either operate as a free running or modulo timer with the possibility of scheduling periodic interrupts on the timer overflows.

Analog Circuitry

The MC9S08QG family has an integrated 8-channel, 10-bit analog-to-digital converter (ADC) providing a simple interface to any sensors featuring analog output signals. The ADC can perform single or continuous conversions on a selected channel. Every conversion can be initiated by software or a Real Time Interrupt (RTI). The ADC may be clocked either from the bus clock, bus clock/2 or ADC internal clock source. In addition, the ADC can be configured to continuously compare a measured voltage to either a given lower or upper limit and to generate an interrupt in case of any transition below or above these limits, respectively. It's important to remember, that this feature can be used to wake up the microcontroller from power saving modes provided the ADC internal clock is selected as the ADC clock source.

The device consists of an Analog Comparator (ACMP) module for comparing two analog input voltages or for comparing one analog input voltage to an internal bandgap reference. This module routes its output to some other on-chip peripherals and on to an external pin. This enables other on-chip peripherals and external systems to perform continuous monitoring and post processing of the comparator output signal. Optional comparator interrupts can be triggered by the rising, falling or both the edges of the comparator outputs. Such an interrupt may be used to bring the microcontroller out of the power saving modes.

The device also contains a temperature sensor whose output is connected to one of the reserved input channels (AD26). This is designed to enable the user to measure temperature of the silicon die without sacrificing any analog input channel. In addition, the internal bandgap reference and power supply voltages can be measured in the same way in order to check the correct operation of the on-chip bandgap voltage regulator and external power supply, respectively.

Communication Modules

There are three communication modules available on the MC9S08QG8 (8- and 16-pin) and MC9S08QG4 (16-pin) devices allowing designers a variety of options in meeting their application requirements.

The Serial Communication Interface (SCI) provides standard UART communications allowing full duplex, asynchronous and non-return-to-zero (NRZ) serial communication between the microcontroller and additional remote devices.

The Serial Peripheral Interface (SPI) enables establishing high speed master (up to 5 Mbit) and slave (up to 2.5 Mbit) SPI connections to additional microcontrollers and peripherals, such as ZigBee® transceivers (MC13192), stepper motor drivers (MC33976), various I/O expanders and smart media card controllers.

The Inter-Integrated Circuit (I²C) provides a method of communication between a number of devices. The interface is designed to operate with maximum bus loading and timing up to 100kbps. This offers the designer an easy interface with LCD controllers, RTCs, DACs and additional external EEPROMs.

MC9S08QG Portfolio and Packaging Options

The MC9S08QGx devices extend the portfolio of Freescale's HCS08 devices into the low pin count domain, while increasing functionality at low voltages. There are 10 devices initially being supported. They differ by the amount of on-chip flash and RAM memories, pin counts and package options. Pin compatibility between 8 KB flash and 4 KB flash devices is supported across all package options. The devices belonging to the MC9S08QG portfolio are listed in Table 1.

Development Tools

The effective and fast programming of MC9S08QG microcontrollers is fully supported by the CodeWarrior® tool chain for HC(S)08 Microcontrollers. For code sizes up to 16 KB, a complimentary special edition of the FastTrack CodeWarrior tool is provided and contains an efficient C compiler, debugger, linker, assembler, simulator, Integrated Development Environment (IDE) and other components supporting rapid, comfortable and worry-free application development.

Every embedded application requires initialization and the use of on-chip peripherals. Due to this requirement, Freescale has licensed the Processor Expert™ auto-code generator and integrated it into the CodeWarrior tool. The Processor Expert auto-code generator significantly simplifies and speeds up the process of application development. In addition, it enables users to program on-chip peripherals without an in-depth knowledge of their structure.

To enable starting a development without the availability of the final application hardware, Freescale offers the DEMO9S08QG8 evaluation module. This module supports fast and easy development of applications through an integrated USB-BDM debugging interface and various software examples demonstrating the programming of on-chip peripherals.

Typical Applications

The MC9S08QG family is increasingly popular for small appliances, power tools, intelligent relays, small industrial drives, automotive electronics, light dimming equipment, handheld tools, wireless communications, scales, heating meters and alarm applications, to name just a few.

Example applications include using three-phase brushless direct current (BLDC), brush direct current (DC) and universal (AC) motor types. These motors, when controlled by the MC9S08QGx microcontrollers, achieve excellent performances at high energy efficiency rates.

First, the successful BLDC drive concept is shown in Figure 2. It is de facto a speed closed-loop BLDC drive employing inexpensive Hall sensors as position and speed feedbacks. The advantage of the MC9S08QG4 device in such a drive lies in maintaining the highest level of component integration and optimal usage of motion control peripherals. This drive requires implementation of “NAND Logic” in the external hardware in order to programmatically extend one PWM device output into three-phase power stage driving capabilities.

The BLDC drive concept can be used in white goods applications, high-end pumps, fans and in other applications requiring a high reliability and efficiency at a lower cost.

Second, the control scheme intended for battery powered drives is presented in Figure 3. This control concept demonstrates the successful microcontroller integration into the brush DC motor control application. The microcontroller driving such an application easily allows implementation of several advanced drive control techniques—for example, close-loop sensorless speed control or torque control.

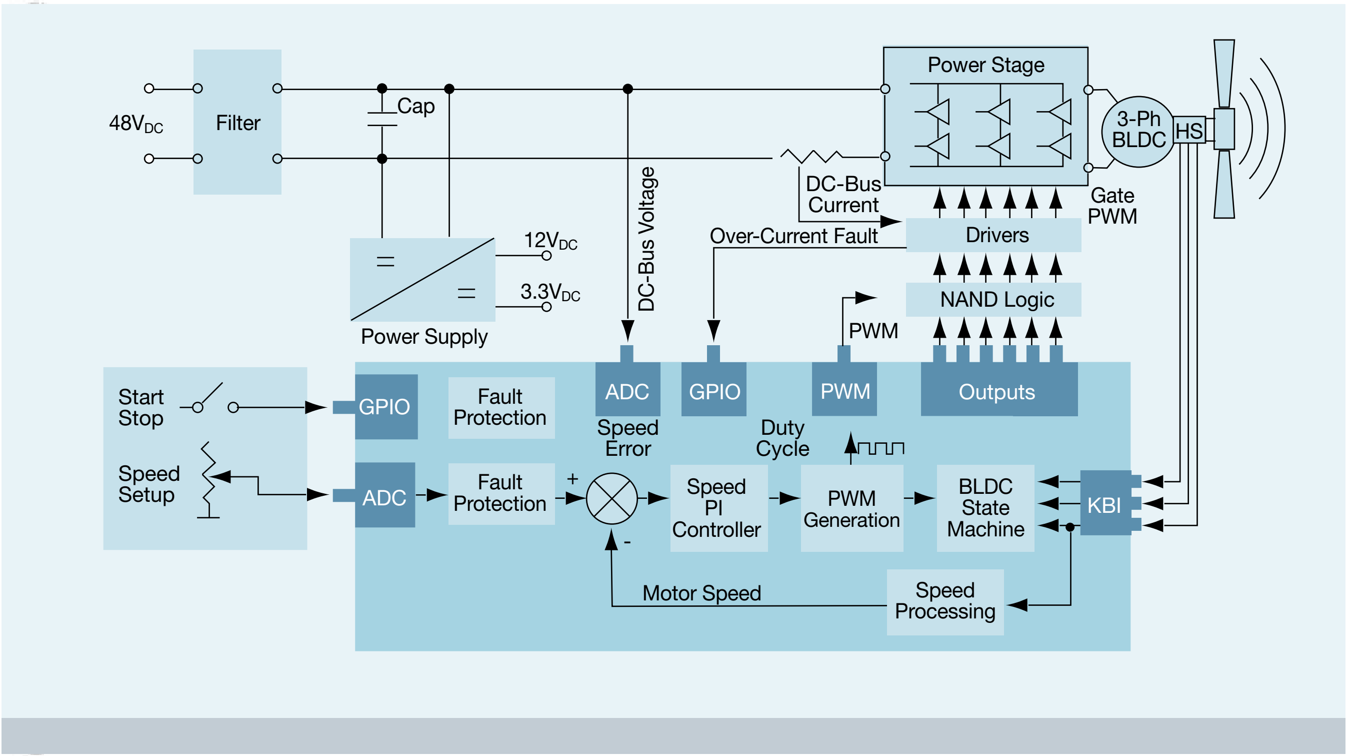
This drive concept can be used for controlling brush DC motors in cordless screwdrivers, battery powered drills, blenders, electric toothbrushes, toy cars or even golf trolleys.

Finally, the implementation of the universal motor drive is shown in Figure 4. The application presents a low-cost, closed loop universal motor chopper control drive system based on the MC9S08QG4 microcontroller. A PWM technique is used to adjust the voltage applied to the motor. Modulation of the PWM’s duty cycles allows the average value of the voltage applied to the motor to be varied. Compared to a phase angle drive, the chopper drive requires an input power rectifier. However, its advantage is a higher efficiency, with less acoustic noise and better EMC compliancy.

Table 1: Devices in the MC9S08QG Portfolio

	Device			
Feature	MC9S08QG8		MC9S08QG4	
Package	16-pin	8-pin	16-pin	8-pin
Flash	8 KB		4 KB	
RAM	512B		256B	
XOSC	yes	no	yes	no
IRC	yes		yes	
ICS	yes		yes	
ADC	8-ch.	4-ch.	8-ch.	4-ch.
ACMP	yes		yes	
DBG and ICE	yes		yes	no
IIC	yes		yes	
IRQ	yes		yes	
KBI	8-pin	4-pin	8-pin	4-pin
MTIM	yes		yes	
SCI	yes	no	yes	no
SPI	yes	no	yes	no
TPM	2-ch.	1-ch.	2-ch.	1-ch.
I/O pins	12 I/O 1 Output only 1 Input only	4 I/O 1 Output only 1 Input only	12 I/O 1 Output only 1 Input only	4 I/O 1 Output only 1 Input only
Package Types	16 PDIP 16 QFN 16 TSSOP	8DFN 8 SOIC	16 QFN 16 TOSSP	8 DFN 8 PDIP 8 SOIC

Figure 2: MC9S08QG4 16-Pin QFN/TSSOP Diagram



The universal motor concept can be used in washing machines, hand tools, vacuum cleaners, dishwashers, single-phase and variable speed drives, and with all low-cost and medium-power applications with variable voltage output.

Today, new international standards on Safety in Home Appliances (IEC 60730) are calling for an independently clocked watchdog timer (COP). Such a timer is capable of putting the system into a deterministic reset state not only in cases of

Figure 3: MC9S08QG4 8-Pin SOIC/DFN Diagram

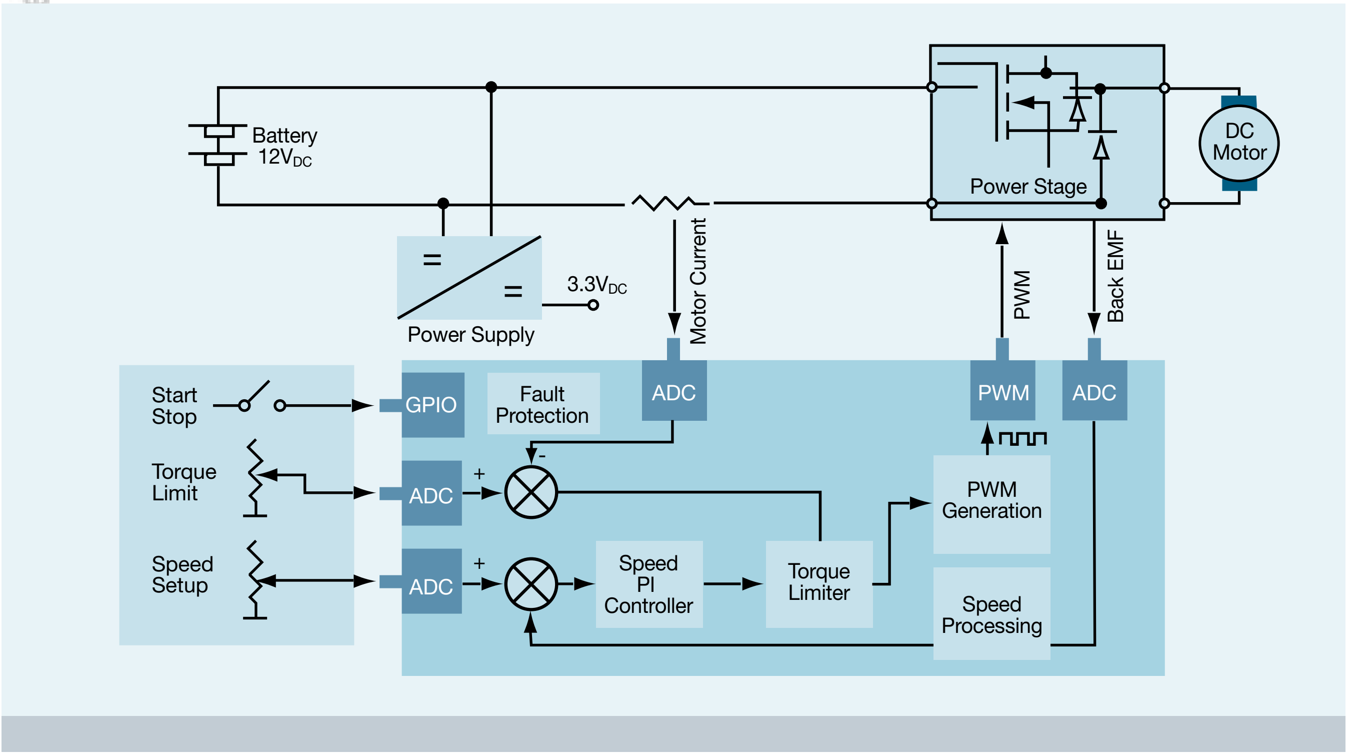
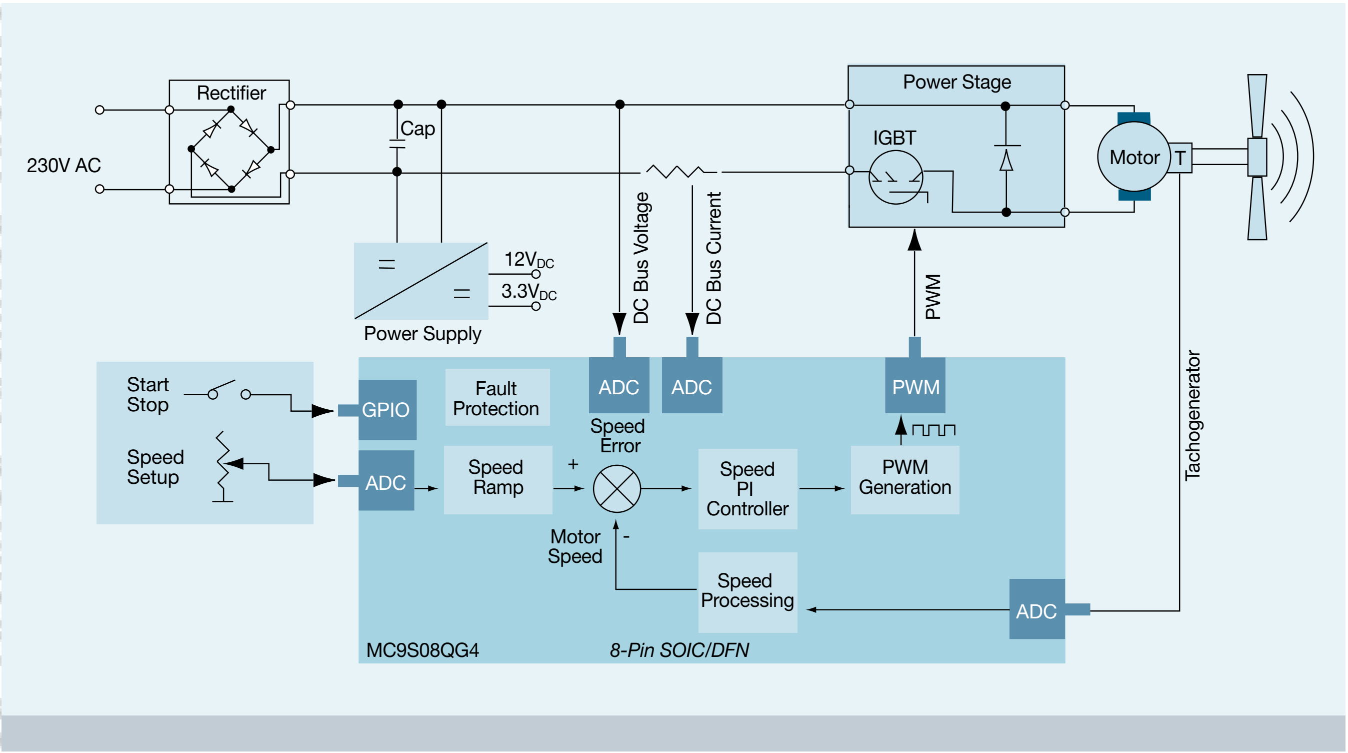


Figure 4: MC9S08QG4 8-Pin SOIC/DFN Diagram



software errors (software stack corruption, program counter corruption, failure to return from interrupt or subroutine and infinite loop), but also in situations where the clock circuit is not working properly. This safety feature is fully implemented on the MC9S08QGx microcontrollers, thus making them suitable for driving home appliances with regard to safety and the new international standards.

Conclusion

The recently launched MC9S08QG family of microcontrollers, their processing core, on-chip peripherals, power saving features and development tools are an ideal solution for energy-efficiency-sensitive control applications. Freescale enables significant cost reductions in the development as well as the final production stages with the MC9S08QGx family of microcontrollers. In the development stage, the advanced debugging capabilities, CodeWarrior full-featured tool chain and Processor Expert package are complimentary. In the production stage, the internal clock source module, analog circuitry and EEPROM emulation reduce the need for external components like crystals or resonators, analog comparators and serial EEPROMs, otherwise indispensable to be populated on the Printed Circuit Board.

Freescale realizes that maintaining a leadership position in the 8-bit microcontroller global market (Gartner Dataquest, 2004) is feasible only through substantial investments towards new technologies and particularly by continual development of smarter and faster devices at a reduced price. Therefore, Freescale engineers are continually working on further extensions of the popular HCS08 microcontrollers. The future families are expected to expand the existing low and high pin count products bringing further performance improvements and expanding the set of on-chip peripherals.

