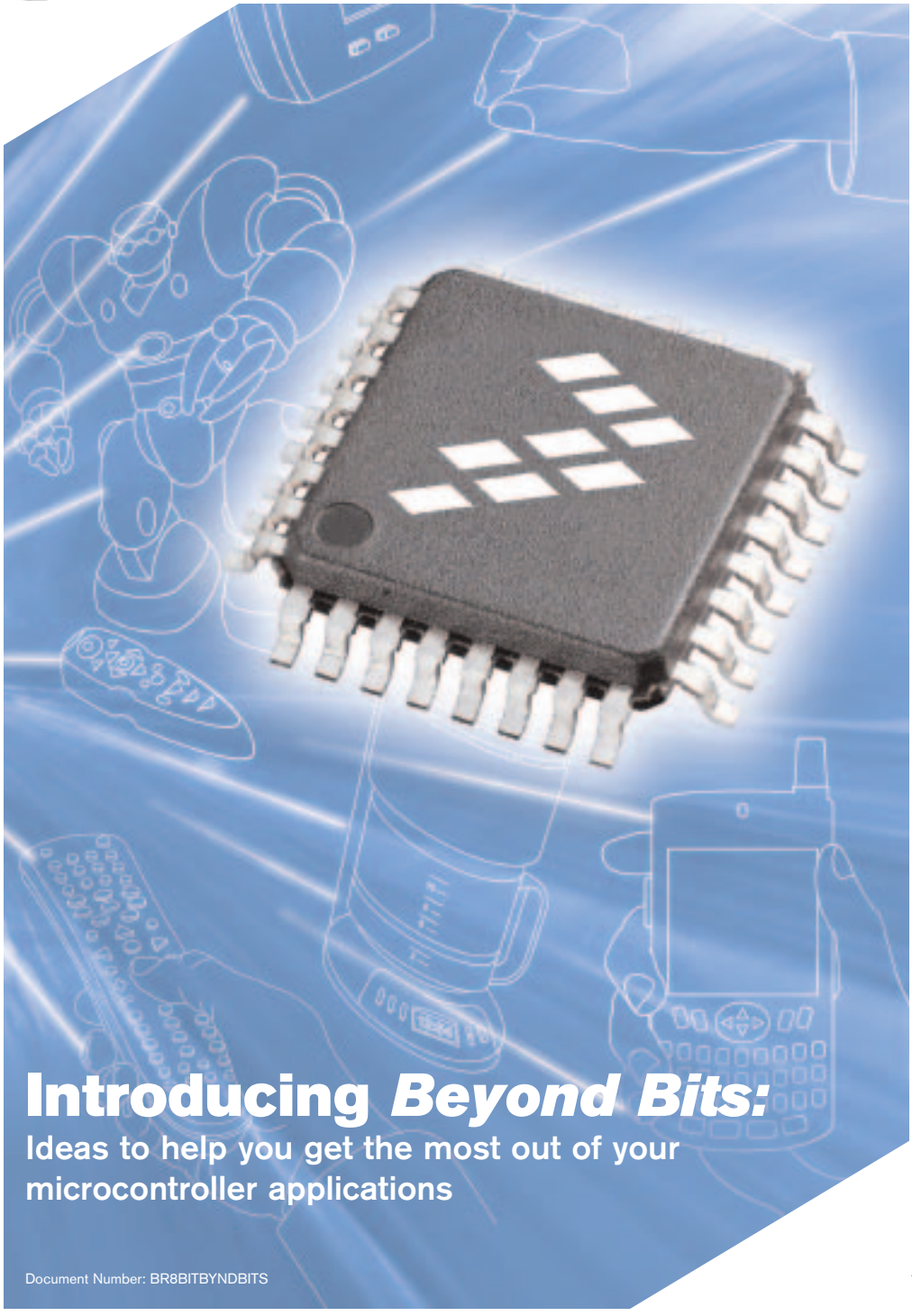
Spring 2006

8-bit Designer Resource. Accelerate. Simplify.

Beyond Bits



Introducing *Beyond Bits:*

Ideas to help you get the most out of your microcontroller applications

Inside:

- Put your 8-bit design on the fast track. Tap into practical design techniques to simplify your product development.
- Value doesn't end with the silicon. Discover services that help you streamline the design process and get products to market faster—with more efficient and economical results.
- Time-saving guides are designed to help you choose the right tools and microcontroller peripherals.

Document Number: BR8BITBYNDBITS

freescale semiconductor

Put Time on Your Side

In the dynamic world of embedded control systems, time is often the designer's most precious commodity. *Beyond Bits* can help put time on your side. In these pages, you'll find comprehensive and practical ideas to help save you time as you turn your design into a real product.

Last year, engineers used more than 4 billion 8-bit MCUs to design their end-customer products. When we created *Beyond Bits*, we took into account how Freescale's customers around the world use our technology. Whether you're already an experienced user or you're preparing to use our technology for the first time, these articles and expert tips can help speed you through the process, every step of the way, enabling you to get products on the market faster.

The language and topics of these articles are compiled to be useful for all levels. We address topics as far reaching as choosing the right MCU and development tools, designing for optimum EMI/EMC performance, future-proofing your design by using compatible families of products and defining some of the most popular communications protocols. We share information about the added services that Freescale offers to cover your MCU needs.

Freescale has evolved the humble 8-bit MCU into a richly featured, high-performance, cost-effective MCU. Our powerful and elegant Complex Instruction Set Core (CISC) provides almost 100 instructions, onboard Flash, a wide range of power capabilities and both digital and analog peripherals. Freescale's 8-bit MCUs are engineered to eliminate compromise when dealing with budgets and deadlines.

We know today that silicon is just a part of a great design. We recognize that each project has complex and differing product requirements. Your industrial, consumer or automotive application may need to connect with something, conserve energy, provide security or allow for updates—or perhaps do all of these things. We recognize that code needs to be written efficiently and that you need the safety net of a higher memory option. We recognize that time is imperative.

Beyond Bits was designed to help you unleash the power of these products in the shortest amount of time. Keep this magazine close to you during every phase of development.

We thank you once again for considering Freescale for your MCU design.

Regards,

Edward Sinnott
Global 8-bit Consumer and Industrial Marketing Manager
P.S. *Beyond Bits* contains a selection of useful articles that are also published in Freescale's library of resources. For more helpful tips on 8-bit, visit **www.freescale.com/8bit**.

We want to hear from you!
Please send your product ideas, tips or questions to us! e-Mail **MCUideas@freescale.com**.

Table of Contents

Practical Techniques for Cost-Effective Protection of Electrical Fast Transients

Improving the Transient Immunity Performance of Microcontroller-Based Applications

This article was excerpted from Freescale Semiconductor Application Note 2764.

By Ross Carlton

Increased competition among appliance manufacturers, as well as market regulatory pressures, are forcing original equipment manufacturers (OEMs) to reduce the cost of their products while ensuring compatible operation in increasingly severe electromagnetic environments. As a result of this focus on cost control, implementing the necessary transient immunity protections to prevent appliance malfunction due to transients on power and signal lines is becoming ever more challenging for the appliance designer. As traditional power supply designs and electromagnetic interference (EMI) controls are sacrificed for lower cost solutions, the appliance designer must employ new techniques to meet the applicable regulatory requirements.

In this article, we discuss the effects of electrical disturbances on embedded microcontrollers (MCUs) and suggest practical hardware and software design techniques that provide cost-effective protection for electrical fast transients (EFT) and other power line or signal line transients of short duration. While this discussion is targeted at appliance manufacturers, the principles also apply to applications in consumer, industrial and automotive electronics.

Transient Immunity Environment

The transient immunity environment for commercial electrical and electronic products includes both electrostatic discharge (ESD) and electrical fast transients (EFT). These transients are defined in IEC 61000-4-2[1] (or ANSI C63.16) and IEC 61000-4-4[2], respectively. These standards include test methods performed by the OEM designer to meet product specifications and regulatory requirements.

The ESD waveform is intended to simulate the discharge from a human operator. The electrostatic discharge is injected at any location that the operator is likely to touch. This includes all user-accessible controls and external connectors. The test levels for ESD vary widely depending on application. Values for air and contact discharge can be as low as 2 kV for commercial applications or as high as 20 kV for some automotive applications. The ESD waveform specified in IEC 61000-4-2 has a rise time of 0.7 ns to 1.0 ns, resulting in a noise bandwidth ($1/\pi t_r$) of approximately 450 MHz.

The EFT waveform is intended to simulate the transients created by the switching of relays or the interruption of inductive loads on the power mains. While primarily intended for injection on the product's AC power cord, the EFT waveform can also be injected onto signal and control lines to simulate the coupling of the EFT onto these lines. While test levels for the EFT transient are specified with amplitudes up to 4 kV, higher levels of immunity performance are sometimes required for particularly severe environments. The EFT waveform specified in IEC 61000-4-4 has a rise time of 3.5 ns to 6.5 ns resulting in noise bandwidth ($1/\pi t_r$) of approximately 90 MHz.

Issues in Embedded Applications

Cost-effective, microcontroller-based embedded applications are particularly susceptible to performance degradation during ESD and EFT events. This sensitivity to fast rise-time transients is to be expected, even for microcontrollers running at relatively low clock frequencies. This sensitivity is due to the process technologies employed. Today's semiconductor process technologies for cost-effective, 8-bit and 16-bit MCUs implement transistor gate lengths in the 0.65 µm to 0.25 µm range. These gate lengths are capable of generating and responding to signals with rise times in the sub-nanosecond range (or an equivalent bandwidth of greater than 300 MHz). As a result, an MCU is capable of responding to ESD or EFT signals injected onto its pins.

In addition to the process technology, MCU performance in the presence of an ESD or EFT event is affected by the design and package of the IC, the design of the printed circuit board (PCB), the software running on the MCU, the design of the system and the characteristics of the ESD or EFT waveform when it reaches the MCU. Only the design of the IC and its package are under the control of the IC manufacturer. All other areas that impact the immunity of the application are under the OEM's control.

Several facets of IC design other than physical gate length can affect MCU performance during transients. These include the composition of ESD suppression devices on input/output (I/O) pins and the layout of I/O pin structures. ESD devices range from simple diodes to complex active filters. Power supply rejection is accomplished through internal capacitance and careful routing on the die. Physical separation of pin inputs from active circuitry is a proven method to reduce transient effects but at a greater cost penalty due to die size impact.

The choice of MCU package can also affect immunity performance. The package type can have a great influence on PCB layout and composition. Surface-mount MCUs generally have smaller footprints than through-hole packages. This can reduce overall PCB dimensions and increase routing density, but it can also provide more space to implement board-level suppression techniques.

A detailed discussion of system design and PCB layout techniques to suppress transients is provided in Freescale Semiconductor Application Note 2764[3]. Software techniques that improve tolerance to transients are also presented in the application note.

Areas of MCU Vulnerability

Considering that most MCUs are specified and designed to generate and respond to signals with rise times comparable to ESD and EFT events, vulnerability to these events should be expected. Areas of MCUs typically vulnerable to ESD and EFT signals include:

> Power and ground pins
> Edge-sensitive or high-frequency digital inputs
> Analog inputs
> Clock (oscillator) pins
> General purpose I/O with multiplexed pin functions
> ESD protection circuitry

Some MCUs have multiple power and ground pins to isolate high-speed digital functions from low-speed or sensitive analog functions. These supply pins should be filtered appropriately to prevent disturbances in one area from affecting another. Low-cost MCUs may only have a single set of power and ground pins, which makes isolation difficult and makes filtering more important. It is easy to understand that a transient that gets propagated to a supply line can also disrupt internal circuitry that has no direct route to the pin that was disturbed.

Edge-sensitive inputs are particularly vulnerable to transients. These are usually timer or external interrupt inputs. Even with external low-pass filtering, a sufficiently large pulse can inject enough energy into the input area to disrupt MCU operation. Pulses that don't disrupt the MCU can still be seen as glitches by the MCU. A software technique to filter out glitches is discussed later.

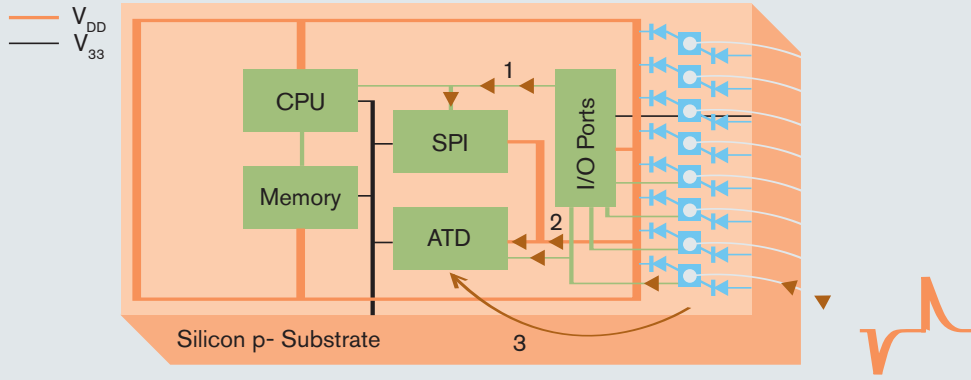
High-speed digital inputs, such as clock and data inputs, are less likely to have low-pass filtering and consequently can register transients as valid data pulses. External isolation techniques are necessary to eliminate this vulnerability.

Analog inputs are generally of lower impedance than digital inputs and can suffer physical damage if not protected during ESD and EFT transients. On most MCUs, however, the analog inputs are multiplexed with general purpose I/O pins and have a small sampling window in which the lower input impedance is active. A transient appearing in an analog input pin during an analog-to-digital conversion will result in distorted data due to the signal disruption. Effective software filtering techniques exist to mitigate this vulnerability.

Most MCUs have a built-in oscillator amplifier so that an external crystal or resonator is all that is needed to ensure a stable high-frequency system clock. The oscillator pins can pass noise pulses as valid clock edges and are considered to be the most vulnerable inputs to the system. Appropriate PCB layout is the preferred method to eliminate this risk.

As shown in Figure 1, transients can travel from the point of entry to affect circuits via several paths. System input signals that exceed the power rails of the MCU will inject current into the I/O pin structure as soon as the signal level exceeds the ESD protection diode's forward voltage. The I/O pin structure and on-chip ESD protection network

FIGURE 1. TRANSIENT CURRENT INJECTION PATHS INSIDE THE MCU



can dissipate small amounts of injected energy. However, if the injected current is greater than the local circuit can handle, this excessive current can find alternate paths through the supply rails or substrate to disrupt other circuitry. Current injection is generally minimized by using series resistors.

General purpose MCUs have I/O ports that can have more that one function multiplexed on a single pin. An electrical disturbance that causes enough energy to disrupt digital logic can also affect the control circuitry that selects the pin function. The resulting fault could change the pin state, the pin direction or the pin function.

Vulnerability is particularly troublesome for general purpose MCUs that are designed to meet the needs of many applications. For these MCUs, it is impractical or impossible to harden all vulnerable areas without adversely affecting functional performance in at least some applications.

Application-specific MCUs can be hardened with greater success, but some vulnerability will continue to exist if the operational frequency or bandwidth of the MCU overlaps the bandwidth of the ESD and EFT signals.

MCU Performance Classification

Failure modes for integrated circuits are classified into one of five categories as shown in Table 1. The classification is determined by the performance of the integrated circuit in the presence of the ESD or EFT signal. This performance is dependent on the type of integrated circuit and its functional and parametric operation as documented in its data sheet.

TABLE 1. CLASSIFICATION OF IC EMC DEGRADATION

Class	Description
A	Normal performance within the specification limits during application of the transient
B	Temporary degradation, or loss of function or performance, that is self-recoverable after the transient is removed
C	Temporary degradation, or loss of function or performance, that requires a system reset to recover after the transient is removed
D	Temporary degradation, or loss of function or performance, that requires operator intervention or cycling power to recover after the transient is removed
E	Permanent degradation or loss of function that is not recoverable due to damage or loss of data

Class A performance is the most desirable and is often required for safety-critical applications. Of course, this level of performance is difficult to ensure without taking proper steps in the design of the application. This is because any transient appearing at a pin that can be processed by the input circuitry has the potential for being interpreted as data and corrupting program execution. Class B performance is considered acceptable for most applications where the main requirement is for no user intervention to recover normal performance. Class C performance can be acceptable for particular applications, or where an external watchdog circuit is used. Class D and E performance levels are not acceptable.

MCU Failure Modes

For MCUs, performance degradation can take many forms. Common forms of temporary degradation include, but are not limited to reset, latch-up, memory corruption and code runaway. MCUs with internal reset circuits can generally

resume operation without operator involvement if the fault is an unexpected reset or code runaway that is caught by a watchdog timer. Recovery from latch-up and volatile memory (RAM) corruption requires cycling the power to the system. Non-volatile memory corruption (Flash, EEPROM) requires a more extensive process of re-programming the system, which can be viewed as a temporary MCU degradation if the system can be reworked or as permanent if it cannot be reworked.

Permanent degradation can include increased leakage current on I/O pins, which can affect analog measurements, input impedances and output drive strength. With increased leakage current, the electronic system may still operate within specification for a while, but it may ultimately fail due to damage from the transient stress. Another type of permanent degradation found in transient environments is destroyed circuitry or bond wires due to an electrical overstress.

Impact of MCU Design Trends

The MCU design trend that particularly impacts transient immunity performance is the drive to continually reduce the minimum gate length of individual field-effect transistors (FETs), making them smaller and faster. This trend is the result of market pressure on semiconductor manufacturers to reduce the cost of their products by making die sizes smaller. The result is that maintaining the immunity performance of MCUs in the face of process technology advances is becoming increasingly difficult. When coupled with continuing cost reductions by OEMs at the application or system level, the immunity problem becomes severe.

MCU designers are challenged to develop better methods to dissipate the energy injected during a transient event. While they would appreciate more area in which to include transient suppression circuits, this is generally not allowed in order to keep the die size and cost to a minimum. Some of the remaining options available to the designer include modifying semiconductor attributes (doping and materials) and changing the vertical structure of the I/O pin.

Conclusion

Achieving transient immunity in a cost-effective embedded application can be a difficult and time-consuming process, particularly if not addressed early and often in the design of an application. Not addressing transient protection as close to the AC mains as possible will increase the complexity of the EMC problem. The initial design of an embedded application should maximize EMC so that design budgets and production schedules are met without delays at the EMC-compliance stage. Cost reductions can be easily implemented at a later date if the desired EMC performance is still achieved. It is always easier to remove components while in production than it is to add them late in the design process.

In general, the EFT or ESD performance of a system can be dramatically affected by the choices made in the software architecture and operation. As stated earlier, these techniques should be viewed as a necessary but last line of defense against adverse system reaction to EFT or ESD events. The software can affect how the system will react to a disturbance if it reaches the MCU, but the hardware PCB board and system hardware design should diminish or eliminate the disturbance before it reaches the MCU.

References

IEC 61000-4-2, *Electromagnetic Compatibility (EMC)-Part 4-2: Testing and Measurement Techniques—Electrostatic Discharge Immunity Test*, International Electrotechnical Commission, 2001.

IEC 61000-4-4, *Electromagnetic Compatibility (EMC)-Part 4-4: Testing and Measurement Techniques—Electrical Fast Transient/Burst Immunity Test, International Electrotechnical Commission*, 2001.

Ross Carlton, Greg Racino and John Suchyta, "*Improving the Transient Immunity Performance of Microcontroller-Based Applications*," Freescale Application Note, AN2764.

Ross Carlton is an internationally recognized expert in integrated circuit EMC and most recently contributed to EMC of Integrated Circuits, the first book on the subject.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Sophisticated and Powerful Peripherals Expand 8-bit Applications

By Gerald Kupris

The HC(S)08 features sophisticated built-in peripheral functions that make these microcontrollers useful in a wide range of applications. The powerful peripherals help reduce stress on the core while providing more calculating power and flexibility to the application itself.

Clock

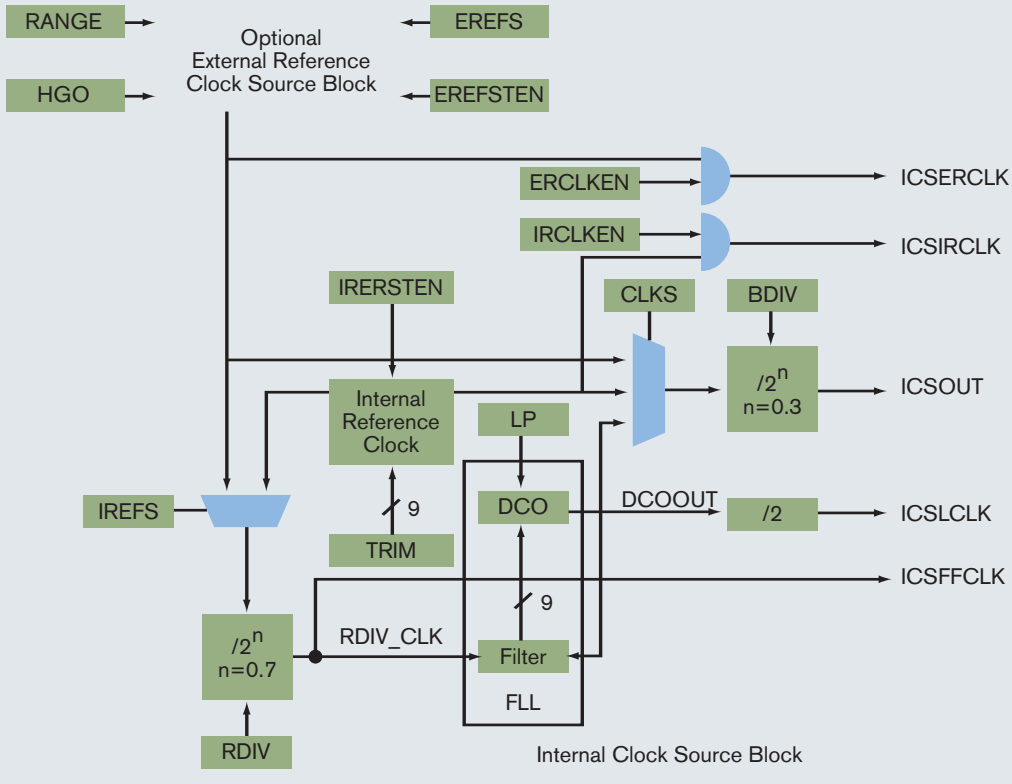
The clock source and distribution is a critical part of any system design. Microcontrollers of the 68HC08 families can be clocked with a maximum internal bus frequency of 8 MHz, and HC(S)08 devices can be clocked with a maximum internal bus frequency of 20 MHz. However, different modes of clock generation are used to create this frequency in different derivatives of the HC(S)08 products.

The MCS08QG Family's internal clock source (ICS) module (Figure 1) 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 using only simple digital logic techniques, eliminating the need for additional external components typically required for Phase-Lock Loops (PLL).

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 has a maximum accuracy of 0.1 percent.

FIGURE 1. BLOCK DIAGRAM OF THE ICS MODULE OF THE MCS08QG



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 and XOSC sources may 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 battery life without switching the microcontroller into power-saving modes.

Timer

The timer is the most important and most fundamental peripheral element on all microcontrollers, and is necessary for all tasks that are relative to real time. Every microcontroller of the HC(S)08 has at least one well-equipped and flexible, programmable 16-bit timer interface module (TIM) that is connected to the external circuitry via at least two I/O channels.

In Figure 2, you can see the structure of the timer interface module. The most important element is a 16-bit counter, which can be configured as a free running counter or as a modulo counter. This counter provides the time reference for all operations such as input capture or output compare. The modulo register TMOD contains the modulo value of the counter. The modulo comparator permits the 16-bit counter to count upward, until its value corresponds to the value stored in the modulo register. The counter is then reset and a new counting sequence begins.

The present counter value can be read by software anytime, without influencing the counting process. The 16-bit counter is clocked with a signal, derived from the internal bus clock of the microcontroller. The clock can be reduced up to the ratio 1:64 by a prescaler before it is supplied to the counter.

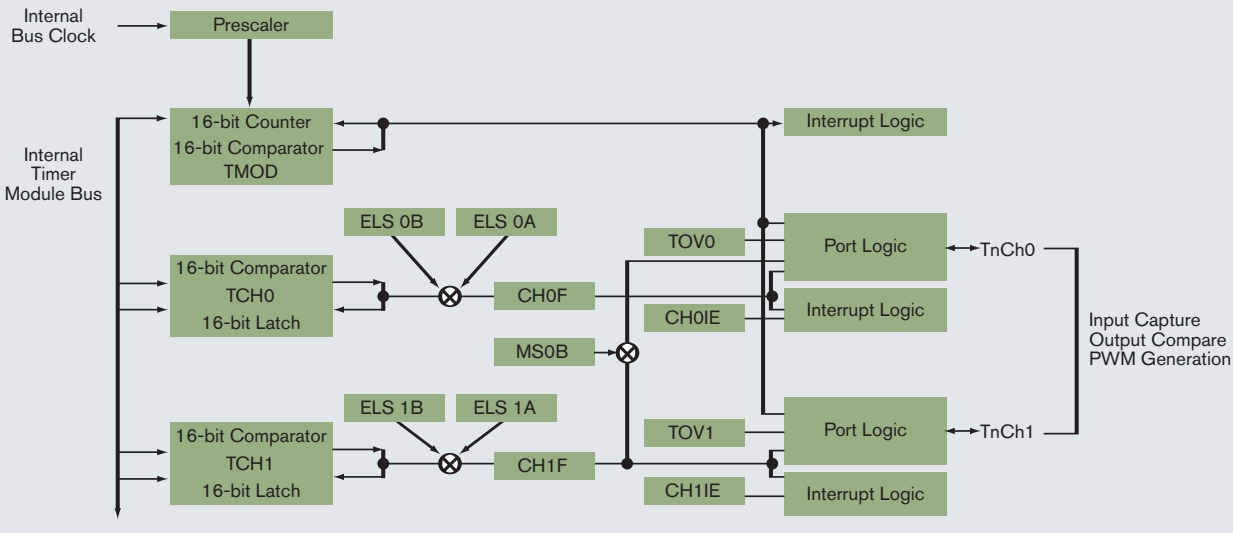
The HC(S)08 timers have timer channels, which can be selected by the user and configured as either input capture (IC), output compare (OC) or pulse-width modulators (PWM).

Analog-to-Digital Converter (ADC)

ADCs, the basic equipment of a microcontroller, are very important links to the analog world. ADCs capture signals and data. Older 68HC08 microcontrollers have an 8-bit ADC on-chip, while newer HCS08 controllers have a 10-bit ADC.

Every converter has an input multiplexer unit, which makes it possible to convert several analog signals into digital information. Depending on the exact derivative, different numbers of analog signals can be input into the ADC, providing a simple interface to any sensors with analog

FIGURE 2. STRUCTURE OF THE TIMER INTERFACE MODULE



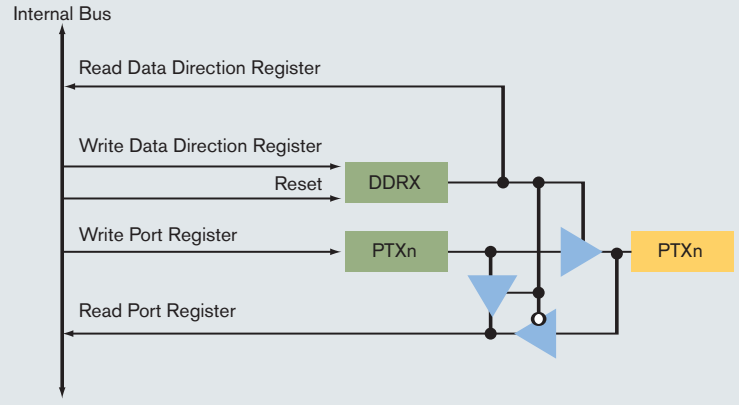
output signals. The channel sequence to be converted is fully controlled by the software.

The ADC of the MC68HC908GP32 has eight channels used in multiplexed mode. It has a resolution of 8 bits and operates in accordance with the principle of successive approximation. On the MC68HC908GP32, the inputs of the ADC are located on port B.

When the conversion is completed, the module can trigger an interrupt or set a respective flag. The module can be configured for single or continuous conversion. When set to continuous conversion, the data register is filled after each conversion with the new result and the old result is overwritten.

The MC9S08QGx is one of the latest microcontroller families and contains a very sophisticated 8-channel, 10-bit ADC module. It 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. 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. This feature can be used to wake up a microcontroller from power-saving modes provided the ADC internal clock is used as the ADC clock source.

FIGURE 3. BLOCK DIAGRAM OF A DIGITAL I/O PORT



Digital Input and Output

The easiest way for a microcontroller to communicate with its environment is via its digital inputs and outputs. On the HC(S)08, these connections are bidirectional, i.e., they can be configured as either inputs or outputs (I/O), usually referred to as low-speed I/O. Furthermore, on most derivatives, many of these pins are multiplexed, which means that they can be used as low-speed I/O or configured, for example, as timer pins. The I/Os are generally configured in groups of eight (ports A, B, C and so on) so that data exchange can be carried out comfortably in 8-bit wide registers. However, all 8 bits of an I/O-port are not always used as low-speed I/O.

FIGURE 4. KEYBOARD INTERRUPT MODULE

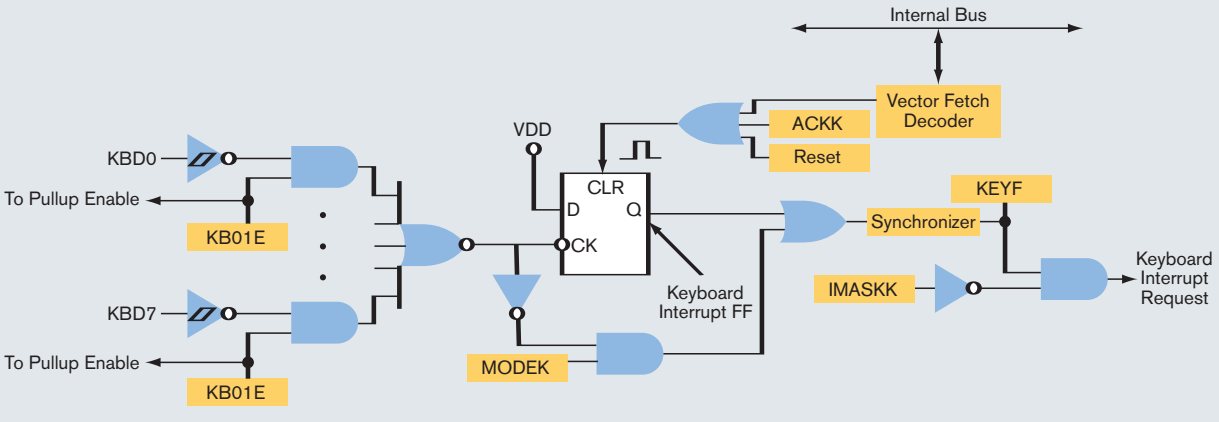
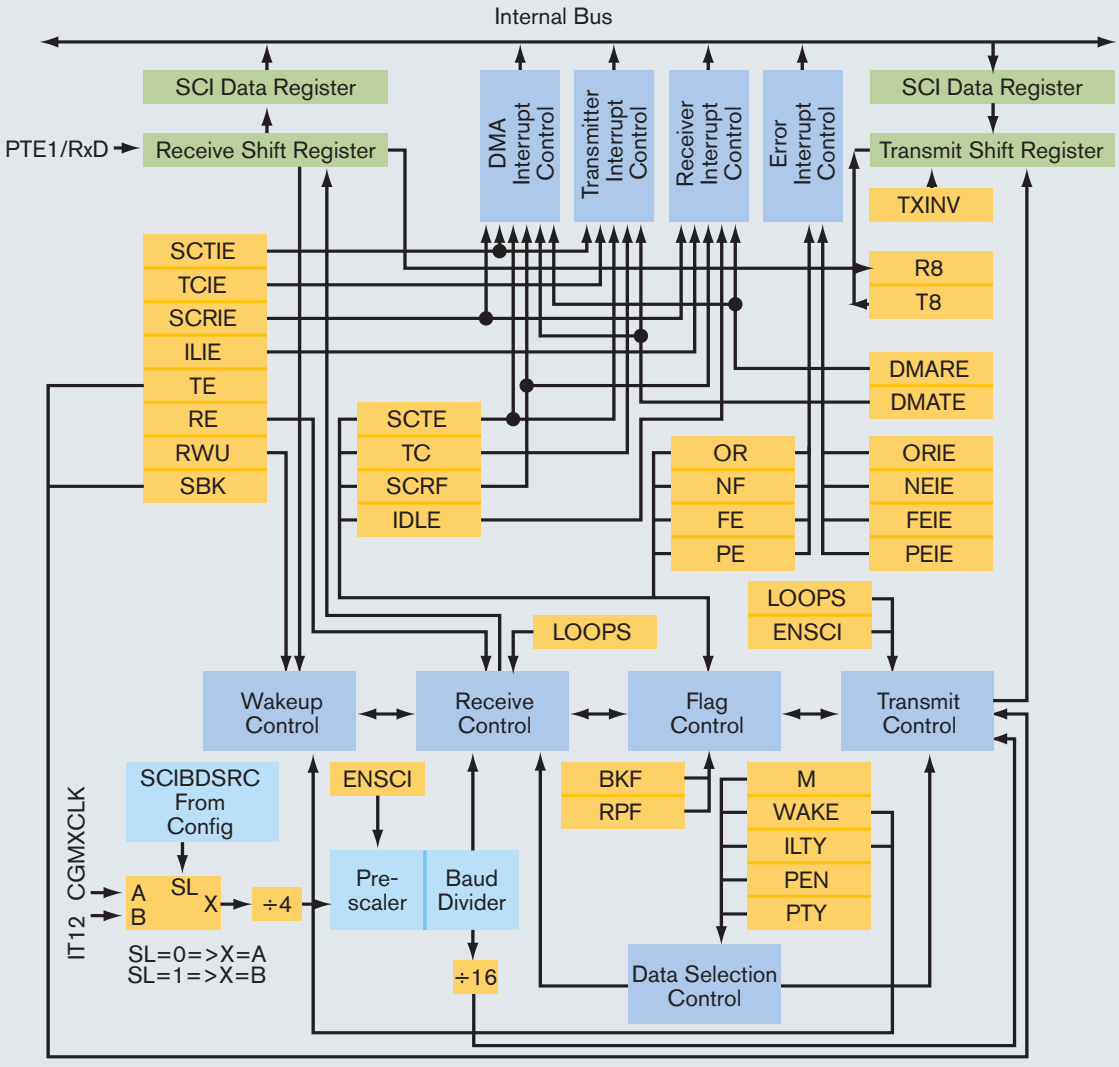


FIGURE 5. STRUCTURE OF THE SCI MODULE



In Figure 3, the general structure of a digital pin is illustrated using I/O-port A as an example. You can see that every I/O-pin has a dedicated bit in a data direction register and in a data register. Writing to the data register only has an effect on the pin if the corresponding bit in the data direction register is set to 1. The pin is then configured as an output, and the content of the data register is transferred to the pin. In order to avoid undesirable spikes at the outputs, it is recommended to write first to the data register and then the data direction register.

When a bit in the data direction register is cleared to 0, the corresponding pin is configured as an input. In this case, a read from the data register results in the value of the logical signal at the pin. A write to the corresponding bit in the data register may be carried out, but it has no effect on the pin until it is configured as an output by setting the corresponding bit in the data direction register.

After reset, all data direction registers are loaded with the value 0, ensuring that all I/O-pins are configured as input.

It must be noted that the digital I/O-pins on various derivatives can have further characteristics, such as:

> Pull-up resistor
> Open-drain output
> High current drive output
> Multiplexed with peripheral elements, such as ADC, timer, and so on

The configuration of the ports and availability of additional options depend on the particular HC(S)08 derivative and are documented in the respective data books.

An interesting option is available for many HC(S)08 microcontrollers: the keyboard interrupt module. This module can trigger an interrupt through one of several pins. A simplified description of the keyboard interrupt module is shown in Figure 4.

Every pin of the relevant port can be configured to trigger or not trigger an interrupt. When the pin is activated and pulled low, the interrupt is triggered. Select a level or edge sensitive interrupt via the MODEK-bit. When the option “level sensitive” is selected, the interrupt status remains unchanged as long as one of the selected pins is low.

The Serial Communication Interface (SCI)

The SCI allows asynchronous serial communication at high speed with peripheral component groups or other microcontrollers. Some derivatives of the HC(S)08 contain an SCI module.

The characteristics of the module are:

> Full duplex functionality
> Standard NRZ (non-return-to-zero) format
> 32 programmable baud rates
> Programmable 8- or 9-bit data format
> Separate transmitter and receiver
> Separate transmitter and receiver interrupts
> Programmable transmit polarity
> Two receiver wakeup options (signal-wakeup and wakeup at a certain address)
> Interrupt-based operation with 8 interrupt flags
> Fault detection at receiver
> Parity check through hardware
> 16-times sampling per bit
> Programmable clock source

The structure of the SCI module can be seen in Figure 5. The module is very complex, and is configured and controlled by a large number of diverse registers.

Details of the operation and configuration of the SCI module can be found in the data book or the description of the respective controller. The SCI can operate in 8- or 9-bit mode. The transmitter and receiver parts of the SCI module work independently, but use the same baud rate. This can be derived from the bus frequency of the controller and adjusted through the prescaler and baud rate register to 32 different values.

At one point in time, the SCI can send or receive only one message, which can be either 8 bits or 9 bits long. It is, however, frequently necessary to send and receive several messages in a packet, such as a data stream. If the CPU has to wait between bytes until the individual messages arrive or are sent, then this time is not available for the actual application.

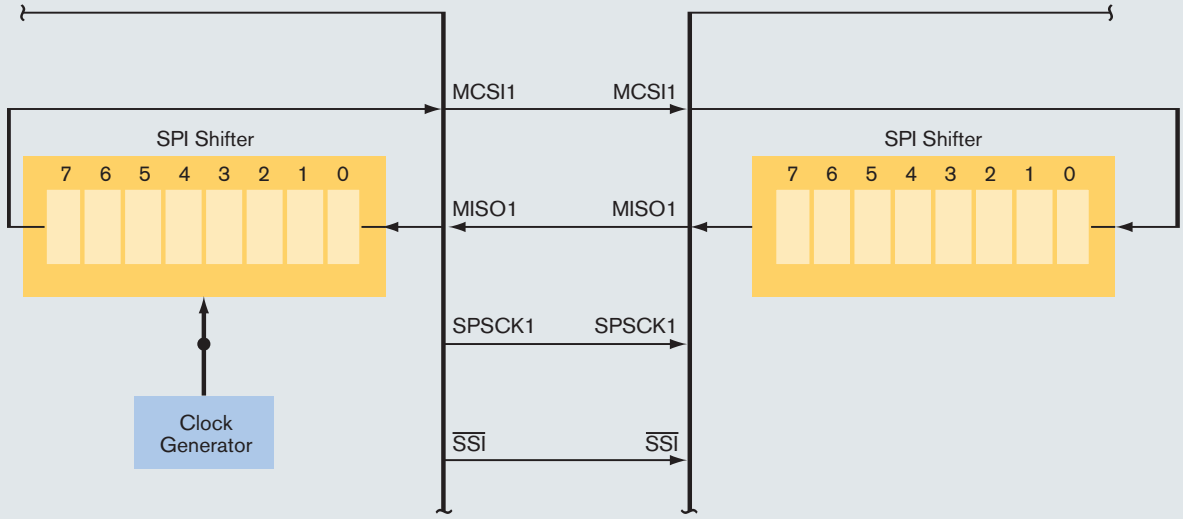
This inefficiency can be circumvented by using interrupt routines and software buffers for data transmission and reception. The data only have to be written to the transmit buffer and the transmit interrupt enabled to send a message consisting of several bytes. While one byte after another is sent through the transmit interrupt routine, the main routine can carry out other tasks. Similarly, the data are written to the receive buffer with the help of the receive interrupt routine.

The Serial Peripheral Interface (SPI)

A synchronous SPI is integrated into nearly all HC(S)08 derivatives. Shown simply, the SPI module consists of an 8-bit write register and a clock generator. It makes very fast serial data transfer possible.

The SPI module of the HC(S)08 microcontroller can be used as a master (clock source) node or as a slave (clock receiver). The serial output of the master (MOSI, Master Out, Slave In) is connected with the serial input of the slave. The data transfer can then be affected from the master to the slave. If the serial output of the slave is connected with the serial input of the master, a data transfer from slave to master is possible. Data can be simultaneously transferred from master to slave and from slave to master (full duplex operation).