Martin Mienkina received the Master of Science and Ph.D. degrees in electrical engineering from the University of Zilina in 1992 and 1997, respectively. He spent a period in industry with Trinec Steel & Iron Works Ltd., designing modern control systems related to steel and rail production. Since 2000, he has been acting in Freescale Czech System Centre (Roznov pR) in the position of system application engineer primarily focusing on motor control applications, software drivers and Digital Signal Controllers. Dr. Mienkina leads Freescale technical marketing activities in Eastern European countries with the main focus on the demand creation and new product development.

Juan Cazares

Building Simple Wireless Applications
With low-end microcontrollers

Wireless communications are part of our daily lives. In places like the office, school or at home, we are in touch with wireless communication devices, including laptops, printers, cameras, handhelds, access and lighting controllers. The complexity of these devices depends on the type of tasks they perform.

Many of these wireless applications execute simple tasks with small microcontrollers and small codes. This article will discuss a wireless application used for home automation. The application is a wireless air controller and is designed to be a low-cost application by using only three electronic devices—one low-end microcontroller (MC9S08QD4), one RF transceiver (MC13192) and one LCD2x16. The other devices needed are passive components like resistors and push buttons.

Figure 1: Air Controller Block Diagram

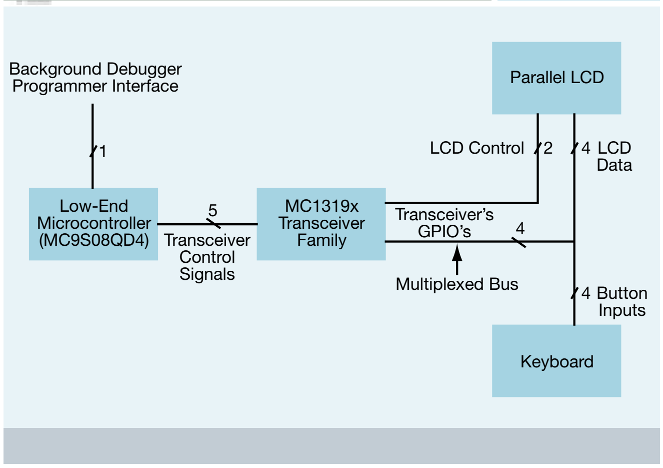

Background Debugger Programmer Interface
1
Low-End Microcontroller (MC9S08QD4)
5
Transceiver Control Signals
MC1319x Transceiver Family
Transceiver's GPIO's
4
Multiplexed Bus
Parallel LCD
LCD Control
2
4 LCD Data
4 Button Inputs
Keyboard

Figure 1 shows the air controller block diagram. The interface used to exchange data between the transceiver and the microcontroller is a serial peripheral interface (SPI). This interface allows reading and writing of the configuration, status and control registers of the transceiver. Also the SPI allows reading and writing of the RAM located inside the transceiver, which is used to transmit and to receive data via RF.

Another signal needed for the communication between the transceiver and the microcontroller is interrupt request (IRQ). The IRQ pin is handled by the transceiver. When the status register in the transceiver changes, a falling edge transition on the IRQ pin is generated. When an IRQ is generated, the first thing the microcontroller executes is a reading of the status register to determine the specific event that generated the interruption. Table 1 shows the microcontroller pins assignment

Figure 2: Microcontroller and Transceiver Connections

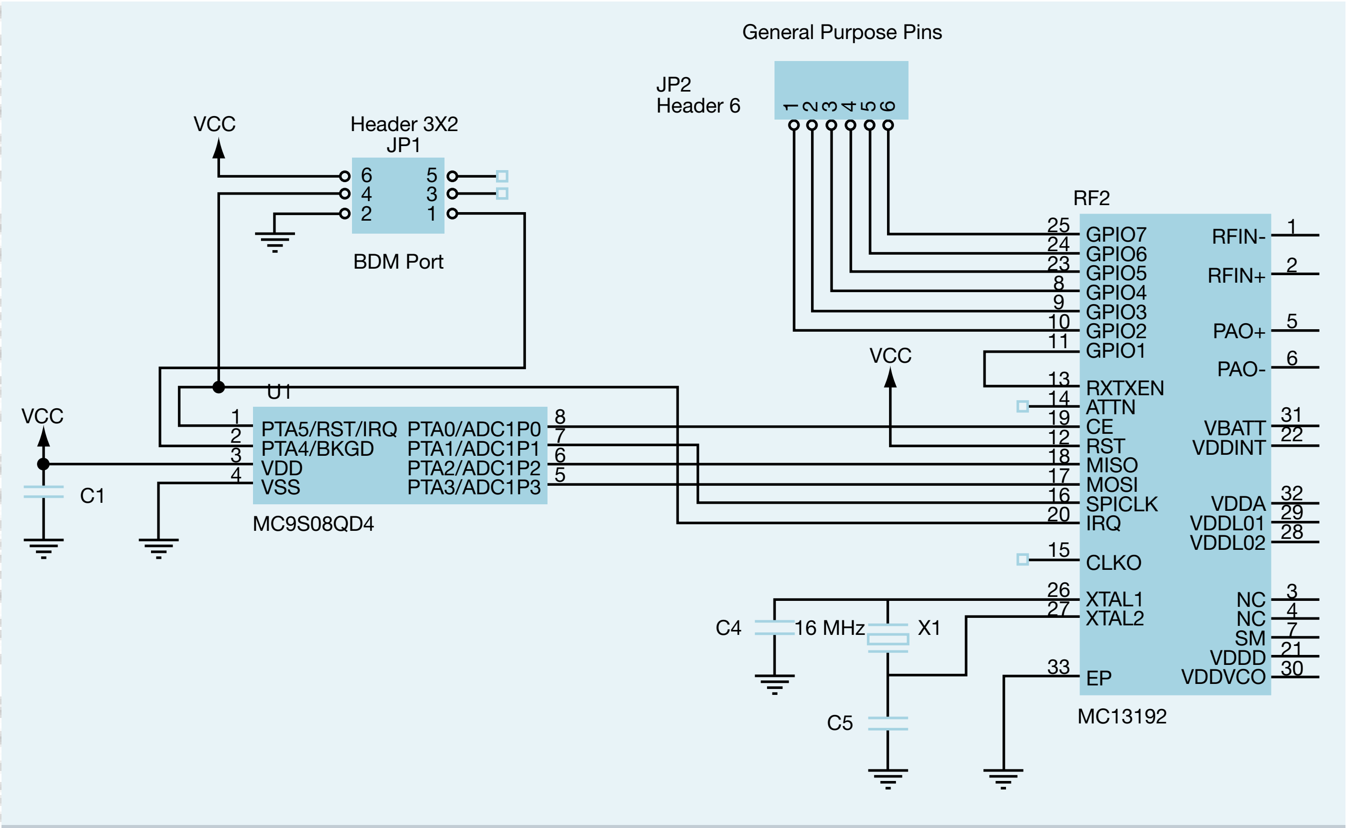

General Purpose Pins
JP2
Header 6
1 2 3 4 5 6
VCC
Header 3X2
JP1
6 4 2 5 3 1
BDM Port
RF2
GPIO7 GPIO6 GPIO5 GPIO4 GPIO3 GPIO2 GPIO1
RXTXEN ATTN CE RST MISO MOSI SPICLK IRQ CLKO XTAL1 XTAL2 EP
RFIN- RFIN+ PAO+ PAO- VBATT VDDINT VDDA VDDL01 VDDL02 NC NC SM VDDD VDDVCO
MC13192
U1
PTA5/RST/IRQ PTA4/BKGD VDD VSS
PTA0/ADC1P0 PTA1/ADC1P1 PTA2/ADC1P2 PTA3/ADC1P3
MC9S08QD4
C1
C4
16 MHz
X1
C5

Table 1: Pin Function Description

Pin #	MCU Pin Name	Type	Pin #	Transceiver Pin Name	Type
1	PTA5/ RST/IRQ	Digital Input	20	IRQ	Digital Output
2	PTA4/ BKGD	Programming and Debugging interface			
5	PTA3/ KBI1P3	Digital Output	17	Mosi	Digital Input
6	PTA2/ KBI1P2	Digital Input	18	Miso	Digital Output
7	PTA1/ KBI1P1	Digital Output	16	SPI clock	Digital Clock Input
8	PTA0/ KBI1P0	Digital Output	19	Chip Enable	Digital Input

One more signal is needed by the transceiver to switch the mode of operation (reception, transmit, doze, hibernate or idle). This signal is called RXTXEN. All of the general-purpose pins on the microcontroller are used to manage some other signals, as Table 1 shows. The RXTXEN signal will be handled by the GPIO1 pin provided by the transceiver. Table 2 describes the connections needed to control RXTXEN.

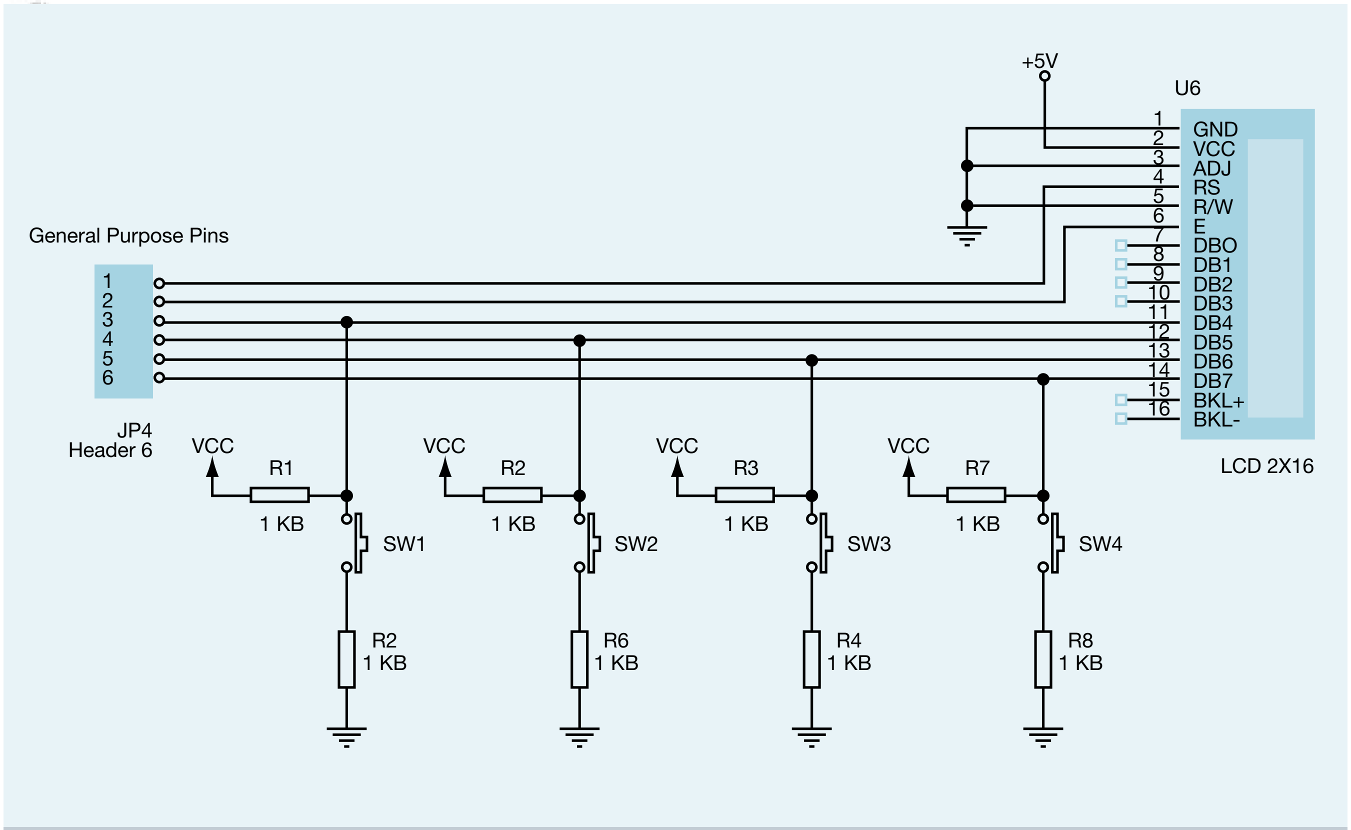
Table 2: RXTXEN Connection

Pin #	MCU Pin Name	Type	Pin #	Transceiver Pin Name	Type
11	GPIO1	Digital Output	13	RXTXEN	Digital Input

Table 1 and Table 2 show the minimum hardware requirements to get the transceiver working. See the suggested schematic in Figure 2. With the hardware connections described above the transceiver can be configured, can transmit and can receive data. Also the timers and GPIO pins provided by the transceiver can be configured via the SPI and used for any desired purpose.

As Figure 2 shows, the transceiver has seven GPIO pins. One of them is used to drive the RXTXEN signal, and the other six can be used for the user application. In this proposed application, the timers and GPIO pins provided by the transceiver will be used to drive four push buttons and one LCD 2 x 16.

Figure 3: Air Controller Connections



The proposed schematic for the wireless air controller application is separated in two parts. The first part of the schematic contains the basic platform to develop any application (described in the Table 1, Table 2 and shown in Figure 2). The second part of the schematic shows the connections for the air controller application (shown in the Figure 3).

Figure 3 shows that the application requires six lines to drive the LCD and four more lines to drive the four push buttons. It has a total of 10 lines that must be driven with only six GPIO pins. To do so, the four data lines of the LCD and the four push buttons must be multiplexed. Table 3 describes the performance of each GPIO pin.

Table 3: Application Pin Description

Transceiver GPIO Pin	Function
GPIO2	RS
GPIO3	E
GPIO4	DB4/SW1
GPIO5	DB5/SW2
GPIO6	DB6/SW3
GPIO7	DB7/SW4

As Table 3 shows, the GPIO4, GPIO5, GPIO6 and GPIO7 are used to drive to the four data lines of the LCD and the four push buttons. The four GPIO pins must be configured as output to put data in the LCD, and these must be configured as input to read the push buttons level.

The application uses the LCD to display different menus, such as current temperature, set time-out and device out of range.

The four push buttons are used to set the desired temperature, to set the fan time-out and to navigate between the menus. Table 4 shows the assigned function for each push button.

Table 4: Push Button Functionality

Push Button	Function
SW1	Show time-out menu
SW2	Decrease value
SW3	Increase value
SW4	Accept

When the application starts, the fan is always off. To turn on the fan, the SW2 or SW3 must be pressed. When the fan is turned on, the current temperature is displayed on the LCD. Once the fan is turned on, the Sw2 and Sw3 can be used respectively to decrease or increase the temperature. The maximum temperature that can be set is 60 degrees. When the fan is off, the SW1 and SW4 do nothing.

A time-out period can be set to turn off the fan automatically. The time-out period can be set from 1 to 99 minutes. If there is no time-out period set, the fan will never turn off automatically. To set the desired time-out:

1. Turn on the fan. If the fan is off, it is not possible to set a time-out period.
2. Press SW1. This will display the time-out menu on the LCD.
3. Press SW2 or SW3 to decrease or increase the desired time-out period.
4. Press SW4 to accept the new time-out period.
5. Once the new time-out period is accepted, the current temperature is again displayed on the LCD.

The application uses two timer comparators provided by the transceiver, one to generate a time-out and the other to read the push button level.

The timer comparator1 is used to generate a time base equal to one minute. This time base allows generating time out periods from 1 to 99 minutes.

The timer comparator3 is used to read the push button level each 100 milliseconds. To read the push button level, the GPIO4, GPIO5, GPIO6 and GPIO7 must be configured as input. The application configures the necessary GPIO pins as inputs, reads the input value and configures the GPIO pins as outputs again.

A simple protocol is used to send the current temperature to the heating-cooling system each time the temperature is modified by either SW2 or SW3 push button.

Conclusion

Using an 8-pin-package microcontroller with few hardware resources illustrates that high-performance microcontrollers are not always needed to develop wireless applications. The transceiver requires a few pins, allowing it to be used with almost any microcontroller from low-end to high-end. Finally, the use of the timers and GPIO pins provided by the transceiver reduces the use of microcontroller resources such as GPIO pins and other modules, like timers. It is also important to keep in mind that low-end microcontrollers can do the simple tasks required in a complex wireless network, reducing the cost of the entire solution.

References

www.freescale.com/files/rf_if/doc/ref_manual/MC13192RM.pdf

www.freescale.com/webapp/sps/site/overviewjsp?code=
ZIGBEE_REFERENCEDESIGN_HOME

Juan Cazares is an application engineer focused on wireless solutions involing SMAC stacks. He has a degree in electronic engineering and has a speciality in control and instrumentation. Before joining Freescale, Cazares worked as a hardware and software designer for gas and gasoline measurement devices.

Gary Streber

Freescale Design Alliance Program
Creating solutions together

As high tech products, markets and companies evolve, the goal to get to market faster with the best solution has remained the same. Better, faster and cheaper are the continued trends. On the other side of the coin, with fluctuating economies, rightsizing and having the available technology, core competencies and everything else needed to get your customers' products out at the right time becomes an ever greater burden than in the past.

Freescale has some of the most compelling technologies and products in the world and one of the best ecosystems in the industry that is filled with top notch experts with the latest application hardware and software know-how. However, we still may not have all of the technology and resources internally you may need to help you get finished product to market.

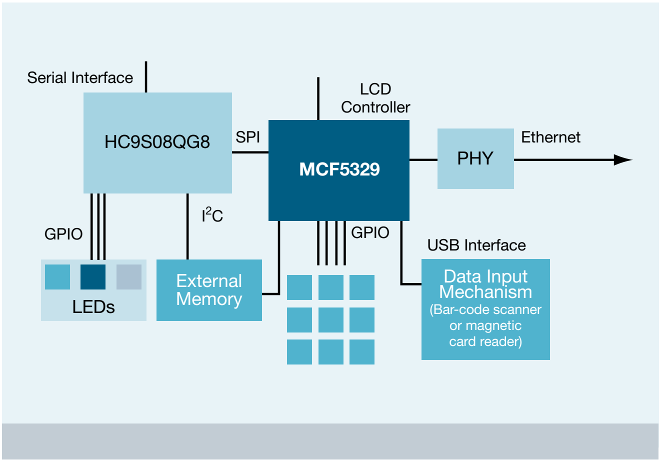
The Design Alliance program (DAP) has been developed so Freescale and our solution ecosystem can help you meet your goals. DAP has over 450 third party members with varying experiences in product and application creation. They can help get your product to market by getting involved with you at any phase, from concept to finished product. We also have the Tools Alliance program (TAP) with about 200 members that offer finished hardware and software tools that can be used to develop solutions by you or your partners. Also, a University Alliance program is in place to help ensure that the future design community is up to speed with Freescale's tools and products.