FIGURE 6. STRUCTURE OF THE SPI MODULE



You can see in Figure 6 that the clock generated by the master shifts data into and out of the shift registers in the master and the slave. A transfer is completed after eight clock pulses—8-bits are transferred when the shift registers have exchanged their contents. During each transfer, which only the master can initiate, each node can simultaneously send and receive data. The functions "only send" or "only receive" are not available. It is of course possible to use only one data direction, which requires only two signal lines between the nodes. For simultaneous sending and receiving, three signal lines are required:

> Master Out Slave In (MOSI)
> Master In Slave Out (MISO)
> SPSCK (SPI Serial Clock) generated by the master, and drives the slave
> Slave Select (SS), used for the selection of different slaves

The SPI module of the HC(S)08 can be operated as master or slave. It is therefore simple to connect two or more microcontrollers, or other units with an SPI interface as a multiprocessor system.

The SPI is mostly used for communicating to simple external write registers, EEPROM, real-time clocks, ADCs, digital-to-analog converters (DACs) and more. Many enhancements are possible through additional select lines. Because only two or three lines are required for the transfer of data via an SPI interface, isolation through optocouplers between the periphery and the microcontroller can be implemented at little cost.

Dr. Gerald Kupris is a Senior Field Application Engineer at Freescale Semiconductor, the former Semiconductor Products Sector of Motorola. He started 1989 as a design engineer. In 1994 he received a Ph. D. in electronics from the Technical University of Ilmenau and started his career as an application engineer. He focuses on projects with embedded processors and microcontrollers.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Freescale Drives LIN and SAE J2602 Solutions to the Forefront

By Matt Ruff

In the late 1990s, five car makers, a communications tools manufacturer, and what is now Freescale Semiconductor founded the LIN Consortium to develop a low-cost communications standard for automobiles. Local interconnect network (LIN) is the UART-based, single-master, multiple-slave networking architecture that was developed for automotive sensor and actuator networking applications. LIN provides a low-cost networking solution for connecting motors, switches, sensors and actuators in the vehicle. The LIN master node typically connects the LIN network with higher-level networks, like the controller area network (CAN), extending the benefits of networking all the way to the individual sensors and actuators.

LIN and SAE J2602 Background

Since the first major release of the LIN specification (version 1.3) in November 2002, significant changes have been made to the standard. The resulting revision, 2.0 of the specification, was released in September 2003. Figure 1 shows the substantial changes in the specification, but both versions are still used by many manufacturers.

Many communications standards only specify protocol information, often referred to as the data link layer of the protocol. A good example of this is the Bosch Controller Area Network (CAN) 2.0b specification. There are also additional specifications for physical interfaces (sometimes called the physical layer), such as the Society of Automotive Engineers (SAE) standard J-2284 for the "high-speed" physical layer for CAN. LIN defines these two elements as part of the specification, as can be seen in Figure 1, but also includes specifications for software and tools interfaces. By including these significant standardized components, LIN is a very comprehensive specification.

A task force of SAE's Vehicle Architecture for Data Communications Standards Committee also looked at the LIN specification for use in North American vehicles. This task force, SAE J2602, developed the SAE J2602 recommended practice for the use of LIN. This specification is based upon LIN 2.0, but reduced the complexity of some software elements of the LIN specification in an effort to reduce the size and complexity of embedded software required in LIN slave nodes. Figure 2 shows how these two standards are related.

Although software may vary in embedded LIN and SAE J2602 slaves, they are both based upon the same protocol specification so the microcontroller technologies used for both remains consistent. This consistency allows both standards to work together to drive total hardware volumes up and costs down.

LIN 8-bit Microcontroller Solutions

Freescale offers a wide range of 8-bit microcontrollers to implement LIN devices, from very simple LIN slaves to complex master nodes. The choice of microcontroller will vary based on performance and cost requirements of an application. The depth of Freescale's 8-bit LIN portfolio helps to ensure that there is a solution for each of these needs.

FIGURE 1. LIN 2.0 SPECIFICATION CHANGES RELATIVE TO LIN 1.3

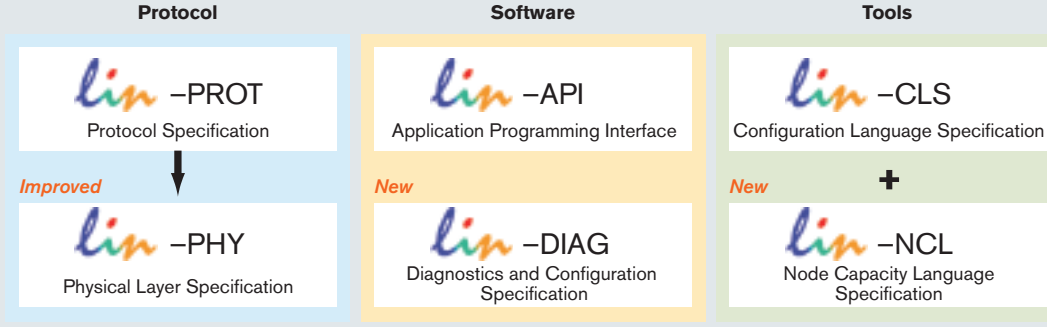
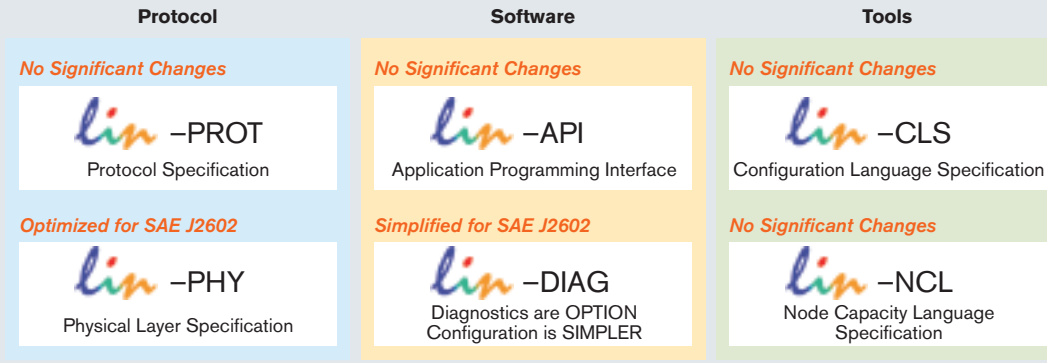


FIGURE 2. LIN 2.0 SPECIFICATION RELATIVE TO SAE J2602



LIN Masters

Common practice for LIN networks is to use LIN as a sub-network, connecting individual motors, sensors, actuators and switches together and then to a larger, faster network. In automotive applications, where LIN got its start, a LIN network would connect to a CAN bus network in the vehicle. This is done in LIN through the master node of the LIN sub-network. Since both CAN and LIN master nodes require precise timing, the cost of a more precise clock source, such as a crystal, is only incurred in the one master node.

Freescale currently offers two families of 8-bit microcontrollers that support LIN master node functionality and CAN networking.

The LIN master node controls all communications on the LIN network and provides the network time base from its accurate crystal oscillator. The resulting low-level software driver for the master is relatively simple, as it only needs to set up and schedule bytes to be sent and received in the form of LIN message headers and data. This can be done with an ordinary UART, such as the SCI on the HC908AZxxA Family devices. To improve performance, however, features have been added to the enhanced SCI (ESCI), which appear on the HC908GZ family of devices to streamline the LIN master low-level driver. For example, the LINT bit of the ESCI forces transmitted break symbols to be sent out as 13-bit long break characters (rather than the typical 10-bits for traditional UART communications) for simpler operation. The SCI can also do this, but the user must shift the baud rate temporarily by 30 percent before sending a 10-bit break symbol so that it appears on the bus as 13-bits to the LIN slaves. Then the baud rate must be restored to the operational baud rate of the network. There are several other solutions with the SCI based solution, but the ESCI simplifies the software by enabling the LINT feature at module initialization and sending break symbols normally with a single write to the SBK bit.

LIN Slaves

As the LIN master provides the base clock for the system, which is embedded in the synchronization byte of every message header, the slaves must recover this baud rate by

TABLE 1. FREESCALE 8-BIT MICROCONTROLLERS FOR LIN MASTERS

Freescale 8-bit LIN Master Microcontroller	Family Key LIN Master Features	Flash Memory Sizes Available	Application Features
HC908AZxxA	MSCAN08, EEPROM and SCI	32 KB, 48 KB and 60 KB	LIN Master with EEPROM
HC908GZ	MSCAN08, ESCI	16 KB, 32 KB, 48 KB and 60 KB	Cost-effective LIN Master with CAN Enhanced SCI for improved LIN Master software performance

measuring it when they receive the header. This is especially important in LIN, as most slave microcontrollers are designed to use internal oscillator clock sources, which are much less accurate than crystals. This is one of the key aspects of the LIN goal to reduce system implementation costs. Because a system will have many slaves and only one master, it makes sense to reduce the costs of the slaves as much as possible.

There are a wide range of microcontrollers that can be LIN slaves, as the protocol itself is simple enough to be entirely implemented in software using a timer and a general purpose output pin. For brevity, we will look at three categories of LIN slave microcontroller solutions from Freescale's 8-bit portfolio.

Bit-Banged LIN Slaves

As stated before, LIN slaves can use a timer input capture channel for reception and a general purpose output pin for the transmit channel. Two application notes, AN2503 and AN2599, detail how this bit-banging technique is done, with slightly different features implemented in each version. Additionally, you can find a reference design on the Freescale Web site that uses an HC908QY4 device to bit-bang LIN a set of steering wheel switches with PWM backlighting.

Look for it on **www.freescale.com** > Reference Designs > Connectivity > Wired Connectivity

The design is titled MC68HC908QY4 LIN Backlit Keypad Slave Reference Design (RD68HC908QY4LKS).

You can do bit-banged solutions on almost any MCU provided you have one timer channel and one output pin, but it is very software intensive. It can be a cost-effective alternative, but the microcontroller CPU has to process a large number of interrupts and the entire process of sending and receiving bytes must be performed in software at a bit level. This is more performance than many applications can spare to simply communicate.

ESCI LIN Slaves

The next step up the performance ladder comes with the advent of the Enhanced Serial Communications Interface (ESCI) module found on many HC908 Family devices. The ESCI is specifically tailored to LIN slave applications. It has an integral arbiter module that can recover the baud rate from the LIN synchronization byte in the header, as well as greatly increased resolution on its prescaler structure, which allows it to adjust for clock frequency drift due to temperature and voltage changes.

Many applications work very well with an ESCI solution, but there is one higher level of performance in the LIN slave microcontroller line. The ESCI still must handle messages on a byte level, which means that every byte of a message, whether sent or received, generates an interrupt to the microcontroller. This is also true of messages that the slave doesn't even wish to receive.

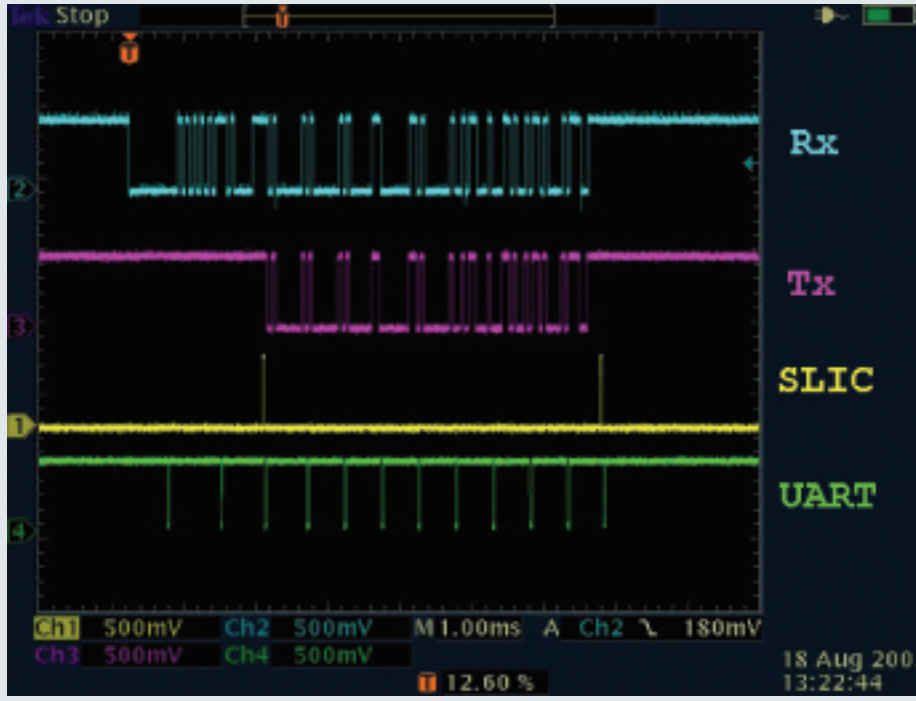
SLIC LIN Slaves

To help eliminate unwanted interrupts and fully automate the low-level LIN protocol functions to streamline interrupt service code and LIN software, Freescale has developed

TABLE 2. FREESCALE 8-BIT MICROCONTROLLERS FOR ESCI-BASED LIN SLAVES

Freescale 8-bit LIN Slave Microcontroller Family	Key LIN Slave Features	Flash Memory Sizes Available	Application Features
HC908GRxxA	ESCI	32 KB, 48 KB and 60 KB	For higher-end LIN slaves where large Flash memory sizes are required
HC908EY16A, HC908EY8A	Internal Clock Generator, ESCI	8 KB and 16 KB	Smaller Flash sizes, lower cost than HC908GRxxA, used extensively in System In a Package (SiP) single package solutions
HC908QC16	ESCI	4 KB, 8 KB and 16 KB	Smallest memory sizes with ESCI module

FIGURE 3. SLIC MODULE INTERRUPT PERFORMANCE VERSUS UART SOLUTION



the Slave LIN Interface Controller (SLIC) module. This groundbreaking LIN communications peripheral device is the result of years of research; it utilizes Freescale patented technology to synchronize to any LIN bus speed at any time. The result of these performance enhancements means that customers spend more of their performance budget on their application and less on simple communications.

The SLIC module has powerful features such as:

> True automatic frame synchronization and autobauding
  - Synchronizes to LIN traffic
  - Eliminates preprogramming of baud rate
  - Can autobaud up to 120 kbps for high-speed reprogrammability
  - No oscillator trimming needed

> LIN interrupt reductions of up to 83.3 percent (Two interrupts per standard LIN frame maximum)

> Automatic LIN error checking and reporting

> Interrupt service routines of less than 150 bytes possible

> Automatic checksum generation/checking

There are many more advantages to the SLIC module, such as an interrupt vector handling mechanism that allows extremely efficient interrupt service routine handling to minimize time spent in the routine. Combined with the full message buffering, which is designed to reduce the number of interrupts to a maximum of two for any LIN message, the SLIC module performance is amazing. Figure 3 shows the interrupt loading for an SLIC-based node and a UART based node.

The SLIC module is currently offered on the HC908QL4 device only, but is expected to expand.

Which Solution to Choose

Every application is different and has different performance and cost targets. A great number of factors must be considered when choosing the right solution for any application. To aid in this process, Freescale benchmarked the three solutions above and the results can be found in a table in AN2633 LIN Drivers for SLIC Module on the MC68HC908QL4. A list of application notes can be found in Table 3.

TABLE 3. LIN-RELATED APPLICATION NOTES

	Application Note	Title
LIN Drivers	AN2767	LIN 2.0 Connectivity on Freescale 8/16-bit MCUs Using Volcano LTP
	AN2633	LIN Drivers for SLIC Module on the MC68HC908QL4
	AN2575	MC68HC908EY16 ESCI LIN Drivers
	AN2503	Slave LIN Driver for the MC68HC908QT/QY Family
	AN2599	Generic LIN Driver for MC68HC908QY4
LIN Applications	AN2884	LIN 2.0 Door Lock Slave
	AN2885	LIN 2.0 Mirror Slave Unit
	AN2600	A Simple Keypad Using LIN with the MC68HC908QT/QY MCU
	AN2623	LIN Temperature Sensor Using the MC68HC908QY/QY MCU
	AN2573	LINkits LIN Evaluation Boards
	AN2560	MC68HC908EY16 IR Receiver for Remote Control of LIN Robot
	AN2396	Servo Motor Control Application on a Local Area Interconnect Network (LIN)
	AN2470	MC68HC908EY16 Controlled Robot Using the LIN Bus
	AN2343	HC908EY16 LIN Monitor
	AN2264	LIN Node Temperature Display
	AN2205	Car Door Keypad Using LIN
	AN2103	Local Interconnect Network (LIN) Demonstration

Conclusion

With worldwide automakers adopting the LIN and SAE J2602 standards, it is certain that LIN will continue to grow. Applications are not limited to automotive systems; LIN can also be implemented in home automation, industrial networking and many other applications where sensors, motors, switches and actuators need to be connected at low-cost.

No matter what LIN application you have, Freescale has an 8-bit microcontroller that will provide the power and performance you need.

References

Reference designs, application notes, and more can be found at the Freescale Semiconductor LIN Web site (**www.freescale.com/lin**).

An Overview of LIN; Advances in Automotive Networking Protocols: LIN Real-Time Automotive Seminar, Class 120, Fall 2004 (**www.esconline.com/seminars/RTA04/rta_papers.htm**).

Implementing Local Interconnect Network (LIN) Slave Nodes, SAE Paper Number #2002-01-1298 (Search the SAE bookstore for this paper at **www.sae.org**).

Freescale Technology Forum 2005 (Americas)–LIN Breakout Sessions:

Local Interconnect Network (LIN) and SAE J2602 Protocol In-Depth (Available at **www.freescale.com/files/ftf_2005/doc/reports_presentations/ASC134_RUFF.pdf**).

Implementing LIN and SAE J2602 (Available at **www.freescale.com/files/ftf_2005/doc/reports_presentations/ABE102_RUFF.pdf**).

Matt Ruff has been working in applications and systems engineering for the past nine years, specializing in automotive communications. His work with CAN and LIN products and standards has resulted in a patent and numerous publications, including articles, papers, and contributions to a textbook on the LIN protocol.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

USB Microcontrollers

Options Expand as Popularity Increases

By Christine Peng

Technology keeps getting faster and more customer-friendly as the Universal Serial Bus (USB) continues to take over conventional serial and parallel ports as the primary source of data communication for PC peripherals. At transfer rates of up to 480 Mbps for high-speed and 12 Mbps for full-speed USB communication, data transfer over a USB interface is at least ten times faster than over a parallel port. Furthermore, PC makers are adding more USB ports to their machines and PC peripheral companies have embraced the benefits of USB over older communication protocols by developing a plethora of new gadgets and modifying legacy peripherals using the flexibility of USB.

USB Background

Although USB was initially developed for the PC peripheral market, over the past 10 years its usefulness has extended beyond simple human interface devices (HIDs), such as mice and keyboards, to other areas, such as uninterruptible power supplies (UPS), game pads, digital cameras and mobile phones. Its tiered star bus topology is similar to that of 10 Base-T Ethernet, which gives it more flexibility by using hubs to connect multiple USB devices to one port.

In addition to its higher data rate support, a USB connection can power certain low-power devices over the bus and up to 127 devices can be connected at one time, which provides advantages over traditional serial and parallel data transfer. With its tiered star topology, USB also offers support for compound (e.g., a monitor with additional USB ports) and composite (e.g., a USB keyboard with an embedded mouse) USB devices that share one USB connection between multiple functions in a packaged device. USB is also plug-and-play compatible, making it ready to use with drivers that automatically load upon connection.

USB supports four transfer types to serve different purposes—control, bulk, isochronous and interrupt. Control transfers send USB commands as defined by the USB 2.0 specification from the host to the device. Bulk transfers send large amounts of data accurately without regard for time. Isochronous transfers, on the other hand, send set amounts of time-sensitive data at a fixed rate without regard for accuracy. Interrupt transfers occur with an upper bound on latency when the device needs attention from the host.

USB Microcontrollers

As USB became increasingly useful for various applications, companies such as Freescale Semiconductor developed microcontrollers to increase the flexibility of customer applications by combining USB with other peripherals on one chip. For instance, data transferred to the microcontroller of a device from a serial peripheral interface (SPI) can be sent directly to the PC through a USB connection. For low-voltage, low-power HID applications, 8-bit microcontrollers are ideal. Because many HID devices are bus-powered, the ability of 8-bit microcontrollers to perform basic data transmission and processing with minimal power consumption make them the optimal choice for these low-end USB applications.

TABLE 1. FREESCALE 8-BIT USB MICROCONTROLLER PRODUCTS

Freescale 8-bit USB MCU	USB Specification	Application
HC08JB1	USB 2.0 low-speed device	Cost-effective, low-end USB HID applications
HC(9)08JB8/HC08JT8	USB 2.0 low-speed device	Cost-effective, low-end USB HID applications
HC908JB12/HC908JB16	USB 2.0 low-speed device	Cost-effective USB HID applications with dual 27 MHz carrier generators
HC908JW32/HC908JW16	USB 2.0 full-speed device	Cost-effective USB FS HID applications with PS/2 clock generator
HC908LD64	USB 2.0 full-speed hub	USB FS hub with embedded functions in digital monitoring systems
HC(7)08KH12	USB 2.0 full-speed hub	USB FS hub for keyboard application

Freescale has developed a line of cost-effective, low-power 8-bit USB microcontrollers that are equipped with a variety of peripherals for various full- and low-speed device functions. These microcontrollers and their features are outlined in Table 1.

The JB series of microcontrollers feature two data endpoints with 8-byte buffers in addition to the 8-byte bidirectional control endpoint. The JB1 is a cost-effective ROM MCU targeted for low-end mouse applications. The JB8 is the Flash version with 8 kilobytes of programming space. It is ideal for mouse applications, and the Flash memory allows the user to perform in-circuit firmware upgrades through USB. The JB12/JB16 microcontrollers improve the JB series CPU clock speed and feature

TABLE 2. FREESCALE 8- AND 16-BIT USB MICROCONTROLLER REFERENCE DESIGNS

Freescale USB MCU	Reference Design	Reference Design Link
HC08JB1	USB 3D mouse	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD68HC08USB3DMSE
HC(9)08JB8/ HC08JT8	USB based sensor-actuator interface	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD68HC908USB
	USB/PS2 dual protocol multimedia keyboard	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD68HC908USBMKEYBD
	USB security key	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD68HC908USBSKEY
	High data rate wireless USB optical mouse: QY4 (for RF) + JB8 (for USB)	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD68HC908USBMSE
HC908JB12/ HC908JB16	USB based sensor-actuator interface (can be modified from JB8 version for	JB16) www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD68HC908USB
HC908JW32/ HC908JW16	N/A	
HC908LD64	USB-DDC ICP interface board	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD68HC908DDCICP
HC(7)08KH12	USB hub keyboard	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD68HC08USBHKEYBD
HC9S12UF32	USB 2.0 to ATA/ATAPI bridge	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RDHCS12UF32ATA

dual 27 MHz carrier generators. Both these devices are ideal for 27 MHz USB receiver dongle applications.

The LD64 and KH12 feature USB full-speed hubs to support specific applications such as digital monitors or keyboards with an integrated USB hub on a single device.

The latest 8-bit USB microcontrollers released by Freescale are the JW32/16 devices, which include 64 bytes of buffer space to share among four endpoints in addition to the control endpoint. The JW Family supports the USB 2.0 full-speed specification and also includes a PS/2 clock generator that can be enabled when not using the USB interface. The JW32 is ideal for high-performance receiver dongle applications in the 27 MHz and 2.4 GHz frequency domains and can also be used as a cost-effective general-purpose USB full-speed device.

For increased speed and a more robust feature set, Freescale's 16-bit microcontroller domain includes the 9S12UF32, which is a USB 2.0 high-speed, full-speed device with many popular built-in memory interfaces and host controllers. It is ideal for all-in-one memory card readers and USB Thumb Drive applications.

Many reference designs, including firmware and Windows demo programs available for Freescale's 8- and 16-bit microcontroller domain, are listed in Table 2. Because Windows drivers for USB applications are application-specific and not microcontroller-specific, Freescale does not include Windows drivers with its reference designs. Generic Windows USB drivers are available on the Internet. A good place to start is **libusb-win32.sourceforge.net**.

Conclusion

As USB becomes more flexible as a communications protocol, microcontrollers with more endpoints, memory and peripherals continue to be developed by Freescale. With the increasing popularity of USB On-The-Go (USB-OTG) and with the development of wireless USB, many more options will become available to USB microcontroller developers and their customers in the near future. What began as a basic improvement of simple PC peripheral connections has evolved into a flexible standard that promises to become even more full-featured in the future.

Christine Peng is a systems and applications engineer in the 8-bit microcontroller division at Freescale Semiconductor. She has a degree in electrical engineering from the University of Texas.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Flash Memory

Secure Your Business

By Meera Balakrishnan

Microcontrollers—An Integral Part of Everyday Life

Look around and you can see microcontrollers everywhere—inside computers, cars, TV sets, mobile phones, MP3 players, washing machines, microwave ovens and phone cards.

Flash in a Pan? Not So

All of these microcontrollers have integrated memory modules, mostly Flash (a programmable, non-volatile memory), which enables field programmability, remote data storage and prototyping. The cost of Flash memory has decreased so much that it is now used in volume production and is not just for prototyping anymore. Flash has many benefits; Flash:

> Is electrically erasable and does not require ultraviolet light

> Provides end-of-line customizing for regional variations in consumer demands

> Provides software-enabled intelligence to address changing legislation

> Supports remote diagnostics and preventative maintenance

> Reduces code obsolescence, which saves on scrapped-product cost

> Standardizes platforms, which reduces product variability

> Eliminates sockets and rework with in-system programmable Flash

> Eliminates the need for external EEPROM by using EEPROM emulation

> Allows cost-effective programming changes and field software upgrades via in-application programmability and re-programmability

> Allows extremely fast programming times of Flash memory; reduces production programming costs through ultra-fast programming

> Allows programming across full operating supply voltage with no extra programming voltage; is useful in cost-effective, reprogrammable battery-powered applications

> Helps reduce time to market with application re-programmability

> Enables the user to define preferred settings with improved write/erase and data retention performance

> Helps to protect code from unauthorized reading and guards against unintentional erasing/writing of user-programmable segments of code with flexible block protection and security

Flash is here to stay.

Need for Security

These days hacking into hardware and software systems can be a full-time profession or hobby for some; therefore, there is an increasing demand for the security of application systems. A typical secure system has different levels of design with security features implemented at each level. The uppermost level is usually related to communications protocols and human interfaces. The applications software level supports the external interfaces and communications protocols and may also perform encryption and user authentication. This may be controlled by the software code program (for example, embedded inside the Flash of a microcontroller), which has built-in security features for authentication, encryption and protection of sensitive information.

Microcontrollers are versatile enough to be used not only for control, but also for protection purposes. Some applications that require secure code in microcontroller Flash memory include bank payment cards, mobile phones, laptops, industrial control applications, wireless communication applications, pay-TV applications and building access control applications. Additionally, secure code in Flash can also be used for blocking non-genuine and refilled cartridges for printers and ensuring servicing of home appliances only by authorized service centers or manufacturers. Security applications increasingly include electronic sensors and locks using Flash microcontrollers.

How Secure Is My Code Inside the Flash Memory?

Given enough time and resources, a determined hacker can break any protection. The overall security of a system is determined by the least secure element. When designing a secure system, a reasonable goal is to make the process of breaking the design expensive and time consuming. One of the first steps in any hardware design is choosing a microcontroller with secure Flash protection features.

Flash Security in Freescale 8-bit Microcontrollers

Freescale Semiconductor is the Flash microcontroller industry leader. Freescale's 8-bit microcontrollers include circuitry to prevent unauthorized access to the contents of Flash and RAM memory, which store the application code. Following is a brief description of some of the salient security features of Freescale's 8-bit microcontrollers.

Engaging/Disengaging Security

Security is engaged or disengaged based on the state of two nonvolatile register bits. Register bits states of 1:0 imply disengaged security, while all other combinations below mean that security is engaged in the microcontroller.

1:1 Engages Security

0:1 Engages Security

0:0 Engages Security

When security is engaged, Flash and RAM are considered secure resources. Some registers within the microcontroller (i.e., direct-page registers, high-page registers) and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any memory location and resources within the microcontroller. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked. That is, write commands are ignored and read commands return all 0s. The on-chip debug module is designed so that it cannot be enabled while the MCU is in "secure" mode. The separate background debug controller can still be used for background memory access commands.

How Flash-Protected Blocks Add to Programming Safety

Block protection prevents the protected Flash region from accidental program or erase changes. Block protection is controlled through the Flash protection register.

In-application Flash programmability does not need two non-volatile elements. If the programming algorithm is contained in Flash, part of the algorithm could be to copy the algorithm itself into RAM. The programming algorithm could then be executed from RAM. This is a reliable method if the microcontroller can execute from internal RAM. But, in the event of a power brown-out or a complete power failure, the internal RAM will be completely erased. Once the programming algorithm operating from RAM has erased the Flash module, the window of liability is open until the programming of the Flash is complete.

Write-Protected Flash Blocks

The previous scenario of RAM corruption due to power supply problems can be overcome by having protected blocks in the Flash memory. You achieve this with a small write-protected block within the Flash array itself. This protected block can contain the programming algorithm for the microcontroller. Write-protection is provided by requiring an external high voltage source for this particular block. If the external high voltage is unavailable, it will not be possible to erase the protected block.

In the actual application, the high voltage would probably be absent. The remainder of the Flash module can use an internal charge pump to generate the required high voltage necessary for programming and erasing the Flash.

One use for block protection is to block protect an area of Flash memory for a boot loader program. This boot loader program can be used to erase the rest of the Flash memory and reprogram it. Because the boot loader is protected, it remains intact even if MCU power is lost in the middle of an erase and reprograms operation.

Vector Redirection

Whenever any block protection is enabled, the reset and interrupt vectors in the microcontroller will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. It also allows the user to reprogram the unprotected portion of the Flash with new program code, including new interrupt vector values, leaving the protected area unchanged.

Block protects cannot be changed from user application programs. If the vector space is block protected, the backdoor security key mechanism (see the following section for more information) cannot permanently change the block protect, security settings or the backdoor key.

Protection Violation Flag

A protection violation flag bit is automatically set to 1 so as to register a command that attempts to erase or program a location in a protected block. (The erroneous command is ignored.)

Backdoor Security Key

A user can choose to allow or disallow a security unlocking mechanism through an 8-byte backdoor security key. The backdoor comparison key is located in Flash memory locations in the nonvolatile register space so users can program these locations exactly as they would program any other Flash memory location. The nonvolatile registers are in the same block of Flash as the reset and interrupt vectors, so block protecting that space also block protects the backdoor comparison key.

If security needs to be unlocked, the user inputs the key through an external interface, such as a keypad. The input key is compared to the key programmed or stored in the Flash memory. If the input key is same as the stored key, then security is disabled.

The backdoor key can be disabled through setting an appropriate bit to 0. When this is finished, there is no way to disengage security without completely erasing all Flash locations. There are, however, other ways of temporarily disengaging security. Security can always be disengaged through the background debug.

Conclusion

Freescale's wide range of Flash-based, 8-bit microcontrollers (MC68HC908 and MC9S08 Families) have security features that lend protection for your code to help make them secure from tampering. Choose from a range of high performance devices with Flash memory, from 1 KB to 128 KB and pin counts from 8 to 80. Visit **www.freescale.com/8bit** for more information.

Meera Balakrishnan is a product marketing manager at Freescale.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Using Fail-Safe Modes for Proper Code Operation

By Harald Kreidl

This article describes two fail-safe modes—brown-out detection and watchdog timer—that Freescale has implemented in its 8-bit microcontroller HCS08 to ensure safer hardware and code operation.

Brown-Out Detection

Introduction

Microcontrollers are quite complex CMOS circuits that need at least two electrical parameters to work properly: a supply voltage (to source current) and a clock supply. The supply voltage is fed from an external power-supply circuit to the microcontroller either directly to the internal CMOS logic (as on HC08) or via an internal on-chip voltage regulator (as on HCS08). Due to several reasons, the power-supply voltage cannot always be kept stable in a real application. For example, batteries become discharged and the voltage drops; the switching of loads causes the system to withdraw so that the voltage decreases; and overload situations and malfunctions cause voltage variations. When the voltage drops below a certain level, the microcontroller cannot work properly. This condition is called brown-out (see Figure 1) and brown-out conditions can result in malfunction of the microcontroller.

During a brown-out situation, the CPU can fetch the random value of addresses into the program counter and cause code runaways. The CPU can execute Flash erase or program routines (of the monitor ROM in HC08s or in Flash programming routines placed in user space of Flash memory) by accident and change the contents of the Flash memory. Outputs can change their state unintentionally and cause damage of external power stages. These dangerous situations must be avoided through external or internal low-voltage recognition. If brown-out situations are recognized, the microcontroller should be notified and set in reset condition.

Recognition of Brown-Out Situations

In the past, brown-out situations were detected by external circuits. Now there are many different discrete and

FIGURE 1. BROWN-OUT SITUATION

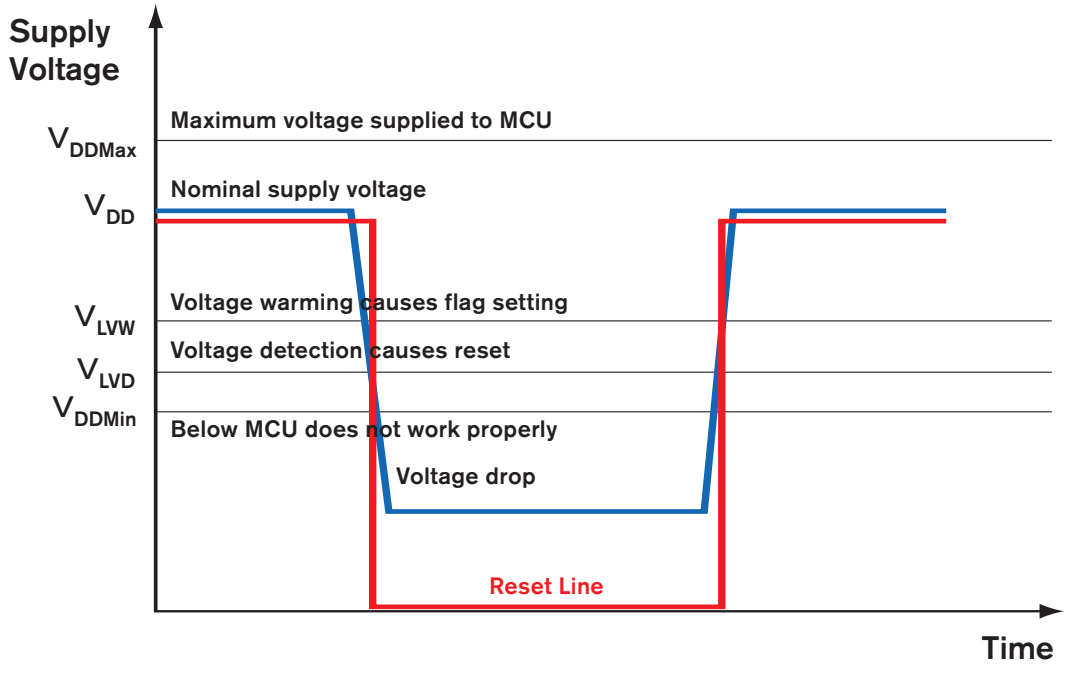
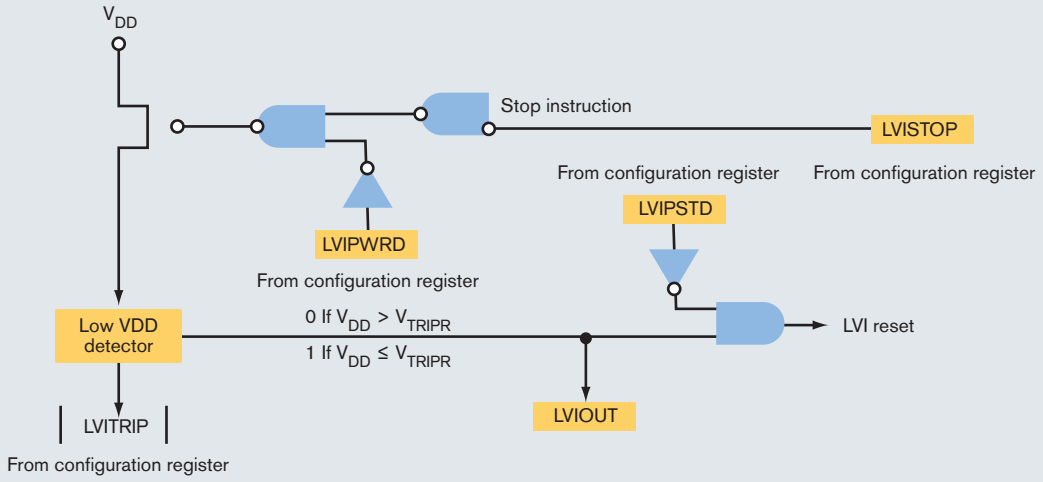


FIGURE 2. LVI BLOCK MODULE DIAGRAM



integrated solutions that differ in effort, cost and level of feature integration. Two examples are dedicated circuits for low-power operation that use three transistors and cost-effective circuits that use only one transistor and integrated solutions. These circuit types monitor the power-supply voltage and draw the RESET input low when the supply voltage falls below the threshold voltage. In order to save system resources and reduce space on the printed circuit board (PCB), Freescale has integrated low-voltage interrupt (LVI) on all 8-bit HCS08 microcontrollers.

LVI on HC08

On HC08, the brown-out detection circuit is an individual module that can be configured with bits in the CONFIG register. The LVI module block diagram is shown in Figure 2.

The software development engineer has the following LVI configuration options:

1. **Enabling or disabling LVI during operation of the microcontroller**
   After reset, the LVI is powered on, and the module is enabled. To disable the LVI module, set the LVIPWRD bit in CONFIG1 register. Additionally, the LVIRSTD bit in the CONFIG1 register must be cleared to enable the reset generation after recognition of a brown-out condition. (This is the case after reset.)
2. **Setting the trip point voltage**
   The standard power supply range of most HCS08 is 2.7V to 5.5V. Therefore, two trip point voltages can be selected by LVITRIP bit in the CONFIG1 register. The bit is 0 (3V operation) after power on reset and is unaffected after another reset cause; a 1 in LVITRIP appoints 5V operation.
3. **Enabling or disabling LVI in STOP mode**
   LVI is disabled in STOP mode after reset and must be enabled by setting the LVISTOP bit in CONFIG1 register.

Figure 3 shows the CONFIG1 register of the MC908QBxx.

FIGURE 3. CONFIG1 REGISTER OF THE MC908QBxx

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	CORPS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD
Reset:	0	0	0	0	U	0	0	0
POR:	0	0	0	0	0	0	0	0

U=Unaffected

Configuration of the LVI

LVI configuration is performed in the CONFIG1 register, which is written once after a reset. If written once, no content change is possible until the next reset. This impacts initialization procedures. All bits in the CONFIG1 register must be written with a single command (i.e., LDA CONFIG1; STA CONFIG1 or MOV x,CONFIG1). Bit set and bit clear commands cannot be used. In C, it is important that the assignment is done so that the C compiler generates a single assignment instruction. See Sample 1.

SAMPLE 1.

```
CONFIG1 = 0b10001000;       // Write once!
 //      ||||||||
 //      |||||||+- COPD: COP Disable Bit, 0 = COP module enabled
 //      ||||||+-- BUSCLKX4: cycle delay, 0 = long
 //      |||||+--- SSREC: Short Stop Recovery Bit
 //      ||||+---- LVITRIP: LVI Trip Point Selection Bit
 //      |||+----- LVIPWRD: LVI Power Disable Bit
 //      ||+------ LVIRSTD: disables the reset signal from LVI
 //      |+------- LVISTOP: LVI Enable in Stop Mode Bit
 //      +-------- COPRS: COP Reset Period Selection Bit
Alternative Writing:
    CONFIG1 =  CONFIG1_LVITRIP_MASK+ // 5 volt operation
               CONFIG1_COPRS_MASK;   // Short WD time
Is assembled in:
     MOV   #$88,_CONFIG1
Do not:
     BSET  7,_CONFIG1
     BSET  4,_CONFIG1
```

The configuration should be done immediately after the reset. Typically, the software engineer starts the code in main(), but the C compiler uses start-up routines to initialize the variables, set the stack pointer and perform other functions. If CodeWarrior™ is used, the start-up routine is placed in Start08.c. The CONFIG1 assignment can be done as the first instruction after reset (reset vector points to Startup(void)!). See Sample 2.

SAMPLE 2.

```
#include "derivative.h"  /* include peripheral declarations */
__EXTERN_C void _Startup(void) {
/* set the reset vector to _Startup in the linker parameter file
(*.prm): 'VECTOR 0
_Startup'
    purpose:    1)  initialize the stack
                2)  initialize run-time, ...
                    initialize the RAM, copy down init data, etc (Init)
                3)  call main;
    called from: _PRESTART-code generated by the Linker      */
    CONFIG1 = CONFIG1_LVITRIP_MASK+ // 5 volt operation
              CONFIG1_COPRS_MASK;   // Short WD time
    INIT_SP_FROM_STARTUP_DESC();
    Init();
    __asm JMP main; /* with a C style main();
                       we would push the return address
                       on the stack wasting 2 RAM bytes         */
}
```

For this reason, the CONFIG1 assignment should be placed in the start-up routine of the C compiler.