To help validate the credibility of our DAP membership, we take steps to verify the Freescale experience and applications listed in our directory. Although, as in many cases involving new product launches and application creativity, the listings may be only a subset of Freescale's full capabilities.

Many of the DAP members operate in a pure design-for-hire or non-recurring engineering (NRE) fee structure to provide the customer the full design and any generated intellectual property (IP). There are others who have created and own IP that can be licensed and/or included as part of a design to help improve the design cycle. Also, with their vast applications and varied market knowledge, DAP members can provide different viewpoints to every design, helping to maximize its market potential.

Indesign LLC and Freescale's Regional Technology Applications Center joined forces to design this POS reference design (Figure 1) utilizing the 32-bit ColdFire MCF5329 embedded controller and MC9S08QG 8-bit microcontroller. For more details on Indesign LLC, visit www.indesign-llc.com.

Figure 1



We recommend DAP members to customers seeking services or who need help to design a product. We have developed many of our own application reference designs, demonstration boards and evaluation boards with DAP members. Some examples include: a point of sale (POS) reference design with Indesign LLC; a ZigBee demo system with FSI Systems and a high brightness light emitting diode (HBLED) demo design with i2systems.

Besides these examples, we have other members with various ZigBee ready platform boards; very complex boards based on Power Architecture technology for the networking, communications, industrial and commercial markets; portable cellular handsets; PLC controller platforms; motor control; consumer reference designs and so on. It is very likely that we have a DAP member with the necessary experience and capability to fill any void you may have.

FSI Systems has worked with Freescale on multiple occasions. This is an example of a ZigBee-enabled system (Figure 2) we have been demonstrating to customers and developers since 2005. For more details on FSI Systems, visit www.fsisys.com.

i2 Systems designed this handy HBLED demo (Figure 3) for Freescale showing how to drive and manage the power control for HBLEDs with the MC68HC908QY4 microcontroller. It was introduced in July 2006 at the Freescale Technology Forum in Orlando, Florida. For more details on i2 Systems, visit www.i2systems.com.

Figure 2

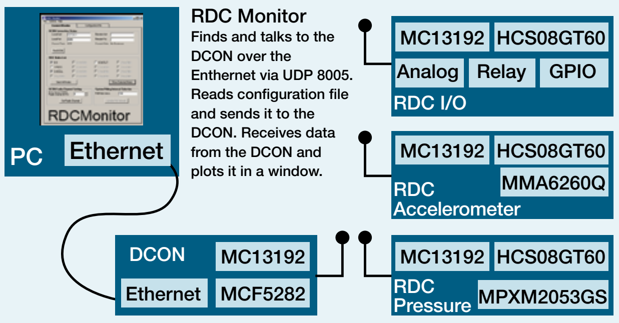
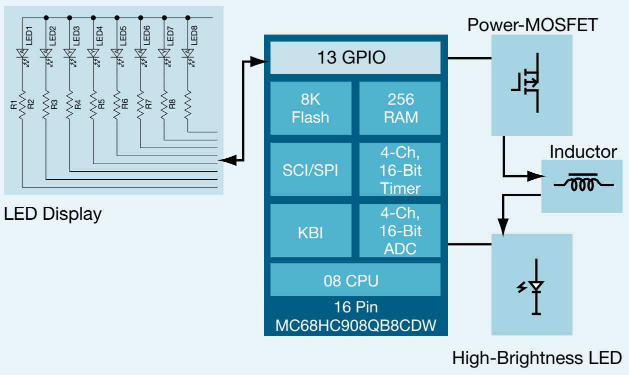


Figure 3



Our DAP members have access to our future roadmaps and preliminary specification, which give them the ability to help ensure the solution they develop for you is the best solution with a solid migration path into the future.

Many of the DAP members participate in our “Alpha” silicon program, which gives them the opportunity to have early silicon access. This access allows those members to be ready to integrate the latest technology into your products. Freescale also has access to new product inputs from customers as well as the DAP community to help us bring you the best possible enabling technology in the industry.

You can find a DAP member with the experience you need by visiting our member directory and selecting the services needed and any filtering desired to find the best possible match at www.freescale.com/webapp/sps/site/dap.search.framework?NEXT_SCREEN=SEARCH.

You can also type a keyword or product name in our search engine at www.freescale.com. The resulting output will have additional filtering options on the left menu bar. Select “third party support” for a list of various alliance members, which can be further filtered from the left menu bar.

To learn more about the Design Alliance Program go to www.freescale.com/designalliance.

When a customer needs help bringing a product to market better, faster and cheaper, we are confident with Freescale’s 53+ years of innovation and technology plus the experience of our DAP members, we, as a team, have the capability to make it happen.

Gary Streber has been involved in customer-focused semiconductor technical sales and third party relations for over 25 years.

Pavel Grasblum

BLDC Motor Control Using 8-bit MCU

Cost-effective sensorless control of BLDC motor

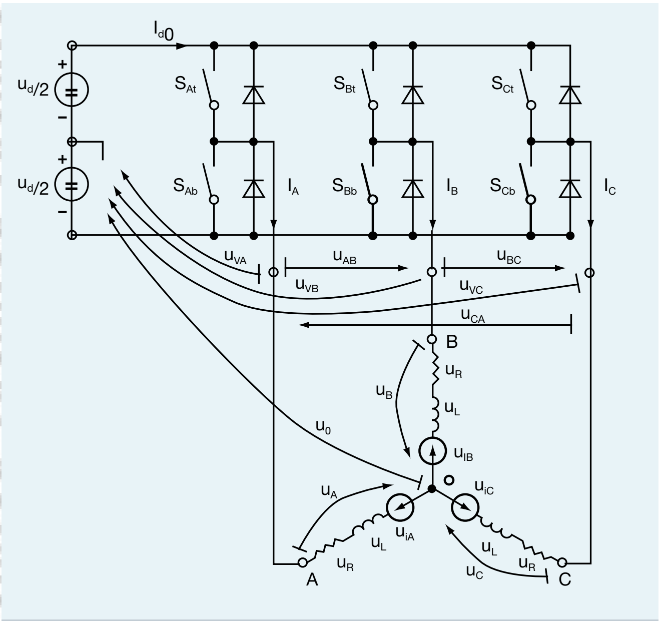
Recent progressive variable speed drives have been designed to increase product performance and system efficiency. Specifically, the applications where direct current (DC) motors were dominant are now designed using a brushless DC (BLDC) motor. The DC motors are very popular in variable speed drives due to simple speed control circuits. On the other hand, due to the brushes, DC motors suffer from a lower reliability, since the brushes wear down and require maintenance or replacement. This DC motor drawback can be eliminated by using a BLDC motor.

The BLDC motor is also referred to as an electronically commuted motor. There are no brushes on the rotor and the commutation is performed electronically at certain rotor positions. The stator magnetic circuit is usually made from magnetic steel sheets. The stator phase windings are inserted in the slots (distributed winding), or they can be wound as one coil on the magnetic pole. The magnetization of the permanent magnets and their displacement on the rotor are chosen in such a way that the back-EMF (the voltage induced into the stator winding due to rotor movement) shape is trapezoidal. This allows a three-phase voltage system, with a rectangular shape, to be used to create a rotational field with low torque ripples. In this respect the BLDC motor is equivalent to an inverted DC commutator motor, in that the magnet rotates while the conductors remain stationary. In the DC commutator motor, the current polarity is reversed by the commutator and brushes. In the brushless DC motor, the polarity is reversed by semiconductor switches, which must be switched in synchronization with the rotor position. Eliminating the commutator results in higher reliability and removes a factor that limits the DC motor's maximal speed. Therefore, the BLDC motor can be employed in applications requiring high-speed.

Replacing a DC motor with a BLDC motor places higher demands on a control algorithm and a control circuit. First, the BLDC motor is usually a three-phase system. Thus it has to be powered by a three-phase power supply. Second, the rotor position must be known at certain angles in order to align the applied voltage with the back-EMF. The alignment between back-EMF and commutation events is very important. In this condition the motor behaves as a DC motor and runs at the best working point. These drawbacks are balanced by the BLDC motor's excellent performance and reliability and the ever falling prices of power components and control circuits.

The simple motor-drive model of a BLDC motor-header (Figure 1) consists of a three-phase power stage plus a brushless DC motor. The power for the system is provided by a voltage source (Ud). Six semiconductor switches (SA/B/C t/b) allow the rectangular voltage waveforms to be applied. The semiconductor switches and diodes are simulated as ideal devices. The trapezoidal control of the BLDC motor is based on energizing only two phases at a time. This means that one top and one bottom semiconductor switch are turned on, each in a different phase. Thus the last phase has both top and bottom switches switched off and is disconnected from the voltage source. Considering all possibilities, we get six possible combinations to energize two phases in the three-phase system. Therefore this technique is also called a six-step control.

Figure 1: BLDC Motor—Drive Model



The most common way to control a BLDC motor is to use hall sensors to determine the rotor position. The control system senses the rotor position and the proper voltage pattern is applied to the motor. However, there are at least two reasons why it might be desirable to eliminate the position sensor. First, the hall sensors need additional connections between the BLDC motor and the control system (usually five wires). Second, eliminating the sensor helps to reduce the overall cost of the system.

Using Figure 1, a mathematical model of the system can be derived. An analysis of the mathematical model shows one of the possible approaches to the sensorless BLDC motor control. This approach is called the back-EMF zero-crossing method, resulting in the comparison of the back-EMF voltage induced in the non-fed phase to half of the DC bus voltage. The zero-crossing event happens in the middle of the commutation period. This assumption is valid only when both semiconductor switches are switched on and no current is going through the non-fed phase used to sense the back-EMF voltage.

The BLDC motor drive, based on the back-EMF zero crossing algorithm, can be implemented in different ways. The simplest is to use a dedicated analog circuit. This implementation is very easy and fast. However the implementation is fixed by hardware and cannot be tailored to customer requirements. Another way is to use a simple 8-bit microcontroller (MCU) in combination with an external circuit evaluating the zero-crossing signals.

There are also 8-bit MCUs with dedicated peripherals for zero-crossing detection. Another way is to use a digital signal controller (DSC) powerful enough to evaluate zero-crossing signals. Freescale has already published several application notes and design reference manuals that explain the implementation of the back-EMF zero-crossing algorithm—AN1913, DRM070 and DRM028. Both the AN1913 and DRM077 describe implementation using the 56F80x and 56F8013 DSCs. The latter describes the implementation using the 8-bit MC68HC908MR32 MCU and external comparators for evaluating the zero-crossing signals. This article deals with implementing of the back-EMF zero-crossing algorithm using the advanced 8-bit MC9S08AW60 MCU.

The MC9S08AWx family of devices is based on the high-performance HCS08 core, supporting 5-volt applications. It is a highly integrated, high-performance family, which comes packed with valuable features, such as 16–60 KB flash memory, 1–2 KB RAM, a flexible internal clock generator that eliminates the need for external components, low voltage detection, analog-to-digital converter (ADC), serial communication modules and many more. The 9S08AWx family is designed to be very robust in a variety of environments with very good electromagnetic compatibility (EMC) performance, and it is qualified for automotive applications. This family is suitable for appliances (white goods), industrial and automotive applications.

The key core features of the MC9S08AWx family are:

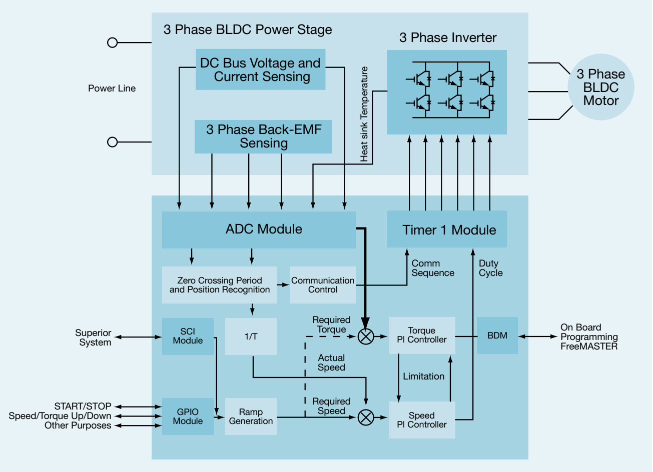
- 8-bit HCS08 central processor unit (CPU)
- 20 MHz internal bus frequency
- 16–60 KB on-chip in-circuit programmable flash memory with block protection and security options
- 1–2 KB on-chip RAM

The key peripherals for motor control applications are:

- 16-channel, 10-bit ADC with automatic compare function
- One 2-channel and one 6-channel 16-bit timer/pulse-width modulator (TPM) module with selectable input capture, output compare and edge-aligned PWM capability on each channel. Each timer module may be configured for buffered, centered PWM on all channels
- 2 x SCI, 1 x SPI, 1 x I^{2}C for communication with the supervisory system

Furthermore, the MC9S08AWx family is the first HS08 family offering a six channel timer/PWM module, which is necessary for full three-phase AC induction and BLDC motor control. The high-performance core with a 20 MHz bus frequency enables fully digital sensorless BLDC motor control without any external comparators.

Figure 2: Application Block Diagram



The application described below covers the following features of the BLDC motor drive:

- Sensorless back-EMF zero crossing algorithm implementation
- MC9S08AW60 MCU control
- Back-EMF sensing by ADC
- Full four-quadrant operation
- Both directions of rotation
- Speed closed loop with PI Controller
- Torque closed loop with PI Controller
- Speed range: 100–1200 rpm (motor dependent)
- Manual Interface
- FreeMASTER Interface

Figure 2 is the application block diagram. The BLDC motor drive consists of a three-phase power stage, three-phase BLDC motor and a MC9S08AW60 controller board. All measured signals are conditioned on the three-phase power stage into the ADC range. There are six quantities measured: DC bus voltage, DC bus current, back-EMF voltages of all three phases and the temperature of the heat sink, if available.

The BLDC motor drive can be controlled manually by the START/STOP switch and UP/DOWN buttons. The drive operation is enabled in the START position, and can be run in two operational modes: speed mode and torque mode. In speed mode you can set the required motor speed by the UP/DOWN buttons. The required speed is compared with the actual motor speed. The speed difference inputs to the speed PI controller, which maintains the desired motor speed independently of the motor load. If the motor achieves the maximal allowed torque, the drive goes automatically into torque mode and motor speed is maintained so the maximal torque limit is not exceeded.

In the torque operational mode the drive works in a similar manner. The required torque (DC bus current limit) is set by the UP/DOWN buttons, and the torque PI controller maintains the desired motor torque. If the motor reaches the maximal speed, the drive starts to maintain maximal motor speed. Both the speed and torque PI controllers are executed at a time independent of the desired operational mode. Both the PI controller outputs are compared and the lesser is applied to the motor. This automatically ensures a smooth transition between the speed and torque operational modes. To avoid saturating the integral parts of the PI controllers, they are limited by each other. If the speed controller is acting, the integral part of the torque PI controller is limited by the integral part of the speed controller, and vice versa.

For remote control, the FreeMASTER interface using a PC is available, or the drive can be controlled by a master system via SCI.

The PWM modulation is performed by a six-channel timer/pulse-width modulator (PWM) module. The modulation techniques can be differentiated according to the voltage applied to the motor or according to the switching of opposing semiconductor switches. If the motor sees either the DC bus or zero voltage during one PWM cycle, we refer to this modulation technique as unipolar. If the motor sees both polarities of the DC bus voltage during one PWM cycle, the technique is called bipolar.

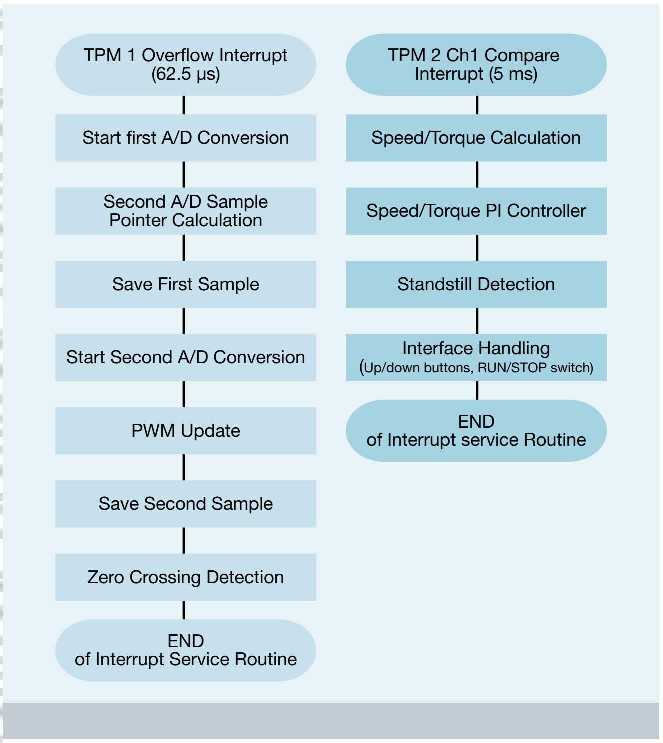
A second differentiator is the way of switching opposing semiconductor switches. If they are switched independently, we refer to it as independent switching. If they are switched in a complementary manner, we refer to it as complementary switching. The right choice in the modulation technique has a significant impact on the drive features. Since the TPM/PWM module is able to generate any type of PWM modulation technique, the application designer can feel free to choose the right modulation for his application.

Also, any of the modulation techniques can be generated in both edge and center aligned PWM modes. There are two essential points of view on how to choose the right modulation technique. The first is the number of operational quadrants in the drive. If a full four-quadrant operation is required, the complementary switching has to be chosen. In the case of a two quadrant operation, the independent switching is sufficient. The second aspect, which has to be considered, is the correct back-EMF sensing. One of the requirements for correct back-EMF sensing is that the back-EMF voltage is sensed when both semiconductor switches are conducting.

Since the MC9S08AWx family is not equipped with PWM-to-ADC synchronization, another approach has to be found to ensure correct back-EMF sensing. The simple solution is to use a TPM/PWM overflow interrupt. In center aligned PWM mode the TPM/PWM overflow is just in the middle of the pulse. When the bipolar complementary switching is chosen, the duty cycle of the semiconductor switches can be always equal to or higher than 50 percent. This can be achieved by using a different commutation table for both clockwise and anticlockwise directions of rotation. Considering the center aligned bipolar complementary PWM, the correct back-EMF sensing is ensured at any operational point of the drive. Moreover, the full quadrant operation requirement is also satisfied.

The software structure consists of two periodic interrupts (ISR's), two event interrupts and a background loop (see Figure 3). The periodic ISR's are a TPM/PWM overflow ISR and a 5ms timer compare ISR. The first event ISR is driven by SCI communication, the second (IRQ ISR) handles the over current event. The time base of the control algorithm is derived from the TPM/PWM overflow ISR, which is called every 62.5 µs (16 kHz PWM). Each call of this ISR performs the ADC conversion, the PWM update and the zero crossing detection. There are two conversions performed during each PWM period.

Figure 3: Main control Interrupt Routines


TPM 1 Overflow Interrupt (62.5 µs)
Start first A/D Conversion
Second A/D Sample Pointer Calculation
Save First Sample
Start Second A/D Conversion
PWM Update
Save Second Sample
Zero Crossing Detection
END of Interrupt Service Routine
TPM 2 Ch1 Compare Interrupt (5 ms)
Speed/Torque Calculation
Speed/Torque PI Controller
Standstill Detection
Interface Handling (Up/down buttons, RUN/STOP switch)
END of Interrupt service Routine

The first conversion differs with each PWM period and is shared amongst the DC bus voltage, DC bus current and heat sink temperature. The second conversion always contains back-EMF voltage of the non-fed phase. The slower control loop is executed every 5 ms. The ISR executes speed and torque measurement/filtering, manual interface handling, torque and speed PI controller calculations, and standstill detection.

The background loop executes the application state machine (ASM), consisting of init, stop, alignment, start up, run and error states. After an MCU reset, the ASM goes through the init state to the stop state. As soon as the user turns the START/STOP switch to the START position the ASM goes into the alignment state, where the motor is aligned to a known position. After a predefined time the ASM continues to the start-up state. At the start-up state the six commutations are performed without any feedback. The timings among the commutations are defined by the start-up table. After the start-up state, the ASM goes into the run state, where zero-crossing is employed and control loop is closed. During the run state requests for new speeds and torques are accepted and maintained by the PI controllers. If a rotor standstill is detected the ASM goes again into the alignment state and tries to execute a new start up sequence. In case of any fault, the ASM goes into the error state and then into the stop state.

The whole software is written in C language, except for some mathematical functions, which are optimized in assembler. The code size is 4.5 KB, and the RAM used is less than 300 bytes. The current MCU load is less than 40 percent, which demonstrates the power of the MC9S08AWx family. A detailed description of the application is available as a design reference manual DRM086 on the Freescale web site (www.freescale.com/files/microcontrollers/doc/ref_manual/DRM086.pdf).

Pavel Grasblum has been working for Freescale for more than 7 years as a system application engineer. He received a Ph.D. in power electronics and electrical drives in 2001. Grasblum works with a wide range of Freescale products (8/16-bit MCUs and DSCs) supporting motor control and power conversion applications.

Ronald Gonzalez and Tatiana Orofino

Tips for Driving LCDs

Increasing display segments with dual MC9S08LC60 devices

Are you looking for a solution to drive liquid crystal displays (LCDs) with up to 320 segments? This article will provide you the path to using two Freescale 8-bit MC9S08LC60 (LC60) microcontrollers (MCUs), as well as instruction on making hardware connections that take full advantage of the LCD control and synchronization between the two MCUs.

Introduction

The LC60 is a member of the cost-effective, high-performance S08 family. This device has a rich set of peripherals and an LCD module which is able to drive displays with up to 160 segments. A 12-bit ADC offers high-precision analog to digital conversion. In addition, flash is configured in two independent arrays, allowing part of it to be used as EEPROM while the application runs. All these features are executed with very low power consumption, characteristic of the S08 architecture, making it attractive for battery-powered and handheld applications. This 8-bit MCU offers a high-integration level ideal for applications such as thermostats, calculators, digital multi-meters, medical monitoring devices, toys and appliances. The powerful peripherals and LCD module help reduce stress on the core. See the block diagram provided in Figure 1 to see how the additional hardware modules work.

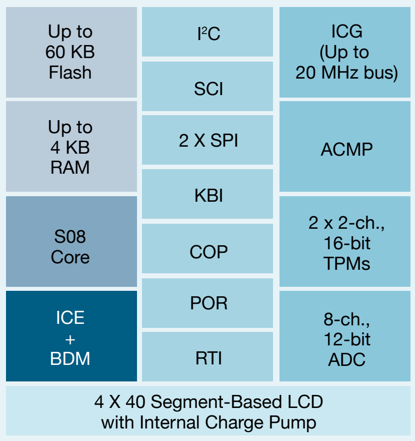
LCD Characteristics

The TN-type LCD glass compatible with MC9S08LC60 should follow the specifications below:

- LCD operating voltage—typical values for LCD glass operating voltage are 3 or 5V. The operating voltage is the nominal voltage level threshold to power on a segment.
- Current consumption—related to the total segment area
- View area—related to the size of the LCD glass
- Driving mode—duty cycle (1/2, 1/3 and 1/4) and bias (1/3)
- View angle—3 o’clock, 6 o’clock, 9 o’clock and 12 o’clock
- View mode—reflective, transmissive and transflective
- Operating temperature—varies

To have 160 segments driven by the LC60 MCU, 40 front planes and four backplanes are combined in a matrix structure with the front planes as rows and backplanes as columns. LC60 MCUs provide four voltage levels for front planes and backplanes, generating all the necessary waveforms for driving all segments. No external voltage regulator is required to drive these levels. One example can be viewed in Figure 2, which considers ½ Duty Cycle and ½ Bias ratios (three voltage levels). The data is loaded to the LCD display when registers LCDRAM [0..20] are loaded with new values. Turning a segment on or off is just a matter of setting or clearing one bit. Please read AN3280, “Interfacing an LCD to MC9S08LC60,” for detailed information.

Figure 1: MC9S08LC60 Block Diagram

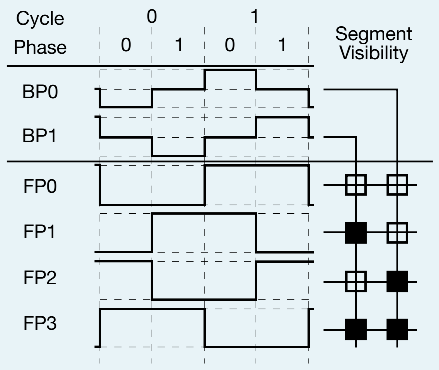


An LC60 MCU can drive LCD displays with up to 160 segments, therefore, two of them will be used to drive 320-segments LCDs. There are a few topics to be considered when connecting two MC9S08LC60 to a single LCD, such as:

- Backplane control
  Only one of the two LC60 MCUs should drive the four backplanes having their signals connected directly to the pins of the display.
- Front plane control
  Each LC60 MCU can control 40 front planes. Together they'll be able to drive 80. Therefore, they must exchange commands and data every time the display needs updating. The suggested interface is I²C, with one LC60 MCU as the Master adn one LC60 MCU as the Slave.
- Clock synchronization
  Each MCU will control half of the LCD front planes. However, only the Master will generate the backplane signals that will be connected to the LCD. For each segment to be turned on, a specific combination between backplanes and front planes is required. In that way, as front planes and backplanes are configured in a matrix structure, any changes to a front plane or backplane signal would require both Slave and Master to have updated synchronization. Therefore, microcontrollers must update LCDRAM and enable segments simultaneously, to have the same clock for synchronization.

Figure 3 shows a block diagram of the system using two MCUs and a 320-segment LCD display.

Figure 2: Waveforms for LCDs with 1/2 Duty and 1/2 Bias Ratios



LCD Initialization Module

The flowchart in Figure 4 represents a suggested LCD initialization routine from Master perspective. This ensures both LC60 MCUs enable the front planes they control simultaneously. After the I²C interface is already configured, the LCD controlled by the Master must be initialized but not enabled. It will be enabled only after the confirmation, via I²C, that the Slave has successfully received all transferred data and is ready to enable the LCD it controls. A keyboard interrupt (KBI) signal is sent to the Slave and both do the LCD enabling synchronously. This interrupt is referred to in the flowcharts as "initOK" command.

At the Slave perspective (see Figure 5), it should first initialize its LCD module, receive all data to be loaded in the LCD glass through I²C and store it in RAM and inform Master that transfer is finished. When initOK is detected (signal sent from Master), it enables the LCD.

Figure 3: System Block Diagram

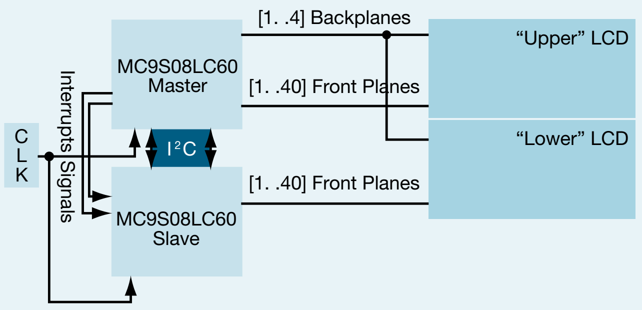


Figure 4 and 5: LCD Initialization Module

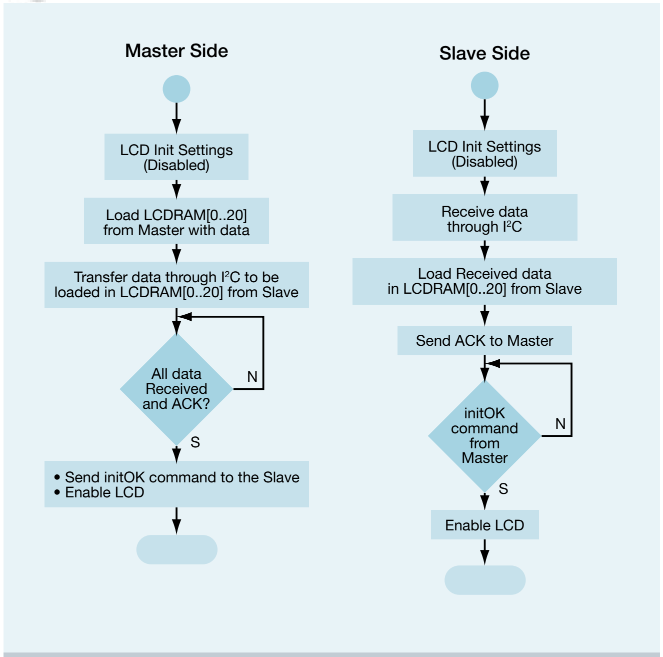
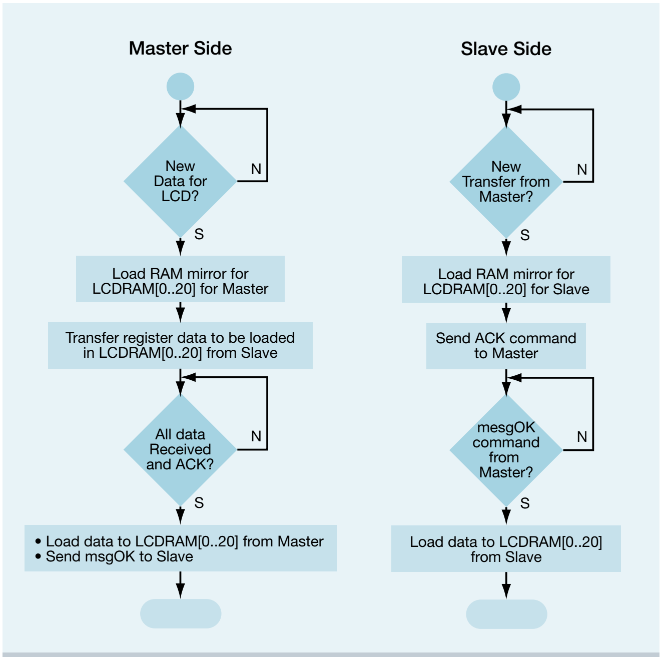

Master Side
LCD Init Settings (Disabled)
Load LCDRAM[0..20] from Master with data
Transfer data through I²C to be loaded in LCDRAM[0..20] from Slave
All data Received and ACK?
N
S
• Send initOK command to the Slave
• Enable LCD
Slave Side
LCD Init Settings (Disabled)
Receive data through I²C
Load Received data in LCDRAM[0..20] from Slave
Send ACK to Master
initOK command from Master
N
S
Enable LCD

Figure 6 and 7: LCD Data Update


Master Side
New Data for LCD?
N
S
Load RAM mirror for LCDRAM[0..20] for Master
Transfer register data to be loaded in LCDRAM[0..20] from Slave
All data Received and ACK?
N
S
• Load data to LCDRAM[0..20] from Master
• Send msgOK to Slave
Slave Side
New Transfer from Master?
N
S
Load RAM mirror for LCDRAM[0..20] for Slave
Send ACK command to Master
mesgOK command from Master?
N
S
Load data to LCDRAM[0..20] from Slave

LCD Data Update

When the display needs updating, a single writing to RAM registers wouldn't be simultaneous at the Master and Slave because Slave must wait for data transmission from Master before display updating. We suggest a mirror, which is a reserved space in Master and Slave RAMs, that receives and stores new LCD data for front planes controlled by Master and front planes controlled by Slave, respectively. These mirrors must have the same size as LCDRAM registers. For data updating in both LCD halves, it's necessary to have all data transferred to mirrors in both LC60 MCUs which stores the new content until they're synchronized.

After I²C signaling, content should be copied into LCDRAM registers at Master- and Slave- sides. Figures 6 and 7 show the flowcharts.

The Master MCU must transfer content to the "lower" LCD—the 40 front planes controlled by the Slave MCU—and wait until the transfer gets acknowledged.

Another keyboard interrupt signal, called mesgOK in this article, must be sent to the Slave. Only after it detects this signal will it start transferring it to its LCDRAM registers.

Summary

This advanced member of the S08 family, MC9S08LC60, comes to add value to Freescale's broad 8-bit microcontroller portfolio. It's cost-effective and easy to implement multiple MCUs to drive LCD displays through I²C.

References

All resources are available for electronic access at www.freescale.com.

Daniel Malik, "XGATE Library: TN/STN LCD Driver", Freescale Application Note, AN3219.

Steven Torres, "Interfacing a LCD to the MC9S08LC60", Freescale Application Note, AN3280.

Steven Torres, "Software to Accompany AN3219 Interfacing a LCD to the MC9S08LC60". Freescale Application Note, AN3280SW1.

Ronald Gonzalez worked as a developer at Philips for 11 years and has been a Freescale Latin America FAE since 1999, supporting automotive and home appliance customers in the region.

Tatiana Orofino is a computer engineer and has worked as a Freescale Latin America FAE since 2005, supporting customers globally as well as ZigBee® technology and ColdFire® products in the region.

Gonzalo Delgado Huitrón and Jaime Herrero Gallardo

SMAC
Wireless connectivity made easy

Overview

Embedded engineers know that the low-end wireless sensor network market is growing rapidly in different applications, such as controlling, monitoring, communication and data acquisition. We may think that having a wireless sensor network stack that satisfies network goals is enough—unfortunately, this isn't quite true. We are living in a competitive market where a few cents can be the difference in the customer selection. For that reason, Freescale has developed and is maintaining a freeware stack for Freescale products that could coexist with media access control (MAC) and ZigBee® protocols. The stack is a simple MAC or SMAC.

SMAC is an uncomplicated software protocol, based on the IEEE® 802.15.4 protocol, implemented to work with Freescale's transceivers with 8-bit microcontroller (MCU) control. It is intended to be used for fast product development and system evaluation. SMAC implements neither the full ZigBee stack nor the complete 802.15.4 layer, but it is a simple and easy to use protocol. Low-cost applications that require basic primitives, such as transmit, receive and power and channel selection are good examples of what SMAC can do.

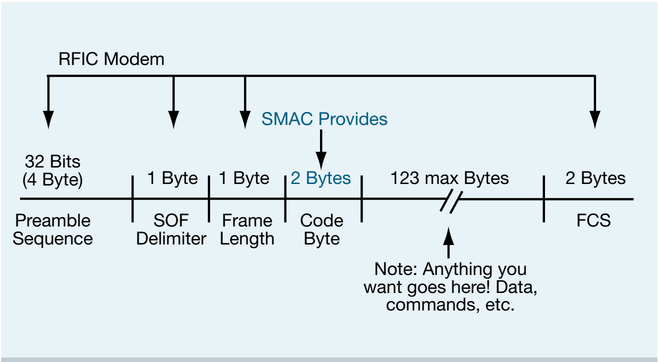
Features and Characteristics

SMAC is the lowest cost solution that can be used in Freescale transceivers and systems in package like MC1319x, MC1320x and MC1321x. It supports star and peer-to-peer networks, but more complex approaches can be developed, creating your own network layer or adding repeater nodes. The integration of the SMAC into the BeeKit software makes development time quick and accurate. It is written in full ANSI C targeted for the HCS08 core. It includes the source code and is distributed with the BeeKit software at no additional cost.

Portability and Small Footprint

SMAC is so small that it can fit into 4–8 KB. The size of the SMAC depends on all the extra elements that are added. Another great advantage of its small footprint is that you can use it with the CodeWarrior tool suite Special Edition. The

Figure 1: SMAC Packet Structure



Special Edition is available without cost and can compile up to 32 files of code. The code is written in C so it can be easily ported to other MCU families and technologies, such as 16- and 32-bit processors and digital signal controllers (DSCs).