LVI in Power-Saving Modes

The initial purpose of the brown-out detection (by the LVI module) is to prevent the malfunction of the microcontroller when the voltage drops to a predetermined low level. Code runaways and indefinite I/O conditions are prevented by using LVI. The HC08 offers two power-saving modes: WAIT and STOP.

In WAIT mode, the clock is disconnected from the CPU. Peripherals can work selectively in power-saving mode. After an event is generated from an internal peripheral or external hardware, the CPU is alerted, and an interrupt service routine is executed. It is recommended that LVI be enabled if WAIT mode is used (i.e., to wait for the analog to digital conversion or to send or receive a byte with SCI, and more) because after wake-up, the CPU executes code.

In STOP mode, the clock system is powered down, and the whole microcontroller goes into sleep mode. No code

FIGURE 4. SIM RESET STATUS REGISTER (SRSR)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
Write:								
POR:	1	0	0	0	0	0	0	0

=Unimplemented

SAMPLE 3.

```
__EXTERN_C void _Startup(void)
{  ...
   switch( SRSR )
     {
        /* External reset was cause of reset */
        case SRSR_PIN_MASK:   // Do stuff here
           break;
        /* Watchdog was cause of reset */
        case SRSR_COP_MASK:   // Do stuff here
           break;
        /* Illegal opcode was cause of reset */
        case SRSR_ILOP_MASK:   // Do stuff here
           break;
        /* Illegal address was cause of reset */
        case SRSR_ILAD_MASK:   // Do stuff here
           break;
        /* Forced monitor mode entry was cause of reset   */
        case SRSR_MODRST_MASK:   // Do stuff here
           break;
        /* Low VDD was cause of reset */
        case SRSR_LVI_MASK:
           DDRA = 0b00000001;      // Pin output
           PTA  = 0b00000001;      // Voltage alarm LED on
           break;
        /* Power on was cause of reset */
        case SRSR_POR_MASK:
           break;
        /* Should not occur */
        default:   // Do stuff here
           break;
     }
   ...
   __asm JMP main;
}
```

is executed, and the only way to wake up the controller is through an external interrupt, such as a reset signal, IRQ pin or KBI pins. Because code can no longer be executed, the initial purpose of the brown-out detection no longer exists; therefore, LVI should be disabled in STOP. Depending on the microcontroller derivative, it is possible to keep part of the oscillator running in STOP mode. Other wake-up sources, such as auto wake-up timer or a real-time interrupt (RTI) counter, are available. In STOP mode, the current consumption is in the single digit micro ampere range. LVI adds about 100 µA to this; therefore, LVI is not often used in combination with STOP mode. LVISTOP flag in CONFIG1 must be cleared (default after reset) in order to disable the LVI module in STOP mode.

Reaction to Brown-Out Detection

Once an under voltage (VDD voltage falls to the LVI trip voltage VTRIPF) is detected, the CPU generates an internal reset signal, fetches the reset vector and starts code execution at _Startup(void). The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin (RST) is held low while the SIM counter counts out 4,096 oscillator cycles after VDD rises above VTRIPR. Sixty-four oscillator cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur. For exact timing of the reset procedure, look at the data sheet of the HC08 because it differs from derivative to derivative. Figure 4 shows the SIM reset status register (SRSR) where software code can check the cause of the reset.

In the start-up routine, the software should check the reset cause and then act accordingly. Because a reset sets all configuration registers to their default state, hardware initialization must be done after an LVI reset. The reset status will dictate which actions to take, i.e., for LVI, a pin is set that might be connected to an alarm LED. The system designer has to define the activities to be performed after an error reset. The Sample 3 code snippet shows how to handle the reset status with that in mind.

The code snippet in Sample 4 shows another way to handle the reset status. Here, the reset status is saved in a

SAMPLE 4.

```
unsigned char reset_status;

__EXTERN_C void _Startup(void)
{  ...
   ...
  reset_status = SRSR;
  // Memorize the reset status to act later
   ...
  __asm JMP main
}
```

FIGURE 5. EXAMPLE OF THE 5-VOLT SPECIFICATION

Low-voltage inhibit reset, trip failing voltage	V_{TRIPF}	3.90	4.20	4.50	V
Low-voltage inhibit reset, trip rising voltage	V_{TRIPR}	4.00	4.30	4.60	V
Low-voltage inhibit reset/recover hysteresis	V_{HYS}	—	100	—	mV

variable and analyzed later in the software. The contents of SIM reset status register (SRSR) is cleared after read.

LVI Specification

An example of the 5V specification (of the 908QB8) is shown in Figure 5.

Two values are specified: VTRIPF is the value for the falling trip voltage. At this voltage, a reset is generated when the power supply voltage drops. In this example, this can happen between 3.90 and 4.50V. The hardware engineer must ensure that in normal operation mode, VDD is always > 4.50V (in order to avoid an LVI reset). Once a reset is generated and VDD does not reach the power-on reset (POR) rearm voltage (which for the 908QB8 is 750 mV), then VTRIPR is the voltage where reset is released if VDD rises to this level. For this device, VTRIPR is 4 to 4.60V. If VDD drops below POR rearm voltage, the SIM module interprets the following rising of VDD as a power-on reset. Both flags of SRSR are then set: POR and LVI; hysteresis occurs in the behavior of the LVI module to avoid oscillation.

A 3V operation is similar but at lower voltages; Figure 6 shows the specified values for the 908QB8 at that voltage.

The operating voltage range of the 908QB8 is 2.7 to 5.5V, but the VTRIPF is in the range of 2.40 to 2.70V. In the worst case, a voltage range of 2.40 to 2.70V does not guarantee operation. It is possible for the VDD to drop and the LVI generates only a reset at 2.4V. From the time when VDD falls below 2.70V and the LVI generates a reset at 2.40V, the operation of the microcontroller is not guaranteed.

LVD on HCS08

The LVI module on HCS08 was modified (compared to HC08) and therefore renamed low-voltage detection (LVD). The LVD system prevents low-voltage conditions by protecting memory contents and controlling MCU system states during supply-voltage variations. The system is comprised of a power-on reset (POR) circuit and an LVD-circuit with user selectable trip voltage, either high (VLVDH) or low (VLVDL). The LVD circuit is enabled when the LVDE bit in the SPMSC1 register is high and the trip voltage is selected by the LVDV bit in the SPMSC3 register. The LVD is disabled upon entering any of the stop modes unless the LVDSE bit is set in the SPMSC1 register. If LVDSE and LVDE are both set, the MCU cannot enter STOP1 or STOP2, and the current consumption in STOP3 with the LVD enabled will be greater.

The LVD system can generate either an interrupt or a reset when the supply voltage drops below the LVD voltage. The interrupt priority of the LVD interrupt is quite high and is defined by the sequence of the interrupt vector. This is shown in Table 1.

The behavior of the power-on operation is also different from that of the HC08s. When power is initially applied to the MCU or when the supply voltage drops below the VPOR level, the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the MCU in reset until the supply has risen above the VLVDL level. Both the POR bit and the LVD bit in the SRS register are set after a POR.

LVD Reset Operation

The LVD can be configured to generate a reset upon detection of a low-voltage condition by setting LVDRE bit to 1. After an LVD reset has occurred, the LVD system will hold the MCU in reset until the supply voltage has risen above the level determined by LVDV. The LVD bit in the SRS register is set following an LVD reset or a POR.

FIGURE 6. SPECIFIED VALUES FOR 908QB8

Low-voltage inhibit reset, trip failing voltage	V_{TRIPF}	2.40	2.55	2.70	V
Low-voltage inhibit reset, trip rising voltage[6]	V_{TRIPR}	2.475	2.625	2.775	V
Low-voltage inhibit reset/recover hysteresis	V_{HYS}	—	75	—	mV

TABLE 1. INTERRUPT PRIORITY OF THE LVD

Vector Address (LOW:HIGH)	Vector	Source Flag	Enable Flag	Priority
0xFFFE:FFFF	Reset	COP LVD Reset pin Illegal opcode Illegal address POR	COPE LVDRE RSTPE - - -	Highest priority
0xFFFC:FFFD	SWI	SWI instruction	-	2nd highest priority
0xFFFA:FFFB	IRQ	IRQF	IRQIE	3rd highest priority
0xFFF8:FFF9	Low Voltage Detect	LVDF	LVDIE	4th highest priority
	…	…		…

LVD Interrupt Operation

When a low-voltage condition is detected and the LVD circuit is configured using SPMSC1 for interrupt operation (LVDE set, LVDIE set and LVDRE clear), LVDF in SPMSC1 will be set and an LVD interrupt request will occur.

Low-Voltage Warning (LVW)

The LVD system has a low-voltage warning flag to indicate to the user that the supply voltage is approaching—but is above—the LVD voltage. The LVW does not have an interrupt associated with it. There are two user selectable trip voltages for the LVW, one high (VLVWH) and one low (VLVWL). The trip voltage is selected by LVWV bit in SPMSC3 register.

LVD in STOP Mode

The LVD system can generate either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 are both set) at the time the CPU executes a STOP instruction, the voltage regulator remains active during stop mode. For the ADC to operate, the LVD must be left enabled when entering STOP3.

LVD Configuration

The LVD configuration on HCS08 is much more complex than that found on HC08 because of HCS08's diverse STOP modes, LVD interrupt capability, reset operation and the LVW low-voltage warning. The software development engineer has to follow these LVD configuration options:

Enabling or Disabling LVD During Microcontroller Operation

After reset, the LVD is powered on and the module is enabled. If you want to disable the LVI module, clear the LVDE bit in the SPMSC1 register. Additionally the LVDRE bit in the SPMSC1 register must be cleared in order to disable the reset generation after the recognition of a brown-out condition. (After reset, this bit is 1 and reset generation is enabled.) If an interrupt is generated instead

FIGURE 7. SYSTEM POWER MANAGEMENT STATUS AND CONTROL REGISTER 1 (SPMSC1)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LVDF	0	LVDIE	LVDRE[2]	LVDSE	LVDE[2]	0	BGBE
Write:		LVDACK						
POR:	0	0	0	1	1	1	0	0

=Unimplemented or Reserved

1 Bit 1 is a reserved bit that must always be written to 0.

2 This bit can be written only one time after reset. Additional writes are ignored.

FIGURE 8. POWER MANAGEMENT STATUS AND CONTROL 3

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LVWF	0	LVDV	LVWV	0	0	0	0
Write:		LVWACK						
POR:	0[1]	0	0	0	0	0	0	0
LVR:	0[1]	0	U	U	0	0	0	0
Any other reset:	0[1]	0	U	U	0	0	0	0
		=Unimplemented or Reserved			U=Unaffected by reset			

1 LVWF will be set in the case when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW}.

of a reset, then LVDIE must be set to 1. Avoid setting LVDIE and LVDRE at the same time.

Setting the Trip Point Voltage

The standard power supply ranges of HCS08 are 1.8 to 3.6V for low-voltage devices and 2.7 to 5.5V for industrial and automotive parts. Consequently, there are two trip-point voltages for each range that can be selected by LVDV bit in SPMSC3 register. The bit is 0 for low trip point voltage, which is the case after POR and is unaffected after another reset cause. A 1 in LVDV appoints high trip point voltage.

Enabling or Disabling LVD in STOP Mode

LVD is enabled in STOP mode after reset and has to be disabled by clearing the LVDSE bit in the SPMSC1 register.

Figure 7 shows the system power management status and Control 1 (SPMSC1) register of the MC9S08QGx.

Figure 8 shows the system power management status and Control 3 (SPMSC3) register of the MC9S08QGx.

In registers SPMSC1 and SPMSC2, flags and configuration bits are in one register. Avoid writing a 1 into the flags LVDACK and LVWACK during the initialization because this can reset pending LVDF and LVWF flags. Writing a zero (!SPMSC1_LVDF_MASK) into the flags is not necessary in the Sample 5 code, but it is better for program documentation.

SAMPLE 5.

```
SPMSC1  = !SPMSC1_LVDF_MASK+    // 0 LVDF flag is read only
          !SPMSC1_LVDACK_MASK+  // 0 LVDACK flag to clear LVDF
          SPMSC1_LVDIE_MASK+    // 1 LVD interrupt enable
          !SPMSC1_LVDRE_MASK+   // 0 LVD reset disable
          !SPMSC1_LVDSE_MASK+   // 0 LVD disabled in STOP
          SPMSC1_LVDE_MASK+             // 1 LVD enabled
          SPMSC1_BGBE_MASK;     // 1 Bandgap buffer enabled

SPMSC3  = !SPMSC3_LVWF_MASK+    // 0 LVWF flag is read only
          !SPMSC3_LVWACK_MASK+  // 0 LVWACK flag to clear LVWF
          SPMSC3_LVDV_MASK+     // 1 High tripp. volt. for detection
          SPMSC3_LVWV_MASK;     // 1 High tripp voltage for warning

Is assembled in:
    LDA  #37
    STA  _SPMSC1
    LDA  #48
    STA  _SPMSC3
Is same as:
    MOV  #37,_SPMSC1
    MOV  #48,_SPMSC3
Do not:
    BCLR 3,_SPMSC1
    BCLR 4,_SPMSC11
```

Reaction of Brown-out Detection

The system reset status (SRS) register shows the cause of the reset. A write to this register clears the watchdog counter (resets watchdog) without changing any value of the register. This is different from the SPSR register found in the HC08, as this register is not cleared by any read access. There is no need to save the contents of this register in a variable.

After a power-on reset, LVI and POR bits are set, and if the reset was forced by the debugger, the SRS register is 0 (no bit is set). The software designer can select interrupt generation instead of reset. After low-voltage detection, the CPU enters the LVD interrupt service routine where the software can do last activities (save conditions to Flash, switch actuators off or on, and more) and then loop

FIGURE 9. SYSTEMS RESET STATUS

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POR	PIN	COP	ILOP	ILAD	0	LVD	0
Write:	Writing any value to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVR:	U[1]	0	0	0	0	0	1	0
Any other reset:	0	Note[2]	Note[2]	Note[2]	Note[2]	0	0	0

1 U=Unaffected

2 Any of these reset sources that are active at the time of reset entry will cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset entry will be cleared.

without triggering the watchdog. A watchdog reset will then occur. In this routine, the user can also acknowledge the low-voltage warning flag (LVWF) that should be set at any LVD event. If this flag is reset, it indicates the low-voltage condition is still valid.

In addition to what is found in the HC08, the HCS08 has a low-voltage warning that indicates to the software that the supply voltage is dropping. This warning flag does not generate an interrupt and must be regularly polled from the software either in the main loop or in a timer interrupt routine.

Watchdog

A watchdog or COP (computer operates properly) is an important safety module and is available in every microcontroller system today. The watchdog wants to be fed periodically; otherwise, it strikes out and resets the microcontroller. In control software code, very often a flag has to be checked and program execution has to wait, i.e., polling the lock bit of a PLL shows that the PLL has locked. If the lock bit is never set, then CPU stays in an endless loop. It can also happen that through software mistakes, the software also can end up in an endless loop. This blocks the system and makes the application inoperable. If the application program gets lost and fails to reset the watchdog counter before it times out, a system reset is generated to force the system back to a known starting point.

The watchdog is a counter that can be reset to start counting from zero when software writes values to a specific location. If the counter overflows, it generates a reset (if enabled). The implementation of the watchdog is different in some devices. The standard HC08 uses the internal bus clock as a clock source. There are also devices that use an extra RC oscillator that should provide some extra safety. Even when the internal bus is no longer working (due to a PLL disturbance), the RC oscillator feeds the watchdog, and the watchdog resets the part. There are also applications where an external watchdog is required, especially in safety critical and automotive applications.

Configuration of the Watchdog

The watchdog is always enabled after reset, and it can be disabled in the CONFIG1 register in HC08 (see Figure 3) and in the SOPT1 register in HCS08 (see Figure 10). There are two timeout periods available that are also selected in CONFIG1 register for HC08 and SOPT1 register for HCS08. The default time after reset is set to long time for both families. No other specific action is required for watchdog configuration on HC08.

The COPCLKS bit in the SOPT2 register information selects the clock source used for the COP timer in HCS08s. The clock source options include the bus clock and an internal 1 kHz clock source. With each clock source, there is an associated short and long time-out controlled by COPT in SOPT1. The COP watchdog defaults to operation from the 1 kHz clock source and the associated long time-out that gives a trigger period of 256 ms ± 30 percent. The software must be designed for triggering after roughly 170 ms.

FIGURE 10. SOPT1 REGISTER IN HCS08

	Bit 7	6	5	4[1]	3	2	1	Bit 0
Read:	COPE	COPT	STOPE		0	0	BKGDPE	RSTPE
Write:								
Reset:	1	1	0	1	0	0	1	U[2]
POR:	1	1	0	1	0	0	1	0
LVR:	1	1	0	1	0	0	1	0
		=Unimplemented or Reserved						

1 Bit 4 is reserved, writes will change the value but will have no affect on this MCU.

2 U=Unaffected

Watchdog Operation

Software must regularly trigger the watchdog by writing to the SRS register (HCS08) or to address $FFFF (HC08). The contents of SRS or $FFFF are not changed by writing. The value written to the address does not matter. Some examples that trigger the watchdog are shown in Sample 6.

Some attention must be paid to the triggering of the watchdog on HC08. The address $FFFF is the location of the reset vector. If Flash programming is in process (i.e., EEPROM emulation), a write to $FFFF changes the address latch in the Flash programming state machine and that can erase the page located at $FFFF, resulting in an erasure of the interrupt table; therefore, either watchdog must be disabled in the case of Flash programming. Flash programming or triggering the watchdog during Flash programming must be done with care. It is also a good practice to service the watchdog immediately after reset and before entering or after exiting stop mode to guarantee the maximum time before the first watchdog counter overflow.

The watchdog triggering (write to SRS or to $FFFF that services [clears] the COP counter) must not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails. In background debug mode (HCS08) or in monitor mode (HC08, only when VTST is present on the IRQ pin.), the COP counter will not increment; it is disabled until exiting these debug modes. When the bus clock source is selected, the COP counter does not increment while the system is in STOP mode because there is no bus clock. The COP counter resumes as soon as the MCU exits STOP mode. When the 1 kHz clock source is selected, the COP counter is re-initialized to zero upon entry to STOP mode, but it continues to count and will reset the part when not triggered in time.

If STOP3 is used and RTI is used to exit STOP3, the user should not use the 1 kHz oscillator as a watchdog for the clock source. The maximum RTI time can be up to one second, whereas the trigger time for the watchdog is about 250 ms—that means the watchdog resets the part before RTI can wake up the part from STOP3. The COP counter begins from zero after the MCU exits stop mode.

SAMPLE 6.

```
In C language:
#define __RESET_WATCHDOG() {asm sta SRS;}  // Makro to trigger watchdog
__RESET_WATCHDOG();       // Trigger watchdog
Alternative (if inline assembly may not be used):
SRS = 0;                  // Trigger watchdog
In assembly language:
    STA   SRS       // Trigger watchdog (HCS08)
    STA   $FFFF     // Trigger watchdog (HC08)
```

Harald Kreidl is a senior engineer at Freescale and has been with the company for eight years. He holds degrees in physics (Dipl.-Ing FH) from Fachhochschule Munich and applied electronics from Technical University Munich. He has co-authored several books about microcontrollers.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

On-Chip Debug—Here and Now

The Best of Yesterday's Debug Monitors and In-Circuit Emulators for Less

By Kenny Lam

Introduction

The development support hardware inside the HCS08 Family consists of the background debug controller (BDC) and the on-chip debug module (DBG). Development can be done in the actual system. Other advantages include:

> Real-time bus capture function

> Simple setup

> Cost effectiveness

This article discusses the user benefits of this system. While similar modules are also available on other Freescale MCU series like the 16-bit HC12 and S12 series, this article focuses only on the HCS08 MCU.

Benefits of On-Chip Emulation

Many traditional emulators rely on external circuitry to access the internal bus of the target MCU. This enables users to obtain useful data for problem analyzing or software debugging during development. Usually, the target MCU will run in special mode where some of its I/O pins become the address or data bus for capturing the required data. The original functions of the I/O pins will be lost. Those pin functions must be rebuilt somewhere in the emulator board. Apart from users paying for a more expensive emulator system, these rebuilt pins may not be exactly the same as the original. For example, the driving characteristics—the ADC functions—may be different. Also, there are usually a number of connection cables between the target board and debugger system that are inconvenient in many situations. Moreover, as the speed of microcontroller devices continues to increase, it becomes more difficult to build such emulators with long connection cables.

With the continuous advancement in the semiconductor wafer process, the cost of more logic circuitry inside a die may be even lower than the cost to build additional bonding pads for internal bus lines. Building on-chip emulator circuitry inside MCU die becomes a more cost-effective approach.

On-Chip Emulation Hardware Requirements

The overall debugging setup consists of the target system, a host PC and a multilink interface that connect the PC and the target board. For instance, there is a USB version BDM Multilink. It is small and inexpensive. The Freescale CodeWarrior™ tool suite can be installed in the host PC to control the BDM Multilink. It provides a user-friendly debugger environment to control the BDC or to configure DBG setting. In addition, an external host can also use the BDC to program the on-chip Flash of the target MCU.

Background Debugger Controller

In the HCS08 microcontrollers, the communication between the target MCU and external debugger system is sent through a bidirectional single-wire background debug interface. This is the BKGD pin of the BDC. Users only need to reserve a small PCB space for a pre-defined connector in the actual application board. It is a 6-pin connector that includes BKGD, RESET, VDD, GND; the remaining two pins are NC for foolproof connectivity. It is connected to the external system during the debugging process. The debugging runs on the actual application system without rebuilding any MCU I/O on certain emulator boards.

BDC, designed to support the background debug mode (BDM), allows the external system to read the MCU's internal memory or enables a single-step user instruction tracing. It is able to access the MCU's internal memory even if the application program is in execution. It can be used to access the control registers and internal buffers of the DBG during debugging. (The DBG will be discussed in the next section.)

The control registers of BDC are not located in the user memory map. They can only be accessed through the BKGD pin. This avoids the possibility of these registers being written accidentally by any user program bugs. The communication is based on a custom protocol. The communication speed can be the CPU bus rate or a special BDC clock rate. It does not require the CPU running at certain particular frequencies; the MCU can run at any frequency according to the actual application needs.

On-Chip Debugger Module

The on-chip debugger module is an in-circuit emulator that is built into the HCS08 MCUs. This module provides the bus state analysis function where it replaces an expensive external data-bus analyzing tool and is capable of capturing real-time bus information without stopping the application program. There are FIFO (8 buffers) and comparators (A and B) inside DBG. The bus data are captured in the internal FIFO and can be retrieved to the external host via the BKGD pin of BDC. The "change-of-flow" information is very useful in emulators. The flexibility for users to capture bus information in different situations is also important during debugging.

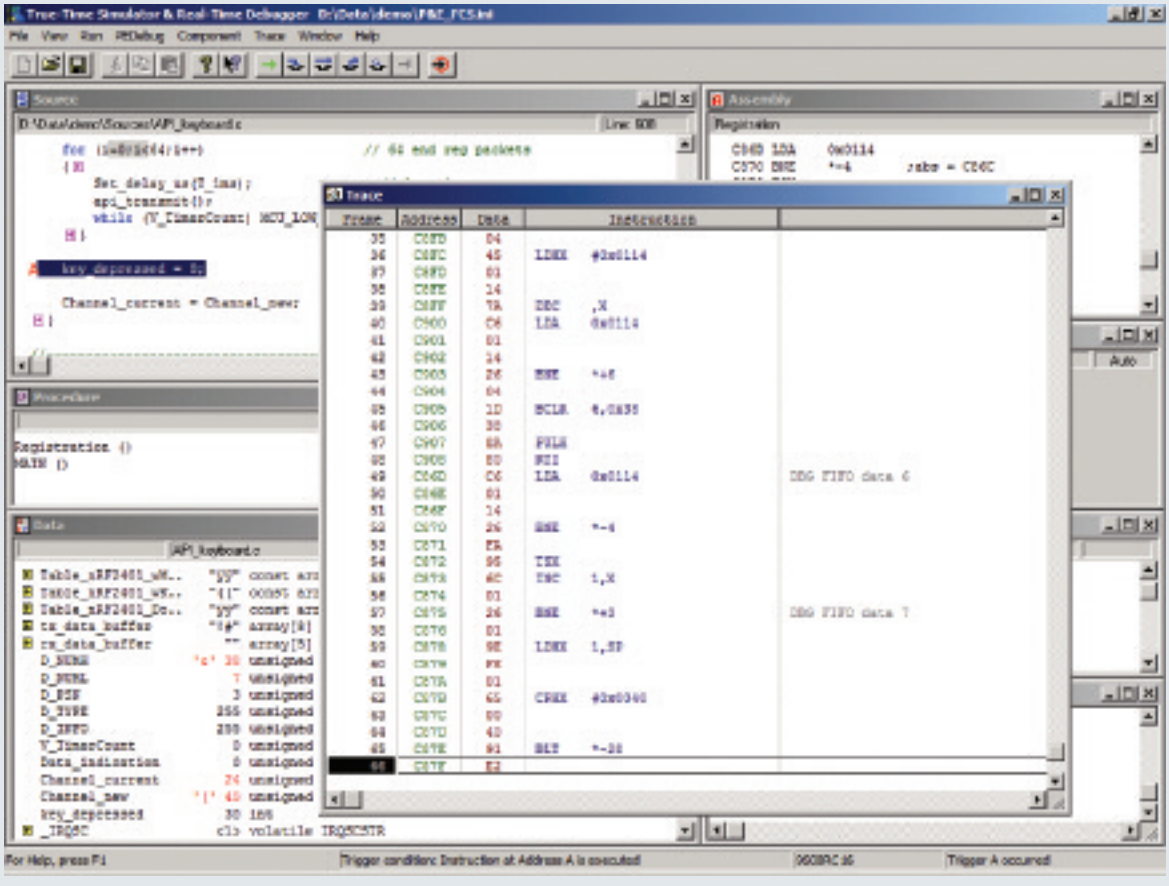
Change-of-Flow Information

One piece of bus information that DBG can be configured to capture is the "change-of-flow" information. This is the information of the recent executed user program instructions where a program sequence change was made. The external host debugger will know the source and object code of the target MCU program, and the external debugger can use such change-of-flow data to reconstruct the execution path.

Figure 1 depicts a CodeWarrior debugger screen on an external host PC. The screen shows different windows where a pre-defined trigger has taken place. The "Source" window shows the source code of the application program running in a target MCU system. In this example, a trigger has been set at an instruction location and the program will halt when that instruction is executed. A red letter "A" is behind that instruction indicating that the comparator A is associated for the trigger. Note: There are actually other setting options so the program can continue capture data even after reaching the trigger point.

A "Trace" window can be opened by selecting the "Bus Trace" from the "View" item of the toolbar. After the trigger has been performed, the "Trace" window shows the reconstructed path of executions. In the right-most column ("FIFO analyze remark"), you see "DBG FIFO data x."

FIGURE 1. "CHANGE-OF-FLOW" INFORMATION IN THE DEBUGGER "TRACE" WINDOWS



They indicate what instruction information the FIFO buffers have stored. It is able to capture up to eight change-of-flow information instructions. At the left hand-side of each FIFO data is the change-of-flow instruction. Usually, these instructions are conditional branch, return from a function call, or start of an interrupt service routine. The "Instruction" column shows what instructions the MCU has executed before the trigger. It tells users what operation the MCU has done before reaching the trigger point and can be very useful for software debugging. The conventional breakpoint may not provide such information. It can show an interrupt that has taken place before the trigger, or which instructions lead the program jump into that particular instruction location.

Note: In the "Trace" windows, by pressing the mouse's left button and scrolling up or down, different instructions can be highlighted. The corresponding instructions in the source windows will also be highlighted. It is convenient for users to trace the exact executed codes from the original source file.

Bus Information Capture

In different situations, users may need different types of control for data capturing, such as when or what information to capture. The DBG is designed to offer such flexibility to the users. The comparators and control registers built inside the DBG provide different triggering options.

There are two 16-bit comparators named A and B inside the DBG hardware. They can be used to compare the CPU address for determining when to perform a trigger. Comparator B can also be used to compare the 8-bit CPU data bus address in some settings.

For example, the capture can be configured to trigger if an instruction in one specified location is executed. This uses only comparator A (the "A-only" trigger mode). In another case, it can cause the trigger to happen when the instruction at the second specified location is executed, but only if another instruction at the first specified location has been executed before. This uses both comparator A and B (the "A then B" trigger mode).

In addition to based-on instruction addresses, the capture can be configured to perform, or trigger, based on memory locations of user-defined variables or peripheral module registers. It can set the capture to occur if any READ or WRITE access that has occurred to the specified memory locations. The comparators A or B will then be used for comparing the memory addresses. For example, the capture can be configured to perform when any one of two specified memory locations has been accessed, or access to any memory within a certain address range (i.e., start and end address of a data array).

It is even possible to trigger the capture only when the content of a memory location is equal to a specified value, or to configure to trigger if it is not equal to that value. The comparator A can be used for the address comparison while comparator B will be used for data bus comparison.

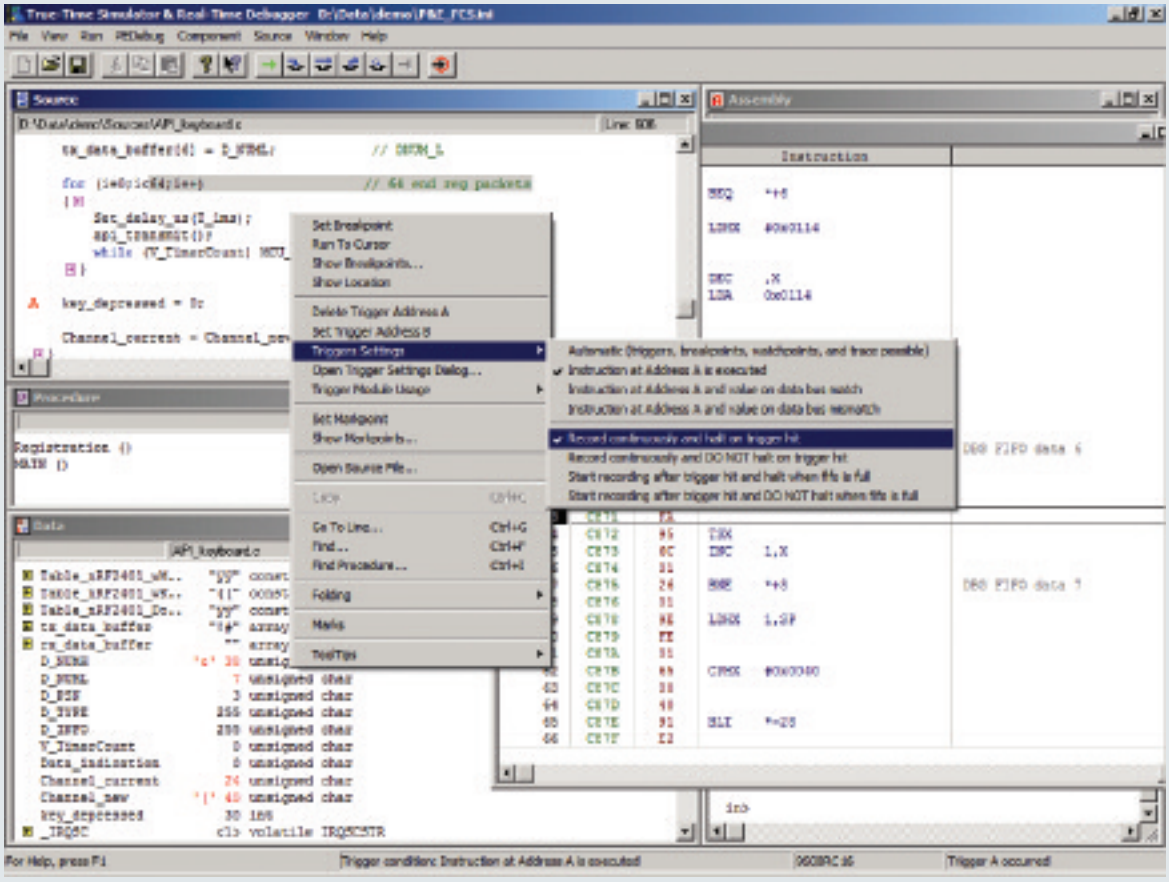
The availability of such bus capture trigger options is very useful to users. They can save much debugging time. For example, it is easier to investigate why a certain variable content is corrupted, or when a peripheral register is overwritten accidentally.

Apart from the change-of-flow information, in certain cases the FIFO can also be configured to store the data associated with read/write access to a memory location (i.e., the recent read/write record of a variable). More trigger options are available in addition to those mentioned above. Refer to the HCS08 manual for more details.

CodeWarrior User Interface

The CodeWarrior debugger provides a user-friendly environment for controlling the debugger hardware and setting different trigger options. For example, in the "Source" window, if users right-click the mouse button at an instruction, a pop-up menu will appear. It allows users to set either comparator A or B to trigger based on that instruction location. If users move the cursor to the "Data" window and right-click the mouse button instead,

FIGURE 2. AN EASY WAY OF SETTING TRIGGER AND CAPTURING BY THE POP-UP MENU



another pop-up menu for memory trigger will appear. Figure 2 shows a pop-up menu based on the instruction's address trigger.

If users want to more directly control the on-chip debugger registers, they can choose the "Expert" mode from the "Trigger Module Usage" of the pop-up menu. In this mode, users can manually adjust the registers and its control bits. It requires the users to have more knowledge about the DBG hardware.

Users can refer to the CodeWarrior Manual for more details on how to use the debugger environment.

References

Eduardo Montañez, "Using the HCS08 Family On-Chip In-Circuit Emulator (ICE)," *Freescale Application Note*, AN2596/D.

Kazue Kikuchi and John Suchyta, "HCS08 Background Debug Mode versus HC08 Monitor Mode," *Freescale Application Note*, AN2497/D.

Freescale HSC08 Microcontroller Family Reference Manual Volume 1, HCS08RMv1/D.

Kenny Lam has extensive experience with a wide range of MCU and DSC applications in industrial and automotive markets.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Upward Compatibility= Design Mobility

Upward Migration Path Helps Designers Add Features and More

By Jintao Zeng

Freescale offers a full range of 8-bit microcontrollers that satisfy customers' various needs. Customers can find the products in Freescale 8-bit MCU families that fit their applications. Among these MCU families, Freescale provides a feature and package upward migration path, which makes it easy for customers to add features or pins to their designs.

Feature Compatibility

Feature compatibility allows users to migrate into a higher-end product without losing any existing functionality in their current product. For example, a MC68HC908QY4 user who wants to add an I²C function into a design can choose MC908QB4, which includes all features of QY4 with an additional I²C. Freescale's rich product portfolio gives customers multiple choices for their feature-upward migration. Figure 1 shows the feature compatibility among our HC08* Q-Family products.

Package Compatibility

Package compatibility allows users to migrate to higher-end products in the same package without changing their existing board design. Freescale maintains package compatibility within every 8-bit microcontroller family product. For example, a MC9S08GT16 user can move into MC9S08GT32 in the same package without changing the board. Function on every pin of MC9S08GT16 will stay the same in MC9S08GT32. In case a customer wants additional pins for his application, the high pin-count product in the same family will have similar pin count allocation. Package compatibility reduces the customers' cost of improving designs and helps them quickly and accurately define their product roadmaps.

At Freescale, we never stop improving our product offering, such as optimizing pin out to bring more functionality in a smaller package and improving EMC performance. With these improvements, package

FIGURE 1. FEATURE AND PACKAGE COMPATIBILITY IN HC908Q FAMILY

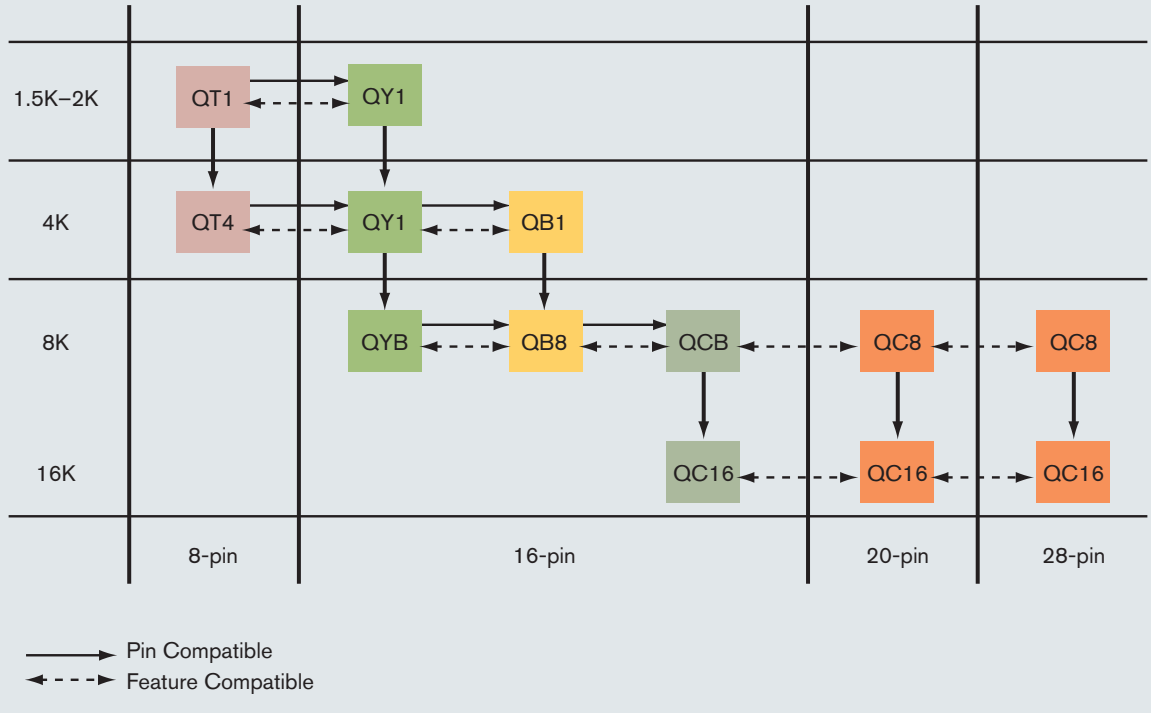
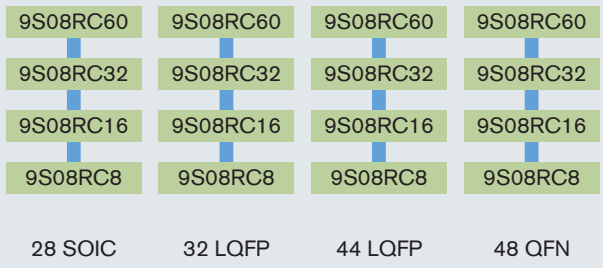


FIGURE 2. PACKAGE COMPATIBILITY IN MC9S08RC FAMILY



compatibility among different families may be disrupted; however, a similarity among these families is still maintained.

Figure 2 shows an example of package compatibility within the MC9S08RC family product.