Figure 2: SMAC in Numbers

Memory (Flash, RAM)		
	MIN	MAX
Only SMAC	2.52 KB, 6B	4.36 KB, 12B
Security Module	0.1 KB, 3B	0.15 KB, 3B
OTAP Module	2.3 KB, 318B	2.57 KB, 461B
Hardware (MCU PINS)		
	MIN	MAX
Transceiver Connection	5	7
RF Support	0	4

Software Architecture

Essentially, SMAC can be defined as a driver between the transceiver and the MCU. It also includes functions to initialize the MCU (MCUInit) and peripherals, such as serial communication interface (SCI), liquid crystal display (LCD), kernel debug interface (KDI) and light emitting diodes (LEDs), plus the security and the over-the-air-programming (OTAP) modules. The SMAC layer is the connection between the application and the SMAC core.

SMAC Primitives

Some of the most important primitives of the SMAC include:

Transmission

MCPSDataRequest is a blocking function used to transmit packets. You must provide a pointer to a previously created packet.

Reception

MLMERXEnableRequest and MLMERXDisableRequest are used for enabling/disabling the transceiver for a reception. A timeout parameter can be passed to this function to wait for a reception for a fixed time length.

MCPSDataIndication is a callback function executed each time a packet is successfully received or a timeout occurs. It is recommended that this is executed as quickly as possible since this function is executed inside an interruption.

Energy Measurement and Management

MLMEEnergyDetect and MLMELinkQuality can be used to measure the channel energy in decibels (dBm) before and after packet transmission. You can use these primitives to create a clear channel assessment (CCA) algorithm and a procedure to efficiently determine how many dBm are required to transmit to a near device.

MLMEMC13192FEGainAdjust is used to adjust the interpretation from the previous functions to make it more accurate. An accurate reading helps you determine if a device is “near” or “far.”

MLMEMC13192PAOutputAdjust sets the power used by the transceiver to transmit a packet. It can be changed at any moment if you find the device is in a measurable range.

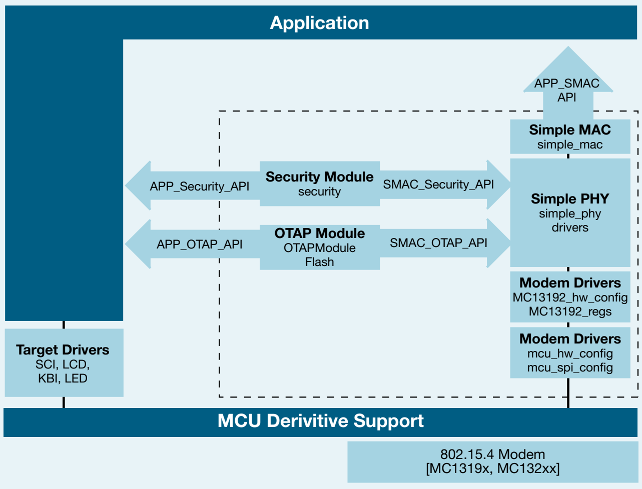
Low Power Consumption

MLMEHibernateRequest and MLMEDozeRequest are used to set the transceiver into a low power consumption state, with hibernate achieving the lowest power consumption.

MLMEWakeRequest is used to wake the transceiver from the low-power modes. Some modes require the MCU’s internal clock to wake up.

You can use these primitives to create a small and custom “network layer” that can manage several IDs and message types. You can even create your own CCA or an algorithm to select the best channel. There are many opportunities for additional enhancements, such as an integration of many functions (CCA-TX-ACK).

Figure 3: Software Architecture



Applications and Modules

Over the Air Programming (OTAP)

The OTAP functions as a wireless bootloader, allowing you to store firmware, select a device in a network and update its firmware with a simple click. This application can be improved to deliver more functionality, such as application backup or to download information (measured from sensors or other data stored in flash).

Repeater

If your application requires a longer range and you don’t want to use external components, you can use this “smart repeater” to increase that range. If this device detects the acknowledgement of a device in range, the packet will not be repeated. You could improve this repeater and create a simple router to manage your network.

Figure 4: Over the Air Programming Process

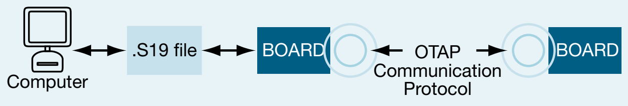
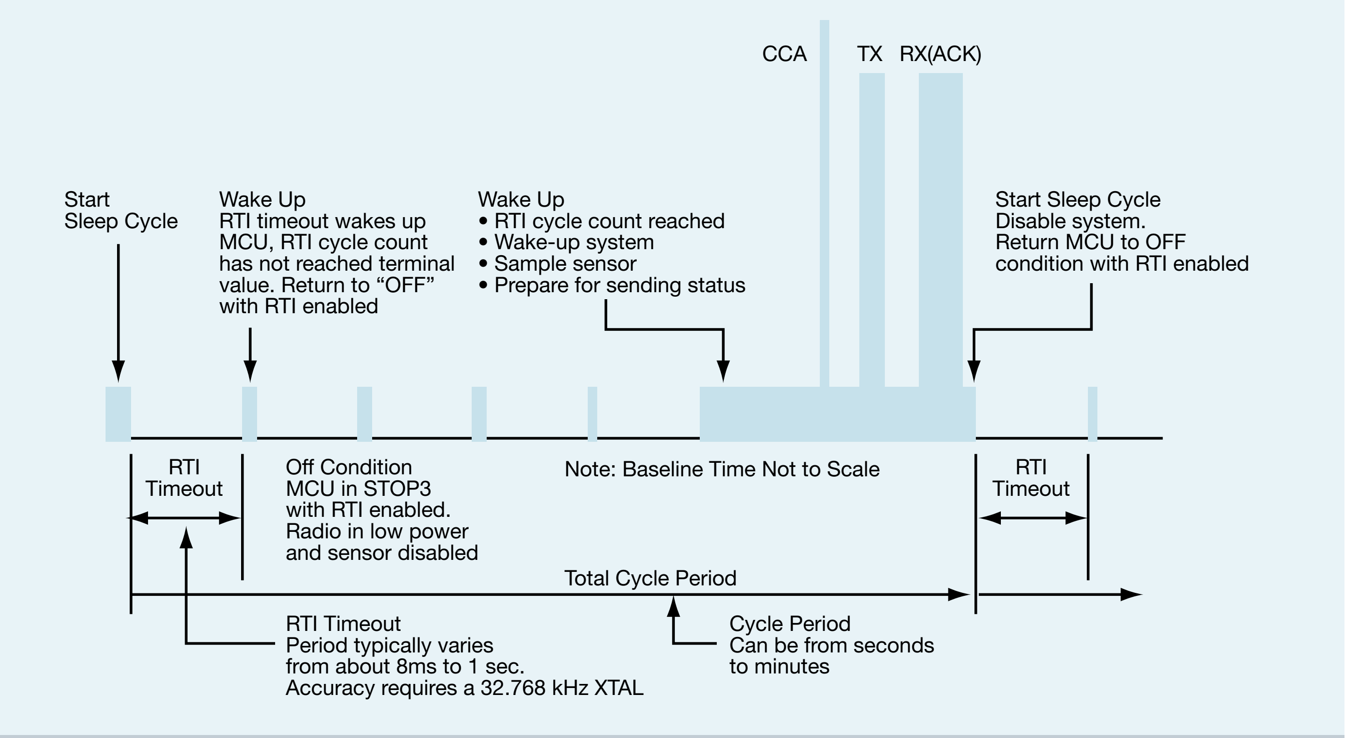


Figure 5: Typical Wireless Sensor Node Profile


CCA
TX
RX(ACK)
Start Sleep Cycle
Wake Up
RTI timeout wakes up MCU, RTI cycle count has not reached terminal value. Return to “OFF” with RTI enabled
Wake Up
• RTI cycle count reached
• Wake-up system
• Sample sensor
• Prepare for sending status
Start Sleep Cycle
Disable system.
Return MCU to OFF condition with RTI enabled
RTI Timeout
Off Condition
MCU in STOP3 with RTI enabled.
Radio in low power and sensor disabled
Note: Baseline Time Not to Scale
RTI Timeout
Total Cycle Period
RTI Timeout
Period typically varies from about 8ms to 1 sec.
Accuracy requires a 32.768 kHz XTAL
Cycle Period
Can be from seconds to minutes

Promiscuous Mode

As seen in Figure 1, SMAC uses 2 code bytes for its transmission and reception packets. Each time a packet is received, the SMAC discards all the packets that do not contain these bytes. SMAC has the ability to disable these code bytes and accept the packets from the ZigBee stack and the 802.15.4 MAC/PHY, making it more versatile.

Security

SMAC implements security with a predefined key and the XOR operation. This might not look like a tough algorithm, but since the source code is included you have the option to add your own security method.

Hints

Most of the time ZigBee and 802.15.4 devices are in a low power mode, which is also the best case for your SMAC application. In the MCU you can employ a low power mode, like Stop 3, and use the real-time interrupt (RTI) to wake up the device in a fixed amount of time. The transceiver can be in hibernation during sleep periods and wake up for the active periods. CCA can be achieved with an energy detect function to detect when to transmit or not. During the active period, the device can send its unique ID and specific data from a sensor read via the analog-to-digital converter (ADC). If this cycle is precisely developed the application’s battery could last a couple of years.

A world of low-cost applications can be implemented with SMAC. You are limited only by your creativity.

Gonzalo Delgado Huitrón is part of the Freescale RTAC team in Mexico. He is an applications engineer working for the Radio Product Division. His areas of expertise are ZigBee, radio frequency ID (RFID) and 8-bit MCUs.

Jaime Herrero Gallardo is an applications engineer for the Radio Product Division at Freescale. He is part of the SMAC developer team. His expertise is in real-time operating systems (RTOS), ColdFire embedded controllers and S08 MCUs.

Joaquin Romo

DDR Memories
Comparison and overview

We realize that one of the most important aspects of a computer is its capability to store large amounts of information in what we normally call “memory.” Specifically, it’s random access memory (RAM), and it holds volatile information that can be accessed quickly and directly. And considering the ever growing system need for speed and efficiency, understanding double-data-rate (DDR) memory is important to system developers.

With improvements in processor speeds, RAM memory has evolved into high performance RAM chipsets called DDR synchronous dynamic RAM (SDRAM). It doubles the processing rate by making a data fetch on both the rising and falling-edge of a clock cycle. This is in contrast to the older single-data-rate (SDR) SDRAM that makes a data fetch on only one edge of the clock cycle.

In addition to well-known computer applications, DDR memories are widely used in other high-speed, memory-demanding applications, such as graphic cards, which need to process a large amount of information in a very short time to achieve the best graphics processing efficiency. Blade servers using many blades, or single purpose boards, powered by a single, more efficient power supply also need fast memory access. This allows the blades to quickly transmit reliable information among each other and create greater opportunities to reduce power consumption. Memory devices are also required in networking and communications applications with tasks ranging from simple address lookups to traffic shaping/policing and buffer management.

This article describes the main characteristics of DDR memories as well as the specifications of power supplies required for these types.

DDR Memory Characteristics

DDR memory’s primary advantage is the ability to fetch data on both the rising and falling edge of a clock cycle, doubling the data rate for a given clock frequency. For example, in a DDR200 device the data transfer frequency is 200 MHz, but the bus speed is 100 MHz.

DDR1, DDR2 and DDR3 memories are powered up with 2.5, 1.8 and 1.5V supply voltages respectively, thus producing less heat and providing more efficiency in power management than normal SDRAM chipsets, which use 3.3V.

Temporization is another characteristic of DDR memories. Memory temporization is given through a series of numbers, such as 2-3-2-6-T1, 3-4-4-8 or 2-2-2-5 for DDR1. These numbers indicate the number of clock pulses that it takes the memory to perform a certain operation—the smaller the number, the faster the memory.

The operations that these numbers represent are the following: CL- tRCD – tRP – tRAS - CMD. To understand them, you have to keep in mind that the memory is internally organized as a matrix, where the data is stored at the intersection of the rows and columns.

- CL: Column address strobe (CAS) latency is the time it takes between the processor asking memory for data and memory returning it.
- tRCD: Row address strobe (RAS) to CAS delay is the time it takes between the activation of the row (RAS) and the column (CAS) where data is stored in the matrix.
- tRP: RAS precharge is the time between disabling the access to a row of data and the beginning of the access to another row of data.
- tRAS: Active to precharge delay is how long the memory has to wait until the next access to memory can be initiated.
- CMD: Command rate is the time between the memory chip activation and when the first command may be sent to the memory. Sometimes this value is not informed. It usually is T1 (1 clock speed) or T2 (2 clock speeds).

Table 1 is a comparison of clock and transfer rates for the RAM memory chipsets that can be found in today’s computers, including SDR, DDR, DDR2 and future DDR3 modules.

Table 1: SDRAM Memories Speed Comparison

Memory	Technology	Rated Clock	Real Clock	Maximum Transfer Rate
PC66	SDRAM	66 MHz	66 MHz	533 MB/s
PC100	SDRAM	100 MHz	100 MHz	800 MB/s
PC133	SDRAM	133 MHz	133 MHz	1,066 MB/s
DDR200	DDR-SDRAM	200 MHz	100 MHz	1,600 MB/s
DDR266	DDR-SDRAM	266 MHz	133 MHz	2,100 MB/s
DDR333	DDR-SDRAM	333 MHz	166 MHz	2,700 MB/s
DDR400	DDR-SDRAM	400 MHz	200 MHz	3,200 MB/s
DDR2-400	DDR2-SDRAM	400 MHz	200 MHz	3,200 MB/s
DDR2-533	DDR2-SDRAM	533 MHz	266 MHz	4,264 MB/s
DDR2-667	DDR2-SDRAM	667 MHz	333 MHz	5,336 MB/s
DDR2-800	DDR2-SDRAM	800 MHz	400 MHz	6,400 MB/s
DDR3-800	DDR3-SDRAM	800 MHz	400 MHz	6,400 MB/s
DDR3-1066	DDR3-SDRAM	1066 MHz	533 MHz	8,528 MB/s
DDR3-1333	DDR3-SDRAM	1333 MHz	666 MHz	10,664 MB/s
DDR3-1600	DDR3-SRAM	1600 MHz	800 MHz	12,800 MB/s

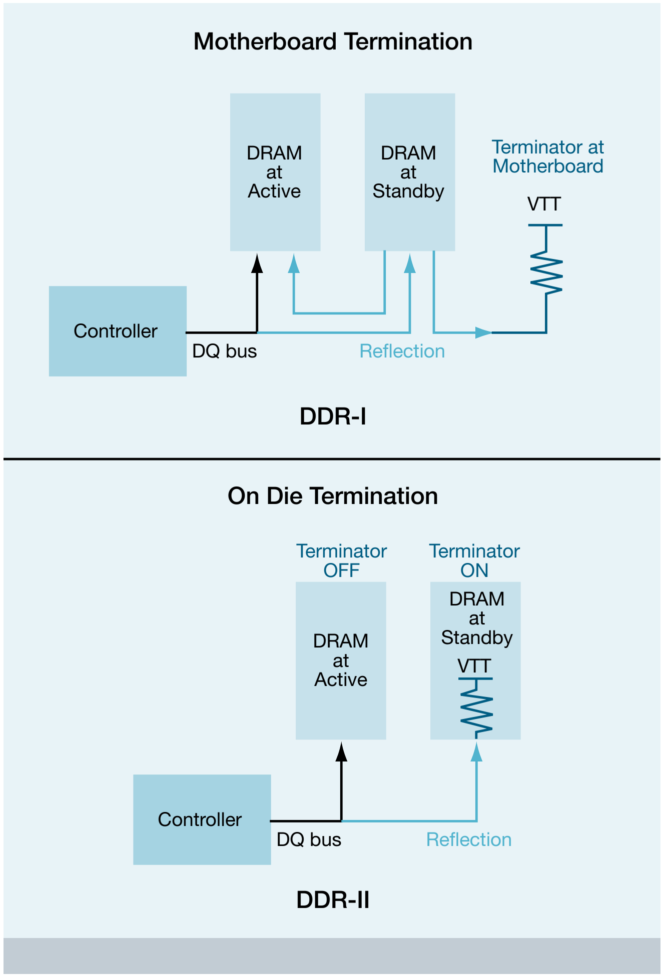
Types of DDR Memories

There are presently three generations of DDR memories:

1. DDR1 memory, with a maximum rated clock of 400 MHz and a 64-bit (8 bytes) data bus is now becoming obsolete and is not being produced in massive quantities. Technology is adopting new ways to achieve faster speeds/data rates for RAM memories.
2. DDR2 technology is replacing DDR with data rates from 400 MHz to 800 MHz and a data bus of 64 bits (8 bytes). Widely produced by RAM manufacturers, DDR2 memory is physically incompatible with the previous generation of DDR memories.
3. DDR3 technology picks up where DDR2 left off (800 Mbps bandwidth) and brings the speed up to 1.6 Gbps. One of the chips already announced by ELPIDA contains up to 512 megabits of DDR3 SDRAM, with a column access time of 8.75 ns (CL7 latency) and data transfer rate of 1.6 Gbps at 1.6 GHz. The 1.5V DDR3 voltage level also saves some power compared to DDR2 memory. What is more interesting is that at an even lower 1.36V, the DDR3 RAM runs fine at 1.333 GHz (DDR3-1333) with a CL6 latency (8.4 ns total CAS time), which matches the CAS time of the fastest current DDR2 memory.

Figure 2.0.1 shows the on die termination (ODT) for DDR2/DDR3 memory types compared to the motherboard termination of DDR1, which is the primary reason why the first two types are physically incompatible with DDR1 devices.

Figure 2.0.1: Termination Differences between DDR-I and DDR-II



DDR2 versus DDR3

The primary differences between the DDR2 and DDR3 modules are:

- DDR2 memories include 400 MHz, 533 MHz, 667 MHz and 800 MHz versions, while DDR3 memories include 800 MHz, 1066 MHz, 1333 MHz and 1600 MHz versions. Both types double the data rate for a given clock frequency. Therefore, the listed clocks are nominal clocks, not real ones. To get the real clock divide the nominal clock by two. For example, DDR2-667 memory in fact works at 333 MHz.
- Besides the enhanced bandwidth, DDR3 also uses less power than DDR2 by operating on 1.5V—a 16.3 percent reduction compared to DDR2 (1.8V). Both DDR2 and DDR3 memories have power saving features, such as smaller page sizes and an active power down mode. These power consumption advantages make DDR3 memory especially suitable for notebook computers, servers and low power mobile applications.
- A newly introduced automatic calibration feature for the output data buffer enhances the ability to control the system timing budget during variations in voltage and temperature. This feature helps enable robust, high-performance operation, one of the key benefits of the DDR3 architecture.
- DDR3 devices introduce an interrupt reset for system flexibility
- In DDR2 memories, the CL parameter, which is the time the memory delays delivering requested data, can be three to five clock cycles, while on DDR3 memories CL can be of five to ten clock cycles.
- In DDR2 memories, depending on the chip, there is an additional latency (AL) of zero to five clock cycles. So in a DDR2 memory with CL4 the AL1 latency is five.
- DDR2 memories have a write latency equal to the read latency (CL + AL) minus one.
- Internally, the controller in DDR2 memories works by preloading 4 data bits from the storage area (a task known as prefetch) while the controller inside DDR3 memories works by loading 8 bits in advance.