Summary

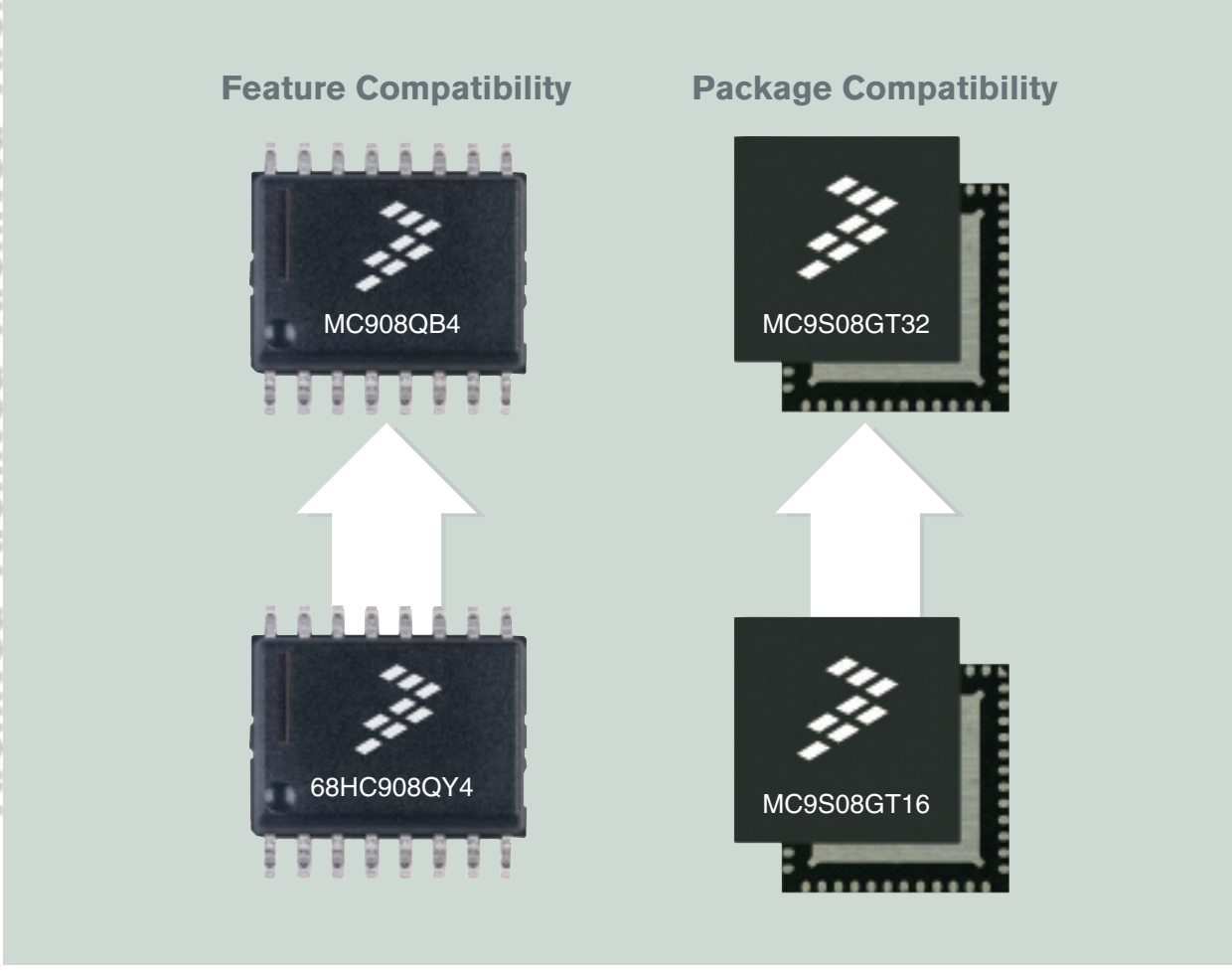
Compatibility is highly valued at Freescale. Our goal is to provide a complete portfolio of market-leading 8-bit MCU products with an easy migration path within or among the families. Feature and package compatibility can help reduce our customers' system designs' complexity and help shorten their new product design cycle.

For more information on product compatibility of Freescale 8-bit microcontrollers, please visit **www.freescale.com/8bit.**

*The HC08 products incorporate SuperFlash® technology licensed from SST.

Jintao Zeng is a product development engineer in 8-bit microcontroller division of Freescale.

Comments about this article?
e-Mail **MCUideas@freescale.com**.



Peripheral Mix and Match

By D. Scott Brown

As a leader in 8-bit microcontrollers, Freescale has a very large portfolio of MCUs with a broad mix of peripheral offerings. Finding the right peripheral match for a particular application can be a daunting task. The following table can help you choose the right MCU for your application based on the system needs and how it must interface to external components. Use this as a quick reference to find the most common uses and features for each peripheral.

For additional functional and feature information, please see the Addendum starting on page 73.

Serial Communications

Acronym	Peripheral Name	Common Uses/Applications	Highlights	MCUs	
SCI	Serial Communications Interface	High-speed, asynchronous communications with peripheral devices and other MCUs Standard universal asynchronous receiver transmitter (UART) communications Conversion to RS-232 connection for serial communication with PC	Two-wire full-duplex, asynchronous communication	**HC08s** **MC68HC908Axxx** **MC68HC908GR8/4** **MC68HC908GP32** **MC68HC908JK/JL8** **MC68HC908JL16** **MC68HC908KXx** **MC68HC908Lxxx** **MC68HC908MRxx** **MC68HC908SR12**	**S08s** **MC9S08Gxxx** **MC9S08RDxx** **MC9S08RExx** **MC9S08RGxx** **MC9S08QGx** **MC9S08AWxx**
ESCI	Enhanced Serial Communications Interface	High-speed, asynchronous communications with peripheral devices and other MCUs Standard universal asynchronous receiver transmitter (UART) communications Conversion to RS-232 connection for serial communication with PC Allows for software emulation of LIN protocol	Two-wire full-duplex, asynchronous communication Adds 13-bit break detect for software LIN emulation	**HC08s** **MC68HC908EYxx** **MC68HC908GR60/32/16** **MC68HC908GT60/32/16** **MC68HC908GZxx** **MC68HC908QBx**	
SPI	Serial Peripheral Interface	Full-duplex, synchronous serial communication with external peripherals Ex: high-resolution analog-to-digital converters (ADCs), serial EEPROM, chip-on-glass LCD screens and other MCUs Communication within PCB board at relative high speeds	Master and slave mode operation Four-wire, full-duplex or single-wire bidirectional operation Maximum master mode frequency = bus frequency/2 Maximum slave mode frequency = bus frequency/4	**HC08s** **MC68HC908Axxx** **MC68HC908EYxx** **MC68HC908GRxx** **MC68HC908GTxx** **MC68HC08GZxx** **MC68HC908GP32** **MC68HC908JK/JL8** **MC68HC908JW32** **MC68HC908LJ/LKxx** **MC68HC908MRxx** **MM908E625/624**	**S08s** **MC9S08Gxxx** **MC9S08RGxx** **MC9S08QGx** **MC9S08AWxx**
I^2C	Inter-Integrated Circuit	Synchronous serial communication with external peripherals Ex: high-resolution analog-to-digital converters (ADCs), serial EEPROM, chip-on-glass LCD screens and other MCUs Communication within PCB board at relative high speeds (slightly slower than SPI) Can be used in multinodal, serial bus communications	Master and slave mode operation Designed to operate at up to 100 kbps with maximum bus loading and timing Capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading Two-wire (pin) interface, half-duplex operation Individual I^2C modules can only transmit or receive at any given time (i.e., single-wire, bidirectional data transfer) Multiple I^2C modules can simultaneously receive a message that is transmitted on a bus		**S08s** **MC9S08Gxxx** **MC9S08QGx** **MC9S08AWxx**
MMI^{2}C	Multi-Master Inter-Integrated Circuit	Full-duplex, synchronous serial communication with external peripherals Ex: high-resolution analog-to-digital converters (ADCs), serial EEPROM, chip-on-glass LCD screens and other MCUs Communication within PCB board at relative high speeds (slightly slower than SPI) Can be used in I^2C system with multiple master devices	Same features as I^2C, but allows system to have multiple Master I^2C nodes	**HC08s** **MC68HC908APxx** **MC68HC908BDxx** **MC68HC908LDxx** **MC68HC908LJ/LKxx** **MC68HC908SR12**	
SLIC	Slave LIN Interface Controller	Designed to provide slave node connectivity on a local interconnect network (LIN) sub-bus LIN is an open-standard serial protocol developed for the automotive industry to connect sensors, motors and actuators	Automates many LIN functions for lower CPU overhead when communicating on LIN bus Full LIN error checking and reporting High-speed LIN capability up to 120 kbps at 8 MHz	**HC08s** **MC908QLx**	
USB 1.1	Universal Serial Bus 1.0/1.1	Serial communication with common PC serial interface Low-speed communication (1.5 Mbps) Targeted to various PC peripheral applications (i.e., wireless mice dongles)	Full universal serial bus specification 1.1 low-speed functions 1.5 Mbps data rate	**HC08s** **MC68HC908BD24** **MC68HC908BD48** **MC68HC908JB8** **MC68HC908JB12** **MC68HC908JB16** **MC68HC908LD60** **MC68HC908LD64**	

USB 2.0	Universal Serial Bus 2.0	Serial communication with common PC serial interface Full-speed communication (12 Mbps) Targeted to various PC peripheral applications (i.e., wireless mice dongles)	Full universal serial bus specification 2.0 full-speed functions 12 Mbps data rate	**HC08s** **MC68HC908JW32**	
DDC12AB	Display Data Channel 1 and Display Data Channel 2AB	Digital monitor applications requiring VESA-certified DDC1 and DDC2AB communication standards	DDC1 hardware; also compliant with DDC2AB protocol Compatibility with multimaster I^2C bus standard	**HC08s** **MC68HC908BDxx** **MC68HC908LDxx**	

Analog Interfaces

Acronym	Peripheral Name	Common Uses/Applications	Functional Description/Features	MCUs	
ACMP	Analog Comparator	Emulate single-slope analog-to-digital conversion Combined with TPM and external RC circuit for temperature measurement Comparing external analog signal with internal reference When output is brought out to pin (not available on all parts), it may be possible to create an op-amp circuit	Full rail-to-rail supply operation Selectable interrupt on rising edge, falling edge and either rising or falling edges of comparator output Option to compare with fixed internal bandgap reference voltage Option to allow comparator output to be visible on a pin, ACMPO (S08QGx only)	**HC08s** **MC68HC908LB8**	**S08s** **MC9S08RGxx** **MC9S08RCxx** **MC9S08RExx** **MC9S08QGx**
ATD	Analog-to-Digital Converter	Conversion of analog signal to digital Systems that need to measure temperature, voltage, pressure and motor speed Interface to sensor inputs	8- or 10-bit resolution Single or continuous conversion		**S08s** **MC9S08Gxxx**
ADC (8-bit)	Analog-to-Digital Converter (8-bit)	Conversion of analog signal to digital Systems that need to measure temperature, voltage, pressure and motor speed Interface to sensor inputs	Linear successive approximation with monotonicity 8-bit resolution Single or continuous conversion	**HC08s** **MC68HC908ABxx** **MC68HC908ASxx** **MC68HC908AZxx** **MC68HC908BDxx** **MC68HC908GP/GTxx** **MC68HC908GR8/4** **MC68HC908KXx** **MC68HC908JK/JLxx** **MC68HC908LB8** **MC68HC908LDxx** **MC908Qxx**	
ADC (10-bit)	Analog-to-Digital Converter (10-bit)	Conversion of analog signal to digital Systems that need to measure temperature, voltage, pressure and motor speed Interface to sensor inputs	Linear successive approximation with monotonicity 10-bit resolution Single or continuous conversion	**HC08s** **MC68HC908APxx** **MC68HC908EYxx** **MC68HC908GR60/48/32/16** **MC68HC908GZxx** **MC68HC908LK/LJxx** **MC68HC908MRxx**	
ADC10	Analog-to-Digital Converter (10-bit)	Conversion of analog signal to digital Systems that need to measure temperature, voltage, pressure and motor speed Interface to sensor inputs	Linear successive approximation algorithm with 10-bit resolution Output formatted in 10- or 8-bit right-justified format Single or continuous conversion (automatic power-down in single conversion mode) Operation in WAIT and STOP modes for lower noise operation Selectable asynchronous hardware conversion trigger Ability to run in STOP3 and wake up MCU from STOP3 mode (S08s only)	**HC08s** **MC908QBx** **MC908QY8** **MC908QxxA**	**S08s** **MC9S08QGx** **MC9S08AWxx**

Other On-chip Peripherals

Acronym	Peripheral Name	Common Uses/Applications	Functional Description/Features	MCUs	
MTIM	Modulo Timer	Separate timebase to use for software interrupts	Simple 8-bit timer with several software-selectable clock sources and a programmable interrupt		**S08s** **MCS08QGx**
TIM	Timer Interface Module	Input capture, output compare and PWM functionality Motor/motion control Combine with RC filter for cheap digital-to-analog converter Driving piezos or LEDs Light dimming Battery charging	Two input-capture/output-compare channels Buffered and unbuffered output compare pulse-width modulation (PWM) signal generation Free-running or modulo up-count operation Toggle any channel pin on overflow	**All HC08s**	
TPM	Timer/Pulse-Width Modulator Module	Input capture, output compare and PWM functionality Motor/motion control Combine with RC filter for cheap digital-to-analog converter Driving piezos or LEDs Light dimming Battery charging	Each channel may be input capture, output compare or buffered edge-aligned PWM 16-bit free-running or up/down (CPWM) count operation		**All S08s**

PWMMC	Pulse-Width Modulator—Motor Control	12-bit PWM for motor control applications (AC motor control, in particular)	Three complementary PWM pairs or six independent PWM signals 20 mA current sink capability on PWM pins Programmable fault protection Complementary mode featuring: – Dead-time insertion – Separate top/bottom pulse-width correction via current sensing or programmable software bits	**HC08s** **MC68HC908MRxx**	
RTC	Real-Time Clock	Used for applications that require keeping track of time/date Calendar, alarm and chronograph functions	Real-time clock (RTC) with clock, calendar, alarm and chronograph functions Selectable periodic interrupt requests for seconds, minutes, hours, days, 2 Hz, 4 Hz, 8 Hz, 16 Hz, 128 Hz	**HC08s** **MC68HC908LK/LJxx**	
XIRC	Crystal or Internal Resonator Circuit	Allows use of external clock source or trimmable internal clock source	The bus clock frequency is one fourth of any of these clock source options: 1. Internal oscillator (trimmable) 2. External oscillator 3. External RC 4. External crystal	**HC08s** **MC68HC908Qxx** **MC68HC908LB8**	
ICG	Internal Clock Generator	Internal clock source that allow pins to be used as I/O Eliminates need and cost of external clock components	Provides multiple options for clock sources; consists of four blocks: - Oscillator block - Internal reference generator - Frequency-locked loop - Clock select block	**HC08s** **MC68HC908GTxx** **MC68HC908KXx** **MC68HC908EYxx**	**S08s** **MC9S08Gxxx** **MC9S08AWxx**
ICS	Internal Clock Source	Low-power internal clock source that allows pins to be used as I/O Eliminates need and cost of external clock components	Provides clock source choices for the MCU Contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock There are also signals provided to control a low-power oscillator (XOSC) module to allow the use of an external crystal/resonator as the external reference clock		**S08s** **MC9S08QGx**
LCD	Liquid Crystal Display Driver (segment-based)	Segment-based display applications	Software-programmable driver segment configurations LCD bias voltages generated by internal resistor ladder Software-programmable contrast control	**HC08s** **MC68HC908LJ/LKxx**	
CMT	Carrier Modulator Timer	Remote-control applications			**S08s** **MC9S08Rxxx**
CAN	Controller Area Network Interface Module	Automotive/Industrial applications that uses the CAN network protocol	Modular architecture Implementation of the CAN Protocol—version 2.0A/B	**HC08s** **MC68HC908AZxx** **MC68HC908GZxx**	
HR-PWM	High-Resolution Pulse-Width Modulator	Targeted for applications that require power factor correction Motor control	One complementary output pair for driving a half bridge Dithering between two frequencies or duty cycles for increased output resolution Programmable deadtime insertion Shutdown input for fast disabling of outputs	**HC08s** **MC68HC908LB8**	
LVI	Low-Voltage Inhibit	Overall system safety; prevents micro from operating when power supply is too low (allowing corruption of code execution)	Programmable LVI reset	**All HC08s**	
LVD	Low-Voltage Detect	Overall system safety; prevents micro from operating when power supply is too low (allowing corruption of code execution) Generating an interrupt instead of resetting the part Allows battery-operated systems to alert user of decaying batteries	Selectable RESET or interrupt off LVD trip point		**All S08s**

D. Scott Brown has held multiple technical marketing positions within Freescale that span from 8- and 16-bit microcontrollers to digital signal controllers. Currently, he is helping to define the next generation 8-bit products and their feature sets.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Power Your Projects with the Right Tools

Interactive Development Ecosystem Guides You Through the Process

By Nathan Lee

Let's say that you are a home builder and have been asked to build a dog house with the following items: list of the customer's required features, wood, hammer, nails and a hand saw.

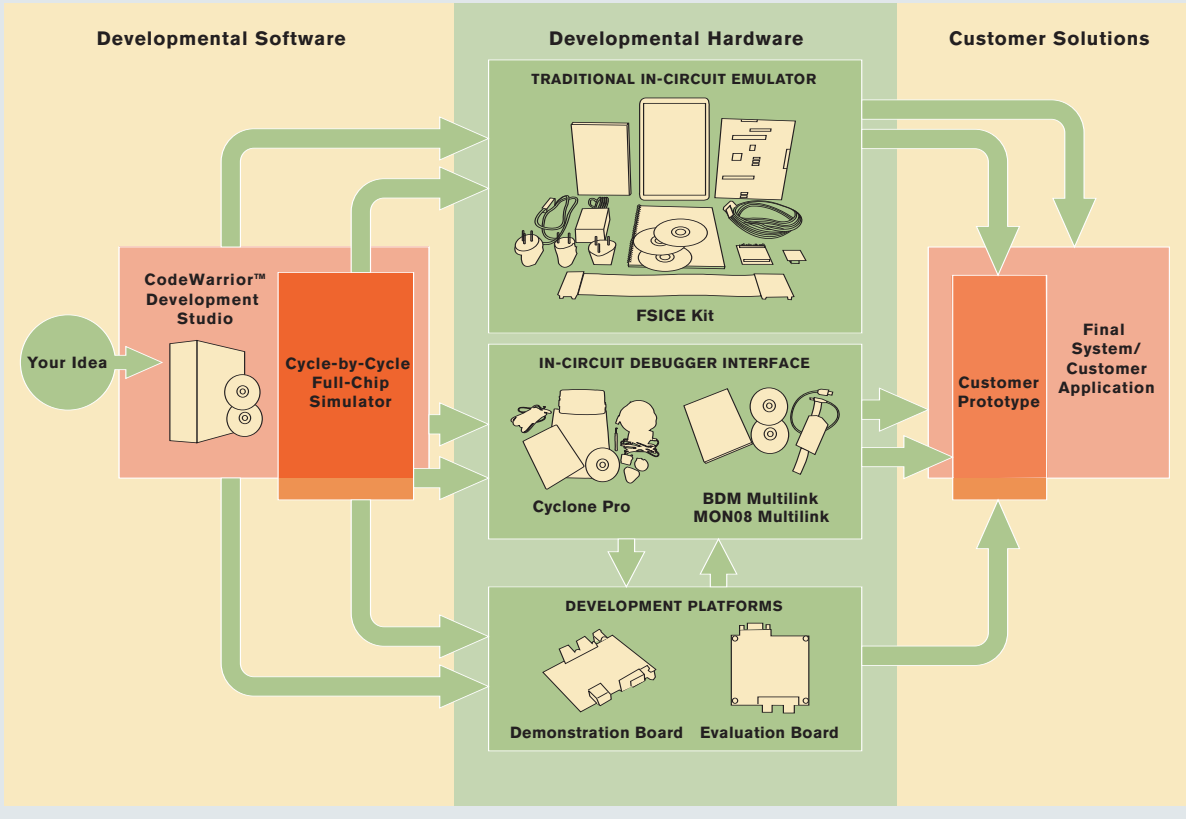
Now imagine that another home builder was given the following materials to build a dog house: list of the customer's required features, wood, hammer, nails, power saw, air tools, nail gun and other building tools.

Obviously, the second building project would be more efficient because of the tools.

Freescale understands that its customers need different tools depending on the design, but you may be asking, "How do I know which development tool is the best one for my project?" Freescale has created an Interactive Development Tools Ecosystem diagram to help you through the selection process.

As you know, all great products start with an idea—the first step in the Interactive Development Tools Ecosystem diagram. These ideas are then translated into software, or firmware, which is the next stage of the process. In order to assist in software creation, Freescale offers a comprehensive development studio that has the flexibility and the ability to grow with your needs. For example, Freescale offers three editions of CodeWarrior™ integrated development environment for the HC(S)08 products: Special Edition, Standard Edition and Professional Edition. The Special Edition of the CodeWarrior tool is available directly from Freescale.com at no cost* and includes the

FIGURE 1. INTERACTIVE DEVELOPMENT TOOLS ECOSYSTEM DIAGRAM



essential tools you'll need to begin your design. Its features include an editor, integrated debugger, unlimited assembler, 16 KB C compiler, project wizard, more than 100 sample projects, code generation tools from Unis (including Processor Expert™ and Device Initialization), a cycle-by-cycle full chip simulator and many more features. Most people are able to begin and finish their designs in the Special Edition of CodeWarrior Development Studio, but some users may need to upgrade to the Standard or Professional Editions. For more information about the different editions of CodeWarrior integrated development environment, visit Freescale.com or CodeWarrior.com.

But what good is the software without the hardware?