Typical DDR Power Requirements

VTT and VREF Considerations for DDRx SDRAM Devices

The termination voltage (VTT) supply must sink and source current at one-half output voltage (½ VDDQ). This means that a standard switching power supply cannot be used without a shunt to allow for the supply to sink current. Since each data line is connected to VTT with relatively low impedance, this supply must be extremely stable. Any noise on this supply goes directly to the data lines.

Sufficient bulk and bypass capacitance must be provided to keep this supply at ½ VDDQ. This same noise issue requires that the voltage reference (VREF) signal cannot be derived from VTT, but instead must be derived from VDDQ with a one percent or better resistor divider. Do not try to generate VREF with one divider and route it from the controller to the memory devices. Generate a local VREF at every device. Use discrete resistors to generate VREF. Do not use resistor packs.

DDR Memories Power Management

DDR1 memory has a push-pull output buffer, while the input receiver is a differential stage requiring a reference bias midpoint, VREF. Therefore, it requires an input voltage termination capable of sourcing as well as sinking current. This last feature differentiates the DDR VTT from other terminations present on the computer motherboard. The termination for the front system bus (FSB), connecting the CPU to the memory channel hub (MCH), requires only sink capability due to its termination to the positive rail. Hence, such DDR VTT termination can't re-use or adapt previous VTT termination architectures and requires a new design. Figure 3.2.1 shows the typical power management configuration for a DDR1 memory.

Between any output buffer from the driving chipset and the corresponding input receiver on the memory module, you must terminate a routing trace or stub with resistors RT and RS (Figure 3.2.1). Accounting for all the impedances, including the output buffers, each terminated line can sink or source ±16.2 mA (this is more than the older value of ±15.2 mA, per the June 2000 Revision of JESD79). For systems with longer trace lengths, between transmitter and receiver, it may be necessary to terminate the line at both ends, which doubles the current.

Peak and average current consumption for VTT and VDDQ are two parameters for the correct sizing of our power supply system. To find the peak power requirements for the termination voltage, we must determine the total lines in the memory system.

Figure 3.2.1: DDR Power Management

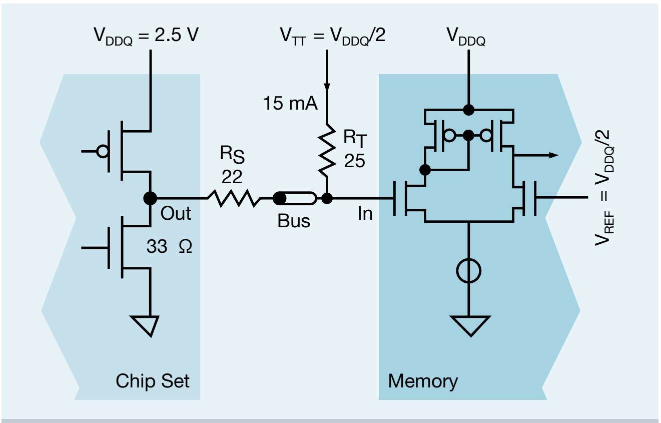


Table 2: SDRAM Devices Comparison

Items	DDR3 SDRAM	DDR2 SDRAM	DDR SDRAM
Clock frequency	400/533/667/ 800 MHz	200/266/333/400 MHz	100/133/166/200 MHz
Transfer data rate	800/1066/1333/ 1600 Mbps	400/533/667/800 Mbps	200/266/333/400 Mbps
I/O width	x4/x8/x16	x4/x8/x16	x4/x8/x16/x32
Prefetch bit width	8-bit	4-bit	2-bit
Clock input	Differential clock	Differential clock	Differential clock
Burst length	8, 4 (Burst chop)	4, 8	2, 4, 8
Data strobe	Differential data strobe	Differential data strobe	Single data strobe
Supply voltage	1.5V	1.8V	2.5V
Interface	SSTL_15	SSTL_18	SSTL_2
/CAS latency (CL)	5, 6, 7, 8, 9, 10 clock	3, 4, 5 clock	2, 2.5, 3 clock
On die termination (ODT)	Supported	Supported	Unsupported
Component package	FBGA	FBGA	TSOP(II) / FBGA / LQFP

Power Sequencing Violations

DDR memories must be powered up in a specific manner. Any violation of the power up sequencing will result in undefined operations. Also, power down sequencing serves as a power saving tool when the DDR device needs to be shut down without all other devices connected to the same power supply. Violating these sequences will present poor power saving results.

Typical Applications of DDRx Memories

Market analyses indicate that DDR is currently utilized in over 50 percent of all electronic systems, and usage is expected to increase to 80 percent over the next several years. DDR is not, and will never be, an “all things to all designs” technology. DDR memory is well suited for those designs that have a high read to write ratio. Quad-data-rate memory, for example, is designed for applications that require a 50 percent read/write ratio.

Today, DDR memories are used mainly for computer applications in dual in-line memory module (DIMM) RAM devices. DDR memories can be used when interfacing with 32-bit microcontrollers and DSPs. Since many memories work with a 64-bit data bus, microcontrollers have to make a double data acquisition to get the less significant 32 bits first and then the more significant 32 bits. DDR memories have also been used to interface with FPGAs, giving those devices wide programming flexibility. FPGAs are often used to customize applications which combine many digital modules, such as a microcontroller with specific application hardware, USB controllers and printing modules. It is also common to find FPGAs used specifically as memory controllers to interface with other devices. DDR memories are suitable for these types of devices because they are fast and inexpensive compared with other RAM memories.

Conclusions

To remain competitive, you have to create new designs that take advantage of today’s advanced technology. To understand the best solution for a specific design challenge, it is important to research well, giving you the opportunity to weigh the advantages of technologies, such as DDR memory, to incorporate them in new product developments. This article gives you a brief and useful overview of DDR memories, describing how they are different from other memories to give you a good starting point in choosing the best solution for your specific needs.

References

- JEDEC STANDARD JESD79, June 2000 and JESD8-9 of Sep.1998.
- Understanding Ram Timings (Article) www.hardwaresecrets.com/article/26/1
- DDR Memories require efficient power management. (Article) powerelectronics.com/mag/power_ddr_memories_require/
- ELPIDA 1 Gb DDR2 SDRAM memory datasheet

Joaquin Romo is an application engineer working in the Power Management and Motor Control Applications organization. Romo is in charge of characterizing new power management products and developing application notes and evaluation boards focused on the consumer electronics market.

Pavel Grasblum

Using the HCS08 TPM Module
In Motor Control Applications

Designers can choose from a wide range of microcontrollers to provide digital control for variable speed drives. Microcontrollers based on Freescale's HCS08 core are particularly well suited for low end motor control applications.

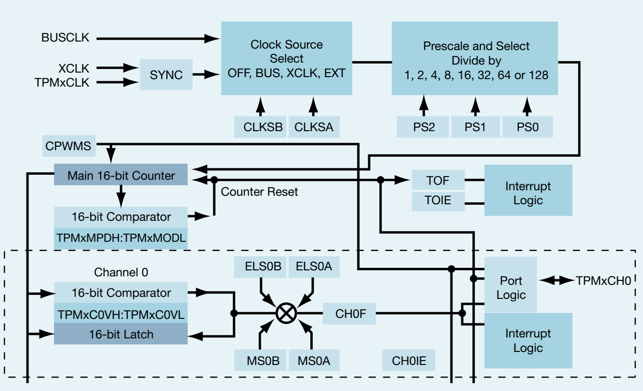
Devices based on this high-performance HCS08 core support 1.8–5V applications and come packed with valuable features, such as up to 60 KB flash memory, up to 2 KB RAM, a flexible internal clock generator that eliminates the need for external components, low voltage detection, an analogue-to-digital converter (ADC), timer (TIM) module, serial communication modules and many more. This article deals with the implementation of various pulse width modulation techniques using a TIM module and how the chosen PWM modulation technique impacts motor control application features.

The PWM modulation techniques can be distinguished according to various parameters, such as edge/centre aligned PWM, unipolar/bipolar PWM, complementary/independent PWM, the number of PWM channels, etc.

The timer/PWM module (TPM module) comes from the well known TIM module for the HC08 family. The basic structure of the TPM module can be seen in Figure 1. The main features are:

- Up to six channels per module
  - Each channel may be input capture, output compare, or buffered edge-aligned PWM
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
  - Selectable polarity on PWM outputs
- Each TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on all channels
- Clock source to the prescaler for each TPM is independently selectable as bus clock, fixed system clock, or an external pin
  - Prescale taps for divide by 1, 2, 4, 8, 16, 32, 64, or 128
  - External clock inputs
- 16-bit free-running or up/down (CPWM) count operation
- 16-bit modulus register to control counter range
- Timer system enabled
- One interrupt per channel plus a terminal count interrupt for each TPM module

Figure 1: Basic Structure of TPM Module



The advanced features of center-aligned PWM modulation extends the field of applications to motor control in small appliances.

Another thing to be considered in PWM generation is the current or back-EMF voltage sensing. A low-cost motor control application usually uses a shunt resistor for current sensing. In this case, the current has to be sensed in the middle of the PWM pulse. This requires the synchronization of the TPM and ADC modules. Since no member of the HCS08 family is equipped with TPM to ADC synchronization, how to synchronize both modules has to be considered at PWM generation. The situation is similar with sensorless brushless DC (BLDC) motor control applications, where Back-EMF voltage has to be sampled in the middle of a PWM pulse. Other PWM generation connected motor control application features will be discussed in detail further on.

Edge/Center-aligned PWM Modulation

This parameter defines how the pulses channels from different TPM channels are synchronized. With edge-aligned PWM modulation all the pulses start at the beginning of the PWM period. With center aligned PWM modulation, all the pulses are centered on the middle of the pulse. The advantage of center-aligned PWM can be seen in three-phase sinusoidal PWM modulation. If the duty cycles in all the phases are different, the transistor switching is spread over the entire PWM period. Since each transistor switching generates electromagnetic noise, a lower overall noise level can be achieved. If all pulses are synchronized in the middle of the pulse, they can be used for TPM to ADC synchronization. The polarity of the output signal can be set in order to have the TPM Overflow interrupt in the middle of the pulse. This interrupt can be used to start the ADC conversion. The edge/center-aligned mode can be set by the CPWMS bit in the TPMxSC register.

Unipolar/Bipolar and Independent/Complementary PWM Modulation

The unipolar/bipolar PWM defines how the voltage is applied to the motor. If the motor sees either the DC bus or zero voltage during one PWM cycle, we refer to this modulation technique as unipolar. If the motor sees both polarities of the DC bus voltage during one PWM cycle, the technique is called bipolar. Unipolar PWM is also called “soft switching,” while bipolar PWM is called “hard switching.” A second differentiator is the way of switching opposing semiconductor switches. If they are switched independently, we call this independent switching. If they are switched in a complementary manner, we call this complementary switching. Both PWM features can be combined with each other, so we get four possible combinations: unipolar independent, unipolar complementary, bipolar independent and bipolar complementary PWM modulation. The right choice in the modulation technique has a significant impact on the drive features. The features of each modulation technique are illustrated in a DC motor connected into an H-bridge, which consist of four transistors, Q1-Q4, with anti-parallel diodes (see Figure 2). The same topology can be considered for a six step control of a BLDC motor, since at any time only two phases are powered.

Figure 2: H-bridge with DC Motor

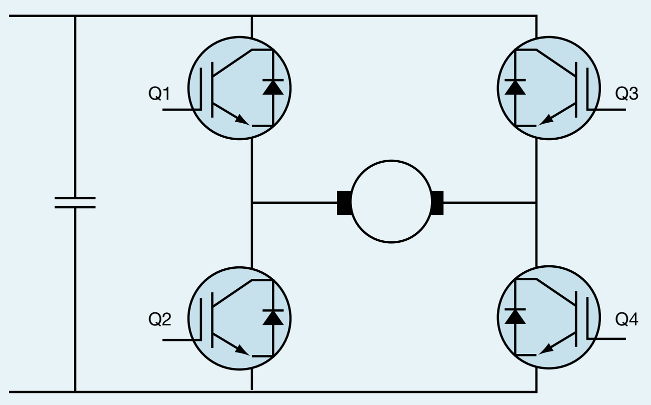


Figure 3: Independent Bipolar PWM Modulation—Phase 1

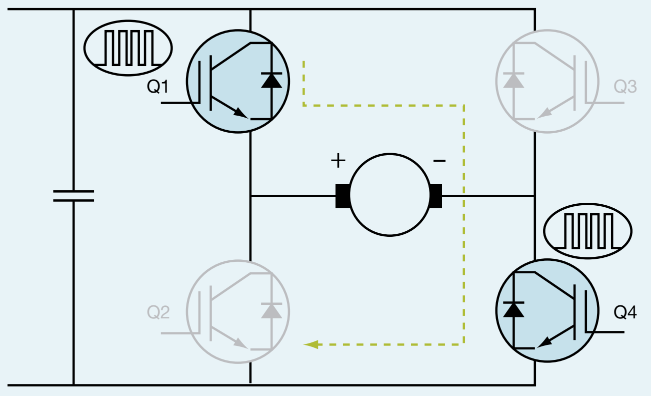


Figure 4: Independent Bipolar PWM Modulation—Phase 2

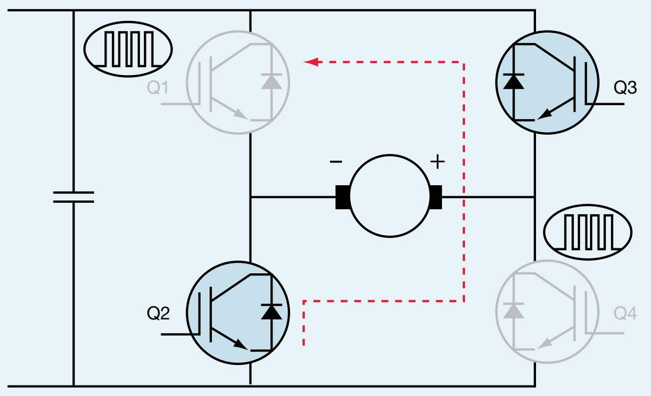


Figure 5: Independent Bipolar PWM Modulation—Detail of PWM Period (center-aligned)

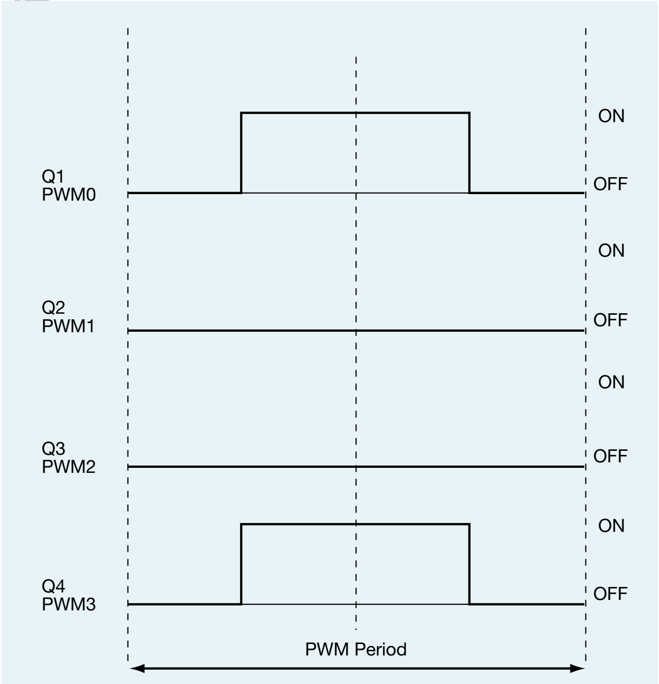


Figure 6: Independent Unipolar PWM Modulation—Phase 1

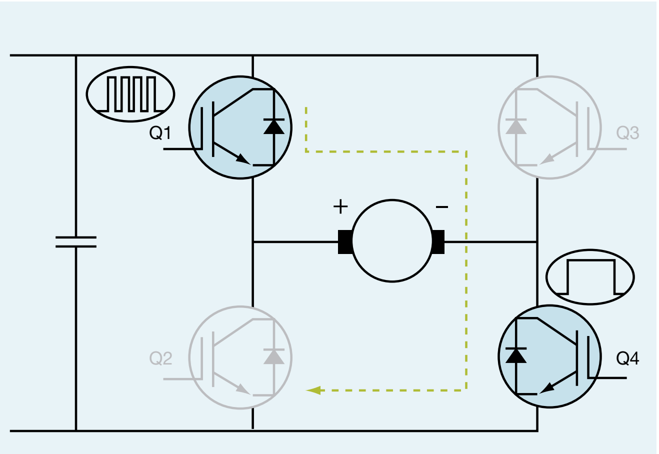
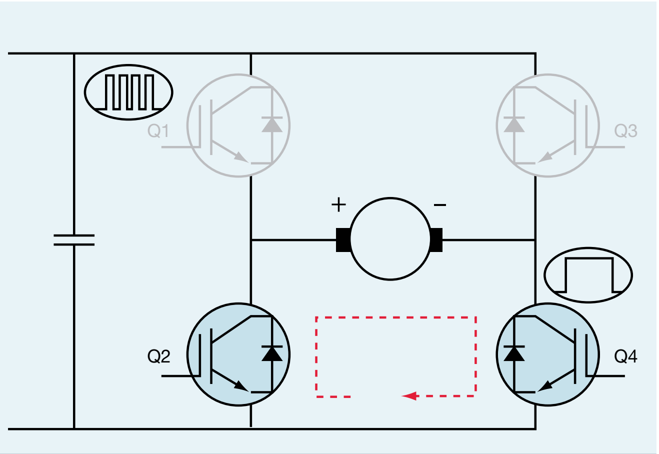


Figure 7: Independent Unipolar PWM Modulation—Phase 2



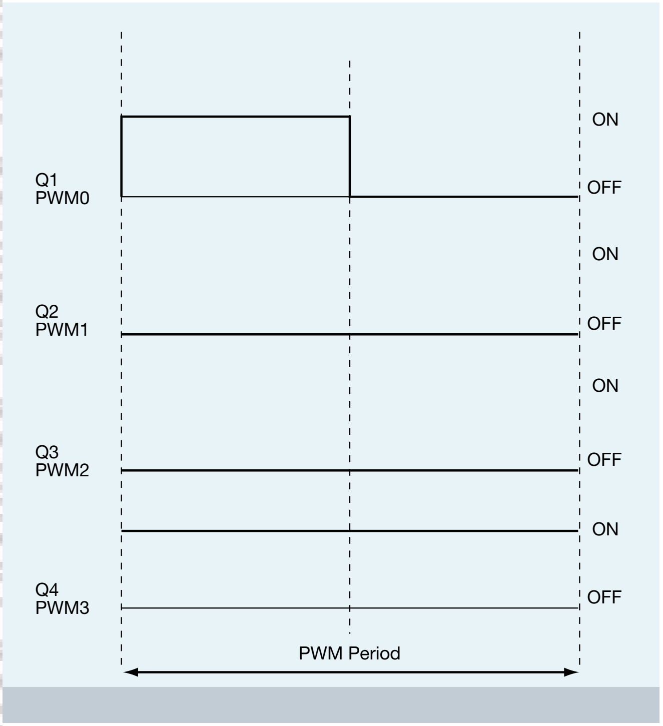
Independent Bipolar PWM Modulation

The principle of independent bipolar PWM modulation can be seen in Figures 3 and 4. The PWM period can be divided into three phases. In the first phase, the transistors Q1 and Q4 are driven by the same signal. When Q1 and Q4 are switched on, the DC motor is connected to the positive DC bus voltage and the current through the DC motor rises. In the second phase, transistors Q1 and Q4 are switched off and current starts to fall. The current flows through freewheeling diodes of Q2 and Q3, and the DC motor is connected to the negative DC bus voltage. Now independent—the transistors in the phase (Q1, Q2 and Q3, Q4) are switched independently and bipolar—the DC motor sees both polarities of the DC bus voltage. If the DC motor current falls to zero before the end of the PWM period, there is a third phase. The current through the DC motor is zero and the DC motor is disconnected from the power supply. The detail of the one PWM period can be seen in Figure 5. The transistors Q2 and Q3 are switched off for the entire PWM period. To run the DC motor in the opposite direction, transistors Q2 and Q3 are driven by a control signal, and Q1 and Q4 are switched off.

This PWM modulation can be generated either as an edge-aligned or center-aligned PWM. There is no advantage in reducing the electromagnetic noise since both transistors are switched in the same time. Alignment can be chosen according to the current sensing method.

This modulation is easy to implement using the TPM module. Two channels are set to generate the same signal. If by chance the circumstances are right, a single timer channel can be used to generate the signal for two transistors (Q1+Q4 and Q2+Q3). On the other hand there are some drawbacks, as the transition from discontinuous to continuous current mode can dramatically change the drive's behavior. With discontinuous current the DC motor exhibits low torque, while in continuous current the torque dramatically increases. The drive can work in two quadrant operation and cannot actively brake. From a current sensing point of view the duty cycle changes from 0 to 100 percent. This means that some minimal pulse width is necessary to measure the current.

Figure 8: Independent Unipolar PWM Modulation—Detail of PWM period (edge-aligned)



Independent Unipolar PWM Modulation

Figures 6 and 7 show the differences between this modulation and the previous one. In the first phase, transistor Q1 is driven by a control signal, and transistor Q4 is switched on over the entire period. The DC motor is connected to the positive DC bus voltage and the current increases. The first phase of the PWM period is the same as in the independent bipolar PWM modulation. In the second phase, transistor Q1 is switched off while transistor Q4 is still switched on. The current still flows through the DC motor but it is closed through the freewheeling diode of Q2. During this second phase, the motor sees zero voltage. The DC motor current starts to fall, but, due to zero voltage on the motor the negative slope is very slow. This means that the current goes into continuous mode with a very narrow pulse, leading to a much better behavior in the drive. The DC motor exhibits high torque from a very small duty cycle, and so tuning the torque and speed controllers is much easier. The drive can work as a motor and not as a brake, in two quadrant operation only. Also, the current ripple is smaller, which means smaller losses in the semiconductor devices.

Figure 9: Complementary Bipolar PWM Modulation—Phase 1 Positive Current

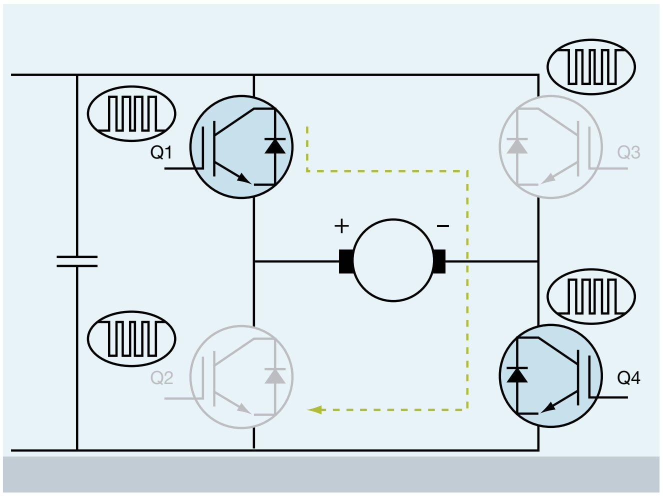
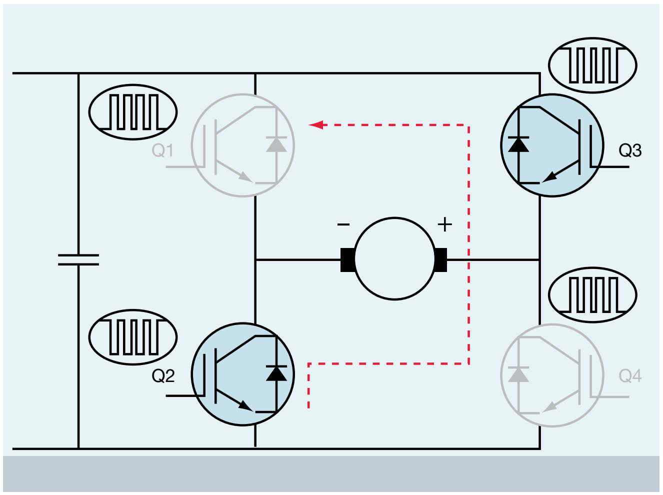


Figure 10: Complementary Bipolar PWM Modulation—Phase 2 Positive Current



Implementation using a TPM module is also simple. The PWM signal is generated by two TPM channels, usually for the top transistors (Q1 and Q3). The bottom transistors can be driven either by other TPM channels or by GPIO pins, since they remain switched on for the entire PWM period. With regards to current sensing, the situation is the same as in the previous case. The duty cycle varies from 0 to 100 percent so some minimal pulse width is necessary for current measurement.

Complementary Bipolar PWM Modulation

Unlike the two previous modulations, all transistors are switched in a complementary manner. Details can be seen in Figures 9 and 10. The transistors Q1 and Q4 are driven by the same signal, Q2 and Q3 are switched off. In the first phase the DC motor is connected to the positive DC bus voltage and the DC motor current increases. In the second phase the transistors Q2 and Q3 conduct current, and Q1 and Q4 are switched off. The DC motor is connected to the negative DC bus voltage and the DC motor current falls through the freewheeling diodes of Q2 and Q3. If the current reaches zero, it can continue to flow in a negative direction. When the current is negative, the drive works as an active brake and works in a four quadrant operation. The current flows in continuous mode under any condition, so the drive has a stable behavior.

Complementary switching requires dead-time generation to avoid cross conduction. Since the TPM module is not equipped by hardware for dead-time generation, dead-time has to be generated by software.

Since the TPM module does not support complementary signals by hardware, setting the TPM module is more complex. The correct settings for all channels can be seen below:

```
PWM0:  (duty_cycle + dead_time),
       negative polarity
       (TMPxCnSC:ELSnB =x,TMPxCnSC:ELSnA =1)
PWM1:  (duty_cycle – dead_time),
       positive polarity
       (TMPxCnSC:ELSnB =1,TMPxCnSC:ELSnA =0)
PWM2:  (duty_cycle – dead_time),
       positive polarity
       (TMPxCnSC:ELSnB =1,TMPxCnSC:ELSnA =0)
PWM3:  (duty_cycle + dead_time),
       negative polarity
       (TMPxCnSC:ELSnB =x,TMPxCnSC:ELSnA =1)
```

The `duty_cycle` is changed from 0 to 100 percent. On the other hand, current measurement is much simpler. Since current flows through the transistors at any time, it can be measured in the first or second half of a PWM period depending on the actual duty cycle. The detail of a PWM period is shown in Figure 11.

Figure 11: Complementary Bipolar PWM Modulation—Detail of PWM Period

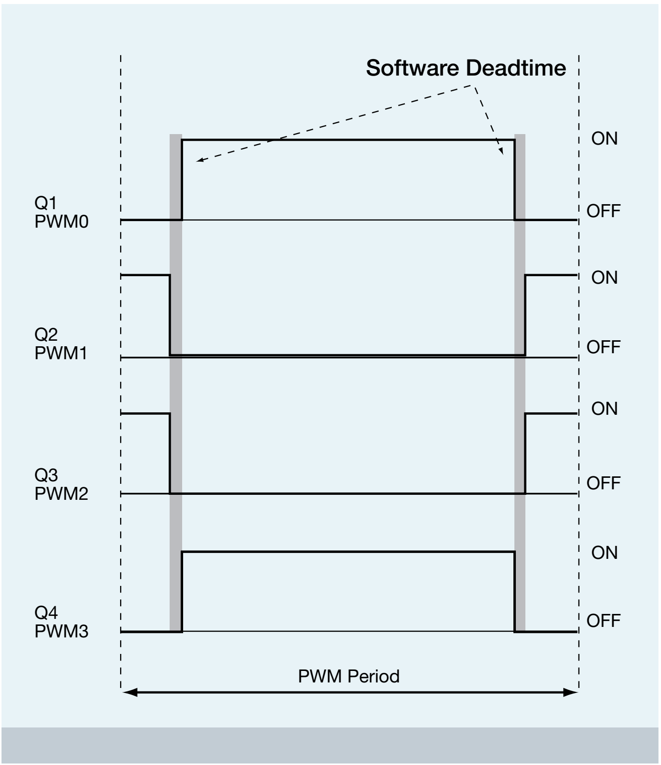


Figure 12: Complementary Unipolar PWM Modulation—Detail of PWM Period Phase 1

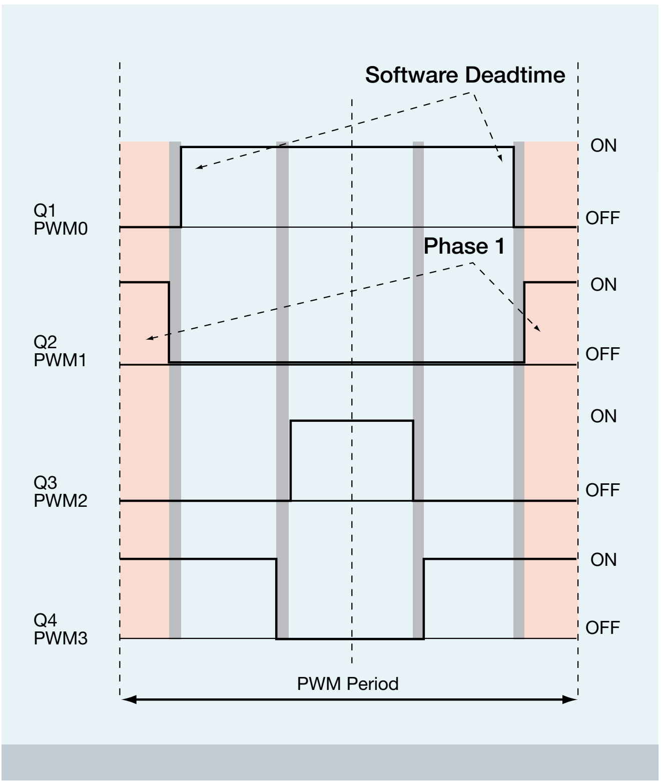
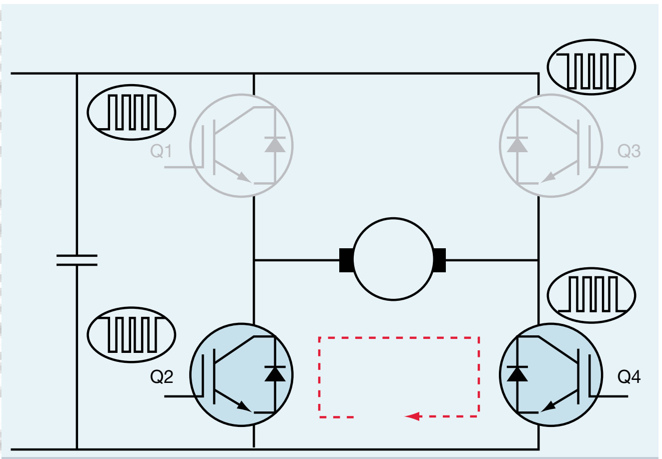


Figure 13: Complementary Unipolar PWM Modulation—Phase 1



Complementary Unipolar PWM Modulation

In this PWM modulation technique, all the transistors are also switched in a complementary manner. The PWM period of the complementary unipolar PWM modulation can be divided into three phases. The first phase can be seen in Figures 12 and 13. Both the bottom transistors are switched on, a zero voltage is applied to the DC motor and current flows through Q4 and the freewheeling diode of Q2.

The second phase is shown in Figures 14 and 15. The transistors Q1 and Q4 conduct current, the DC motor is connected to the positive DC bus voltage and the current flows through the DC motor.

The third phase is similar to the first one, but both upper transistors Q1 and Q3 are switched on. On the motor, zero voltage is applied and the DC motor current is closed through Q1 and the freewheeling diode of Q3. Phase 3 is depicted in Figure 16.

This type of the PWM modulation offers the lowest current ripple since the motor is connected twice to the power supply. It allows using a lower switching frequency to get the same current ripple in comparison to the previous PWM modulation technique. The drive features are also equal to the previous type of PWM modulation.

The implementation of this modulation is again quite complex, since the complementary signals and the dead-time have to be generated by software. Configuration of the TPM channels is shown in the next paragraph:

```
PWM0: MODULO/2 +/- duty_cycle + dead time
PWM1: MODULO/2 +/- duty_cycle - dead time
PWM2: MODULO/2 -/+ duty_cycle + dead time
PWM3: MODULO/2 -/+ duty_cycle - dead time
```

Figure 14: Complementary Unipolar PWM Modulation—Detail of PWM Period Phase 2

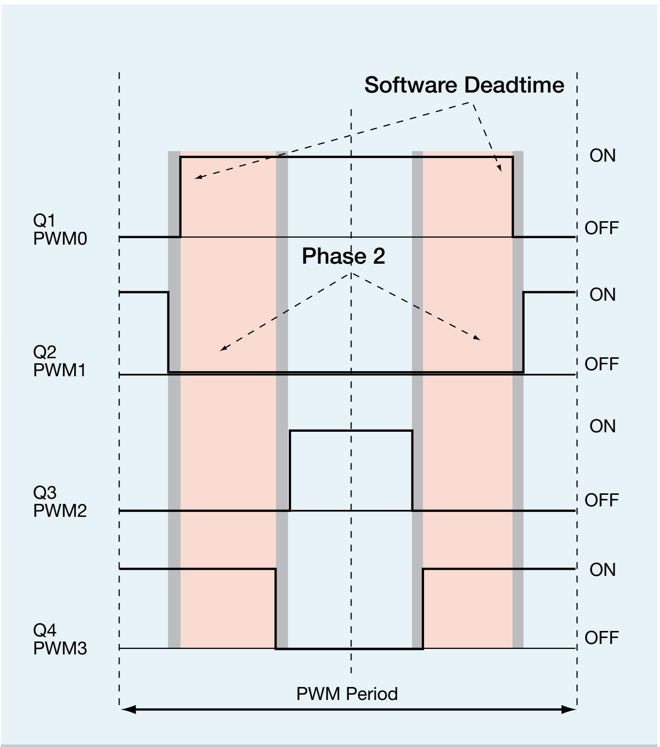
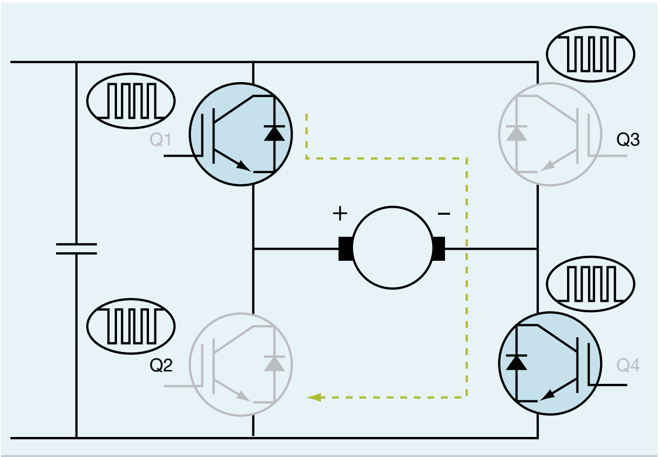


Figure 15: Complementary Unipolar PWM Modulation—Phase 2



The `duty_cycle` is changed from 0 to 50 percent. All the top channels are set to negative polarity (TMPxCnSC:ELSnB =x,TMPxCnSC:ELSnA =1) and all bottom channels are set to positive polarity (TMPxCnSC:ELSnB =1,TMPxCnSC:ELSnA =0). The notation of TPM channel configuration means that if the variable `duty_cycle` is added to channels PWM0 and 1, the `duty_cycle` is subtracted from PWM2 and PWM3, and vice versa.

Sinusoidal PWM Modulation

The sinusoidal PWM modulation requires six TPM channels when considering three phase motors. This modulation uses center-aligned PWM, due to the reduction of electromagnetic noise. All three phases work in a complementary manner and the `duty_cycle` is modulated by a sinusoidal table every PWM period, shifted by 120 electrical degrees in every phase. The configuration of the TPM module is similar to the previous case. The `duty_cycle` is changed from 0 to 100 percent. All the top channels are set to negative polarity (TMPxCnSC:ELSnB =x,TMPxCnSC:ELSnA =1) and all bottom channels are set to positive polarity (TMPxCnSC:ELSnB =1,TMPxCnSC:ELSnA =0).

Based on the drive type (DC motor—one direction of rotation, DC motor—both directions of rotation, BLDC motor, AC induction motor, PM synchronous motor, etc.), the appropriate microcontroller has to be selected. Table 1 shows the number of necessary TPM channels dependent on the type of motor and the operational mode. Table 2 summarizes the available members of the HCS08 family, including the number TPM modules and channels.

The practical implementation of PWM modulation using a TPM module can be found in the application notes or designer reference manuals (e.g. complementary bipolar modulation for a BLDC motor in DRM086) on the Freescale web pages.

Figure 16: Complementary Unipolar PWM Modulation—Detail of PWM Period Phase 3

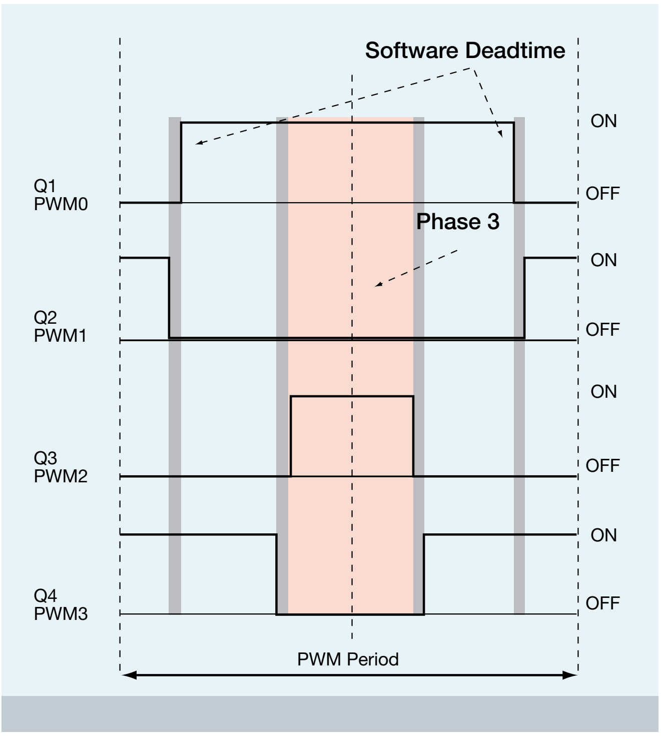


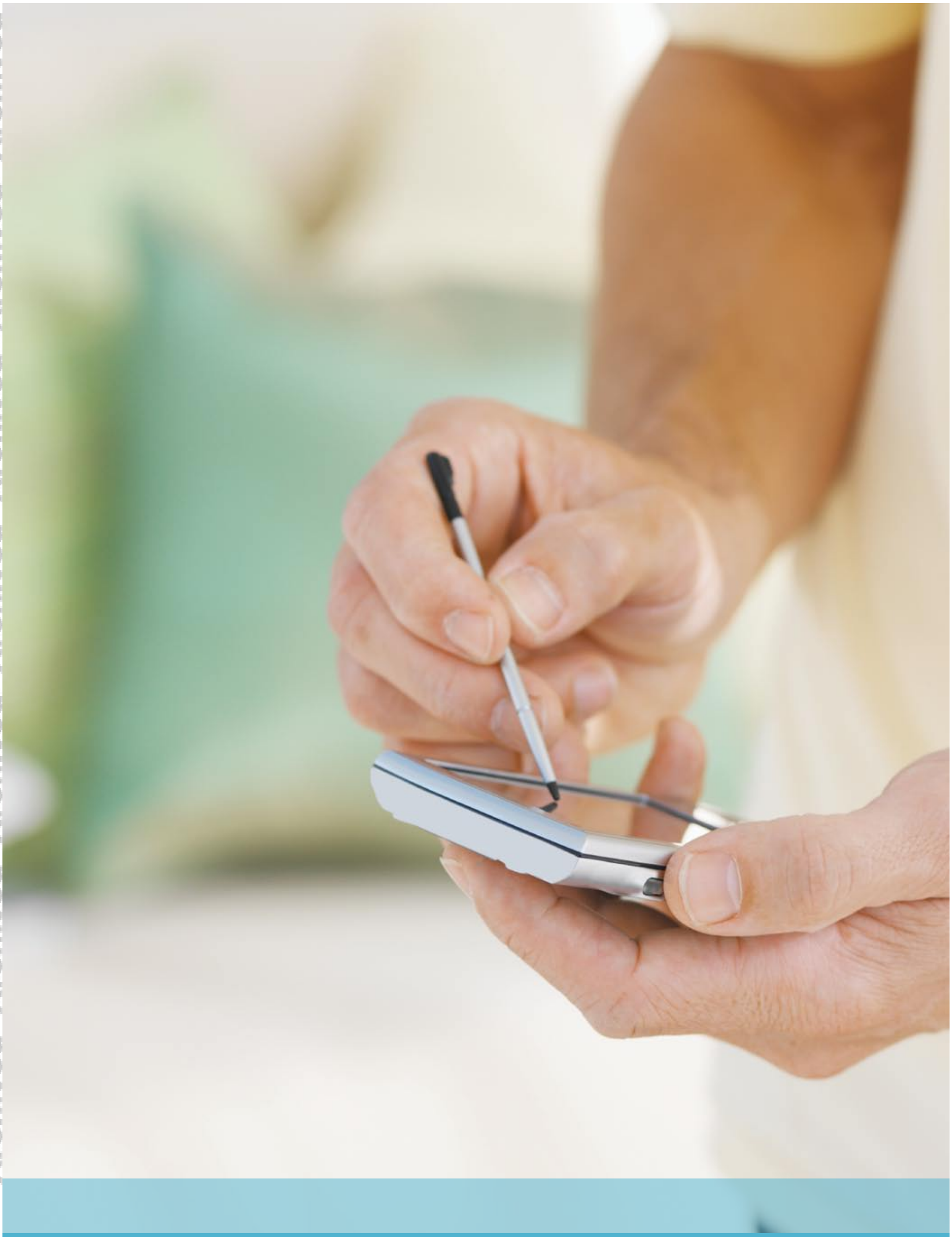
Table 2: HCS08 MCU Overview

MCU	Number of	
	TPM	Channels
MC9S08QGx	1	2 (1 8-pin)
MC9S08GBx	2	3/5
MC9S08GTx	2	2/2
MC9S08AWx	2	2/6

Table 1: Number of TPM Channels for Different Types of Motors and Operational Modes

Operational Mode	Modulation Technique	Number of TPM Channels		
		DC Motor	BLDC Motor	AC/PMSM Motor
1Q	Single PWM	1	—	—
2Q	Independent bipolar	4	6	—
	Independent uipolar	2 + 2 GPIO	3 + 3 GPIO	—
4Q	Complementary bipolar	4	6	—
	Complemetary unipolar	4	6	—
—	Sinewave modulation	—	—	6

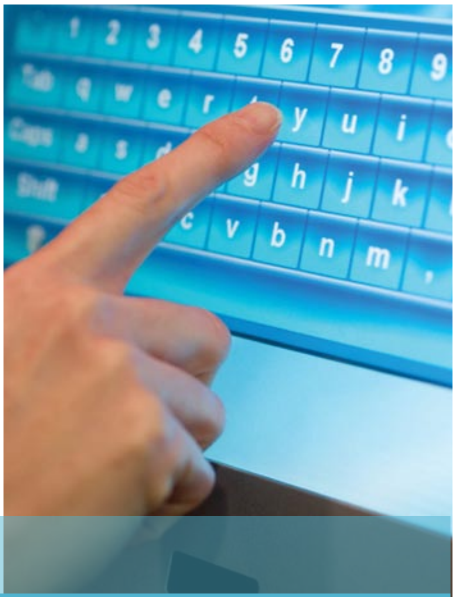
Pavel Grasblum has been working for Freescale for more than 7 years as a system application engineer. He received a Ph.D. in power electronics and electrical drives in 2001. Grasblum works with a wide range of Freescale products (8/16-bit MCUs, DSCs) supporting motor control and power conversion applications.



Product Summary
8-bit Products

8-bit Microcontrollers Product Summary

Device	Flash	RAM	USB	ADC Channels		SCI (UART)	ESCI	SPI	I²C	ACMP	Timer	Clock Type	Package					Dev Tools			Applications/Additional Features
				10-bit	8-bit								DFN/QFN	QFP/LQFP	TSSOP	SOIC	DIP	DEMO	EVB	FSICE	*All RS08, S08 and HC08 products include COP, LVI, POR and KBI
MC9S08AW60	60 KB	2 KB		16		2		√	√		6 + 2-ch.	ICG w/FLL	48	64, 44				√			High integration, flash programmable to 5V
MC9S08AW32	32 KB	2 KB		16		2		√	√		6 + 2-ch.	ICG w/FLL	48	64, 44				√			High integration, flash programmable to 5V
MC9S08AW16	16 KB	1 KB		16		2		√	√		4 + 2-ch.	ICG w/FLL	48	64, 44				√			High integration, flash programmable to 5V
MC9S08QG8	8 KB	512B		8		√		√	√	√	2-ch., MTIM	ICS	8, 16		16	8	16	√			High performance, low voltage, small package
MC9S08QG4	4 KB	256B		8		√		√	√	√	2-ch., MTIM	ICS	8, 16		16	8	8, 16	√			High performance, low voltage, small package
MC9RS08KA2	2 KB	62B								√	MTIM	ICS	6			8	8	√			Ultra-low end, new RS08 core for small MCUs
MC9RS08KA1	1 KB	62B								√	MTIM	ICS	6			8	8	√			Ultra-low end, new RS08 core for small MCUs
MC9S08GT16A	16 KB	2 KB		8		√		√	√		3 + 2-ch.	ICG	48, 32	44			42	√	√		High performance, flash programming down to 1.8V
MC9S08GT8A	8 KB	1 KB		8		√		√	√		3 + 2-ch.	ICG	48, 32	44			42	√			Flash programming down to 1.8V, small package
MC9S08QD4	4 KB	256B		4							2 + 3-ch.	ICS				8	8	√			Low-end, flash programmable to 5V
MC9S08QD2	2 KB	128B		4							2 + 3-ch.	ICS				8	8	√			Low-end, flash programmable to 5V
MC9S08LC60	60 KB	4 KB		8 12-bit		√		2	√	√	2 + 2-ch.	ICG w/FLL		80, 64				√			Integrated Liquid Crystal Display (LCD) driver with high segment count
MC9S08LC36	36 KB	2.5 KB		8 12-bit		√		2	√	√	2 + 2-ch.	ICG w/FLL		80, 64				√			Integrated Liquid Crystal Display (LCD) driver with high segment count
MC9S08QE128	128 KB	8 KB		24 12-bit		2		2	2	2	1 + 6-ch, 2 + 3-ch.	ICS		80, 64				√	√		Ultra-low power S08 device with 1.8V to 3.6V op range
MC9S08QE64	64 KB	4 KB		24 12-bit		2		2	2	2	1 + 6-ch, 2 + 3-ch.	ICS		80, 64				√	√		Ultra-low power S08 device with 1.8V to 3.6V op range
MC908QC16	16 KB	512B		10			√	√			2-ch. 16-bit 4-ch. 16-bit	OSC		16, 20, 28		16, 28		√		√	Two independent timer blocks, increased flash and RAM



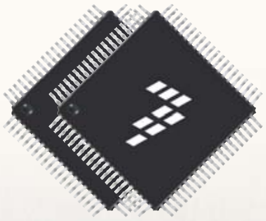
Product Summary
32-bit Products

32-bit Microcontrollers Product Summary

Device	Freq. (MHz)	MIPS @ Max Freq.	Cache (KB)	SRAM (KB)	Flash (KB)	DMA	GPT*	PIT	10/100 FEC	Crypto	USB	CAN	I2C	UART/ USART/ PSC	SPI	ADC	Other	GPIO Max	Debugging Interface	Package
MCF51QE	50	46		8, 4	128, 64		6-ch., 16-bit 2x 3-ch., 16-bit						2		2	12-bit	RTC w/32kHz Osc	70 16-bit RGPIO	ICE and BDM	LQFP 80, LQFP 64
MCF520x	166	159	8 K I/D	16		16-ch.	4-ch., 32-bit	2	1 (Optional)				1	3 UART	QSPI		DR/SDR SDRAM Controller	30, 50	JTAG and BDM	LQFP 144, QFP 160 MAPBGA 144 MAPBGA 196
MCF521x	66, 80	76		16, 32	128, 256	4-ch.	4-ch., 32-bit 4-ch., 16-bit	2				1 (Optional)	1	3 UART	QSPI	12-bit		33, 44, 55	√	LQFP 64, LQFP 100 MAPBGA 81
MCF521xx	66, 80	76		16	64, 128	4-ch.	4-ch., 32-bit 4-ch., 16-bit	2					2	3 UART	QSPI	12-bit	RTC w/32kHz Osc	56	JTAG and BDM	LQFP 64, MAPBGA 81
MCF5221x	66, 80	76		16	64, 128	4-ch.	4-ch., 32-bit 4-ch., 16-bit	2			Full-Speed Device/ Host/OTG		2	3 UART	QSPI	12-bit	RTC w/32kHz Osc	56	JTAG and BDM	LQFP 64, MAPBGA 81
MCF5222x	66, 80	76		16, 32	128, 256	4-ch.	4-ch., 32-bit 4-ch., 16-bit	2			Full-Speed Device/ Host/OTG		1	3 UART	QSPI	12-bit	RTC	56	JTAG and BDM	LQFP 64, MAPBGA 81
MCF5223x	60	57		32	128, 256	4-ch.	4-ch., 32-bit 4-ch., 16-bit	2	1	Optional		1 (Optional)	1	3 UART	QSPI	12-bit	EPHY, RTC	73	JTAG and BDM	LQFP 80, LQFP 112, MAPBGA 121
MCF523x	80, 100, 150	144	8 K I/D	64		4-ch.	4-ch., 32-bit	4		Optional		1 Standard 2nd Optional	1	3 UART	QSPI		16-ch. eTPU Optional 32-ch. eTPU SDR SDRAM	142	√	QFP 160, MAPBGA 196 MAPBGA 256
MCF5253	140	125	8 K I	128		4-ch.	2-ch., 16-bit				High-Speed On-the-Go	2	2	3 UART	QSPI	12-bit	IDE, I2S SDR SDRAM	60	JTAG and BDM	MAPBGA 225
MCF527x	100, 166	96, 159	8 K I/D 16 K I/D	64		4-ch.	4-ch., 32-bit	4	1 Standard 2nd Optional	Optional	Optional Full-Speed Device		1	3 UART	QSPI		SDR SDRAM DDR SDRAM	61, 69, 97	√	QFP 160, MAPBGA 196 MAPBGA 256
MCF528x	66, 80	76	2 K I/D	64	256, 512	4-ch.	4-ch., 32-bit 8-ch., 16-bit	4	1			1	1	3 UART	QSPI	10-bit	SDR SDRAM DDR SDRAM	150	√	MAPBGA 256
MCF532x	240	211	16 K I/D	32		16-ch.	4-ch., 32-bit	4		Optional	Full Host, Full/High** OTG	1 (Optional)	1	3 UART	QSPI		SVGA LCD DDR/SDR SDRAM	63, 97	JTAG and BDM	MAPBGA 196 MAPBGA 256
MCF537x	180, 240	158, 211	16 K I/D	32		16-ch.	4-ch., 32-bit	4	1	Optional	Optional Full Host, Full OTG		1	3 UART	QSPI		DDR/SDR SDRAM	46, 62	JTAG and BDM	QFP 160, MAPBGA 196
MCF547x	200, 266	308, 410	32 K I 32 K D	32		16-ch.	4-ch., 16-bit	2	2	Optional	Optional High-Speed Device		1	4 PSC	DSPI		PCI DDR/ SDR SDRAM	83, 99	√	PBGA 388
MCF548x	166, 200	256, 308	32 K I 32 K D	32		16-ch.	4-ch., 16-bit	2	1 Standard 2nd Optional	Optional	Optional High-Speed Device	2	1	4 PSC	DSPI		PCI DDR/ SDR SDRAM	83, 99	√	PBGA 388

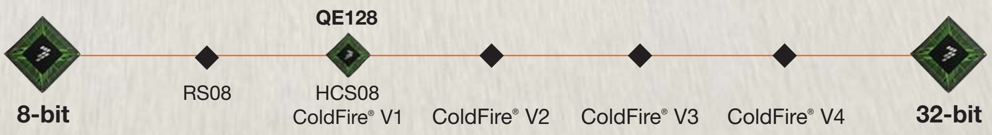
*GPT may support PWM and/or DMA capabilities
**USB On-The-Go high-speed functionality via ULPI Interface

This is a select list of our products. To obtain the full product summary documents for our 8- and 32-bit microcontrollers, search “8BITCIPRODMPFS” and “BRCOLDFIRESUM” at www.**freescale.com**.


MC9S08QE128
MCF51QE128

It's official: 8-bit and 32-bit are now compatible.

The Controller Continuum: Flexis™ MCU Series



Introducing the Flexis series: the world's first 8-bit and 32-bit pin, peripheral and tool compatible microcontrollers. The QE128 family, our first device duo in the series, gives you a single migration path across a price and performance spectrum. Now when your designs demand, you can easily move between 8-bit and 32-bit while sharing tools such as our new CodeWarrior® Development Studio for Microcontrollers with Processor Expert.™ Plus, peripherals that run on ultra-low power help extend battery life, yet another competitive advantage. Accelerate your time to market with the unprecedented design flexibility of the Controller Continuum. Only from Freescale.

View our free webcast at freescale.com/flexis

freescale™ semiconductor