Freescale offers three classes of developmental hardware to meet your needs, including a traditional in-circuit emulator—Freescale In-Circuit Emulator (FSICE). The FSICE is a high-performance emulator system with unlimited break points that allow product developers to test a program by freezing it at every central processing unit (CPU) cycle and logging the results. This emulator is most often used by industries that need a significant amount of test documentation, such as automotive and health care-related instrumentation and monitoring. FSICE is a stand-alone platform for HC08 that only interfaces directly with the customer prototype or final system or customer application. The FSICE system supports only the HC08 product line, but this does not mean that Freescale has forgotten about the HCS08 product line. In fact, Freescale has decided to integrate an in-circuit emulator (On-Chip ICE) onto the HCS08 silicon. The On-Chip ICE provides developers with FSICE-like capabilities at any time in the development process, and it can be accessed with the standard BDM in-circuit debugger interface. For more information on the On-Chip ICE, please refer to Application Note AN2596: Using the HCS08 Family On-Chip In-Circuit Emulator (ICE) at **www.freescale.com**.

The next class of developmental hardware, the in-circuit debugger interface cables, includes HCS08 BDM Multilink (USBMULTILINKBDM), HC08 MON08 Multilink (USBMULTILINK08), and CyclonePro (M68CYCLONEPRO). The BDM and MON08 Multilink cables are cost-effective debugging and Flash programming interfaces for the target MCU. The Cyclone Pro provides all the capabilities of the BDM and MON08 Multilink plus it adds the ability to function as a stand-alone programmer. To use these cables, the target board must have the target MCU and the proper interface header. (A 6-pin BDM header is used for HCS08 MCUs or a 16-pin MON08 header is used for HC08 MCUs.)

The last class of developmental hardware includes the development platforms: evaluation boards (EVB) and demonstration boards (DEMO). DEMOs provide a cost-effective platform to program and debug project application code, with basic I/O functions and I/O headers for easy expandability. The latest DEMOs have integrated the USBMULTILINK cables to provide the best development system available on the market for the price. DEMOs may be all you need for complete product development, but they can also be used as a first evaluation step to be followed by more sophisticated Freescale tools. EVBs provide more functionality than DEMOs with expanded capabilities, including user I/O functions, prototyping area for custom interfacing and selectable microcontroller modes of operation. You have the option to either step up to an EVB after initial evaluation on a DEMO, or you can start with an EVB and move directly to the customer prototype or add other tools to the process.

The next time you begin a design, remember that Freescale has the proper development tools for your design. For more information about Freescale development tools including part numbers, please visit **www.freescale.com**.

*Subject to license agreement and registration.

Nathan Lee is one of the leading development tool experts at Freescale.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Bright Approaches to Microcontroller-Based LED Drivers

Advantages and Limitations to Different Topologies

By Pedro Pachuca and Rod Borras

In this article, we address microcontroller-based LED drivers. We explore the different topologies achievable with a microcontroller at the heart of the system. We discuss the topology tradeoffs in detail, with an emphasis on the major features and limitations: communication, voltage and current capabilities, dimming techniques and switching speed.

High-Intensity LEDs and their Drivers

High Intensity Light Emitting Diodes (HI-LEDs) are semiconductor devices that allow current flow in only one direction. They are formed by the union of two semiconductor materials creating a PN junction. High intensity LEDs are different from standard LEDs based on their power output; traditional LEDs are generally limited to less than 50 mW, while HI-LEDs can provide 1 to 5W.

Figure 1 represents a typical voltage-current relationship in a HI-LED. Almost no forward current (I_F) will flow through the HI-LED until the forward voltage (V_F) exceeds the internal barrier voltage. If the V_F is increased further, the curve follows the shape of a knee and suddenly rises at a rapid linear rate.

The light output of an LED is proportional to the forward current, so if the I_F is not controlled properly, it can result in an unacceptable variation in light output. Also, exceeding the manufacturer's maximum I_F specification can seriously reduce the LED's useful life.

High-intensity LEDs should be controlled by electronic drivers—their primary function is to generate a source of constant current. These circuits can provide luminosity control using the techniques described in this article, and in some cases, can compensate for temperature changes as well.

HI-LED manufacturers suggest dimming the LED by pulsing it at its constant nominal current in order to make sure that the system will provide color uniformity.

Simple Topologies and Their Tradeoffs

The challenge in designing a high-intensity LED driver is to create a well-controlled, programmable, constant current source with high efficiency.

Using a Series Resistor (Linear Approach)

The simplest way of setting a current is by adding a series resistor, as shown in Figure 2A. The advantages are a cost-effective, simple implementation and no generated noise due to switching. Unfortunately, this topology has two major drawbacks: reduced system efficiency due to significant losses in the resistor and the inability to change the luminosity. This solution requires a constant voltage source to achieve a constant current. For instance, if we assume that V_{DD} is 5V, and that the LED's V_F is 3.0V, then to produce a constant current of 350 mA, you will need: R=V/I, or in this case, R = (5V-3.0V)/350 mA = 5.7Ω.

We can see that with these values, R will dissipate $R*I^2$ or 0.7W (almost as much as the LED), so the overall efficiency will inevitably fall below 50 percent.

This approach assumes a constant V_{DD} and a constant V_F. In fact, V_F varies with temperature and so will the current. Using a higher V_{DD} will minimize the overall current variation due to the V_F; it will also create significant losses in the resistor, further reducing the efficiency.

FIGURE 1. LED VOLTAGE-CURRENT RELATIONSHIP

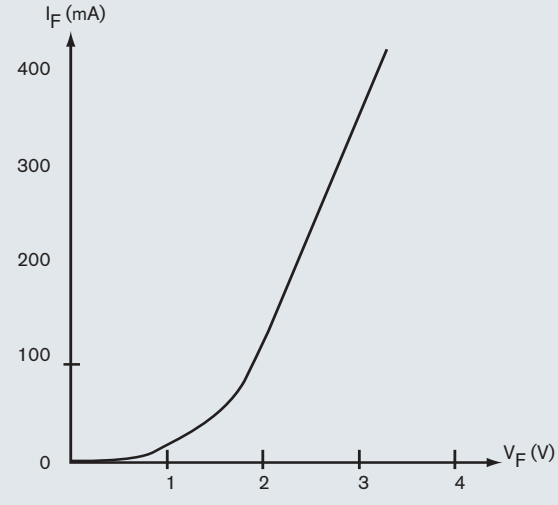
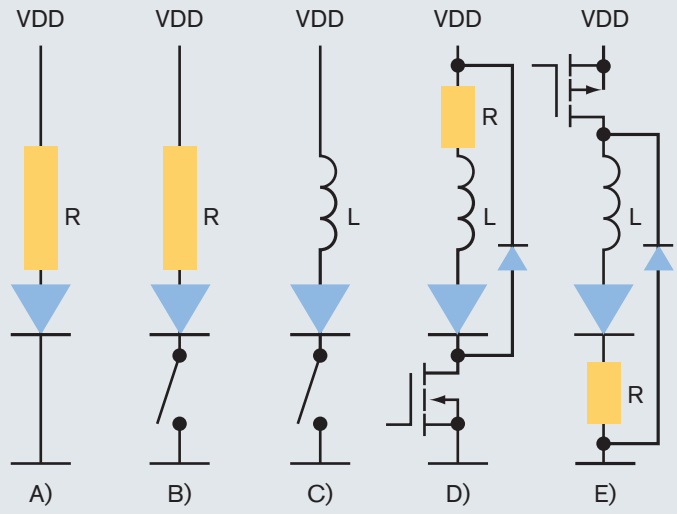


FIGURE 2. LED DRIVER TOPOLOGIES



Once we have created a constant current through the LED, we need to find a way to set different luminosities. Given that these LEDs always need to be driven at their nominal current, we can achieve luminosity control by turning that current source on and off with a programmable duty cycle. This will require a switch as represented in Figure 2B.

Using a Linear Current Source

By adding a transistor and/or an op-amp, the current can be very accurately programmed to 350 mA. Unfortunately, the overall efficiency and R power losses remain the same.

Using a Low-Side Switch (Switched-Mode Approach)

Figure 2C shows the low-side switch concept. We can program the current flowing through the LED by allowing current to build-up in inductor L when the switch is on, and ramp down when the switch is off, as seen in Figure 3. As with any inductive load, we need to provide a path for the current when the switch is open. This is achieved via the freewheeling diode represented in Figure 2D. We have replaced the switch with an N-channel MOSFET and added resistor R to measure the current through the LED.

The switch will turn on when the current decays to the lower current threshold (i.e., 300 mA) and will turn off when the current builds up to the upper threshold (i.e., 400 mA).

This example has the switch on the "low-side" (hence the name), and is very easy to implement. To turn on the field effect transistor (FET), you need only a 5V voltage on its gate that could be provided directly from one of the microcontroller outputs. Furthermore, this topology no longer needs a constant V_{DD} voltage and will regulate the current, even for fluctuating input voltages.

Current sensing resistor R has to be in the "high-side" portion of the circuit. If it were connected to the source of the MOSFET, it would only see the LED current while the switch was on, and could not be used to set a secondary threshold. (See Figure 3.)

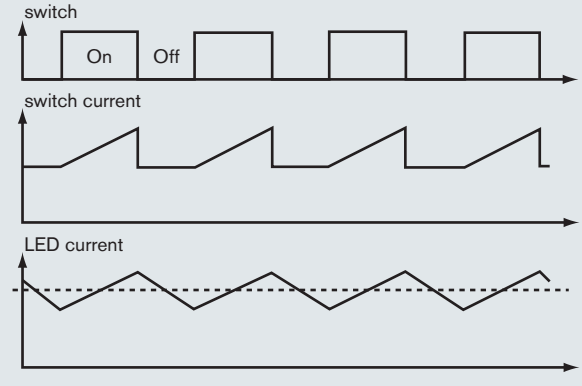
This topology, which looks like the front-end of a boost converter, has the advantage of using an N-channel, low-cost FET, but requires a differential voltage measurement across R to extract the current through the LED.

The switch actually provides two functions: first, it allows for a programmable current to be set in the inductor; and second, it allows for luminosity dimming.

Using a High-Side Switch

The high-side switch is the exact same circuit as the low-side switch, except the load and the transistor have traded places. Figure 2E shows the switch is now in the "high-side." We have also changed the FET from N-channel to P-channel. An N-channel FET would require a V_{GS}>5V to fully turn on: in this topology, the N-channel's source voltage will vary, and will often be above 3V so we would

FIGURE 3. LED AND SWITCH CURRENTS



need at least 8V on its gate. This requires a gate-drive circuit, like a charge-pump, and makes the overall circuit slightly more complex. It is simpler to have a P-channel FET, and provide a V_{GS} of -5V to it, once again directly from a microcontroller output. This topology is similar to the front end of a buck converter.

The main advantage of the high-side switch is that the current measurement is done directly across R, and therefore does not have to be a differential measurement.

Dimming Techniques

There are many techniques to dim LEDs; some are patented. Here is a brief description of some of them. In all cases, the average luminosity is achieved by turning the LED fully on (at its nominal current) and off at very high speed (to avoid flickering), and is proportional to the percentage of time the LED is on.

Pulse Width Modulation

This technique uses a fixed frequency of period T, as shown in Figure 4. The dimming is achieved by varying the pulse width. Figure 4 shows three different luminosity levels with duty cycles of 6 percent, 50 percent and 94 percent.

Frequency Modulation

This technique, published by Artistic Licence, uses the concept of a fixed-width control pulse as described in

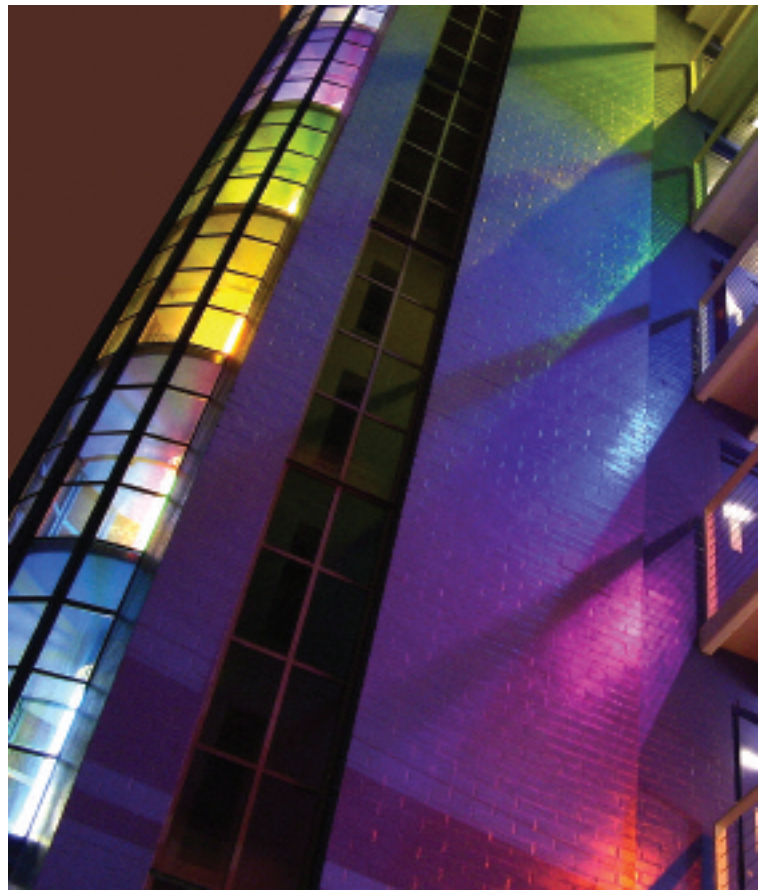


Figure 5. Pulse A is always of the same duration. The luminosity is controlled by how often Pulse A repeats itself.

Bit Angle Modulation

This new technique, also published by Artistic Licence, is based on a binary pulse train that contains the intensity value. Every bit in the pulse train is stretched proportionally

FIGURE 4. PULSE-WIDTH MODULATION

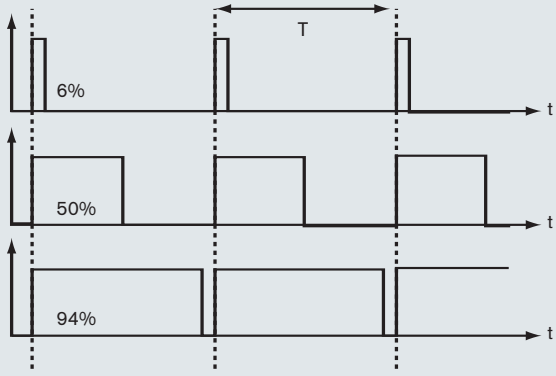
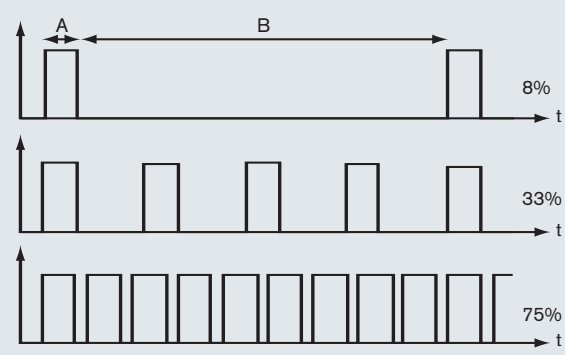


FIGURE 5. FREQUENCY MODULATION



to its significance. If the least significant bit b0 has a duration of 1, then bit b1 has a duration of 2, bits b2 through b7 have durations of 4, 8, 16, 32, 64 and 128 respectively. This is shown in Figure 6.

Communication Protocols

DMX512

DMX512 is a standard published by U.S. Institute of Theatre Technology (U.S.I.T.T.). The protocol, initially used to control lighting dimmers, has been extended to control lamp movement, slider projectors and many other lighting accessories. DMX512 runs over an EIA-485 standard. Data transmission is based on 8-bit asynchronous serial communication, one start bit, two stop bits and no parity; it allows 256 dimming levels.

Digital Addressable Lighting Interface (DALI)

DALI is a standard developed for communication with electronic ballasts. It is included as an appendix to ECG standard IEC 929. DALI is designed for the use of standard components and for simple wiring, which means low costs.

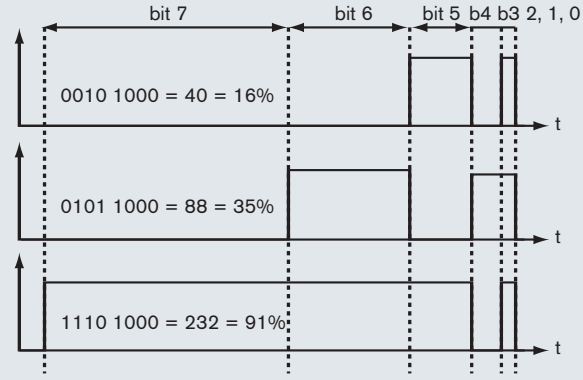
Fields of application may be dimming lights and pre-setting values for different lighting environments, and properly adjusting light settings depending on the direction of daylight, energy savings and more.

DALI is based on the master-slave principle: the user operates the system through the controller (master); the controller sends messages to all the ballasts (slaves) containing an address and a command. The address determines whether the ballast should listen. Each ballast is digitally addressed and therefore is insensitive to electromagnetic noise (improvement over the analog 1-10V dimmer switch system).

ZigBee™

ZigBee is a communication protocol resulting from the combination of Home RF Lite and the IEEE® 802.15.4 specification. ZigBee operates in the 2.4 GHz and 868/915 MHz ISM bands, and lighting applications are one of its primary markets due to its capability to offer low power consumption at low cost. ZigBee offers network capabilities useful in lighting systems, plus the advantages of wireless control.

FIGURE 6. BIT ANGLE MODULATION



Limitations Using a Microcontroller

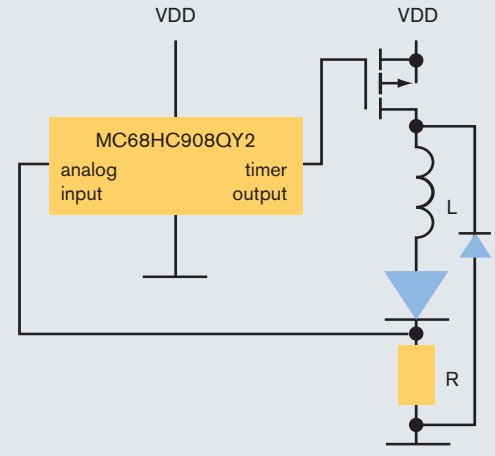
Voltages and Currents

If V_{DD} is the supply to both the LED and the microcontroller, then there is only enough voltage to drive one LED. The simple topologies we have discussed do not allow for the LED voltage to be higher than V_{DD} (see Figures 2 and 7). For more LEDs in series, with the benefit of all being at the same current, V_{DD} must be higher and now requires a power supply for the microcontroller.

Physical Interfaces to Support Communication

The microcontroller only provides simple synchronous (SPI) or asynchronous (SCI) communication. Additional hardware and software is needed to implement DALI, DMX, LIN and more.

FIGURE 7. MICROCONTROLLER-BASED LED DRIVER



Constant Current Regulation and Switching Speed

The key parameter in this application is switching speed. Larger inductors, which are more costly, are required for slower switching speeds. Most microcontrollers can accomplish an A/D conversion in about 15µs. Add a few instructions to compare the read value to internal thresholds, and the conversion is up to 30 to 40µs for the full analysis per ON or OFF cycle with an uncertainty of about 15µs. This error dictates the minimum inductor value as shown in Figure 8. Another approach is to set arbitrary ON and OFF durations, and then readjust these to try and accommodate the two current thresholds. This indirect method allows for a smaller, lower cost inductor, but it's less accurate.

FIGURE 8. BASIC DESIGN CONSIDERATIONS

Assumptions
1. VDD=5V, VF=3V
2. IF=350mA, with lower/upper thresholds at 300mA/400mA

Resistor R
Max power losses: 1/4W, so R $Imax^2$ <0.25W
Imax=400mA, so R <1.56 Ohms
R set at 1 Ohm, for a max voltage drop of 400mV

Inductor L
Max 10% error (40ma)
with 15us uncertainty, and V=L di/dt
L> (5V-3V) * 15us/40mA, so L>750uH

Dimming and Modulation Speed

At 100 percent luminosity, there is no need to modulate the transistor. At the other extreme—for the lowest luminosity level, i.e., one percent—it will be necessary to have the transistor on for one percent of the time. Given the fact that dimming must be done at 100 Hz or higher to avoid flickering, the PWM frequency must be 10 kHz or more. The eye can detect minute changes in the low luminosity range, and therefore 100 steps are not enough. If 4,000 steps were required (12-bit resolution), the PWM frequency would have to be around 400 kHz, which is almost impossible for a simple microcontroller.

The Future

It is simple to design a microcontroller-based, high-intensity LED driver. The three main limitations encountered are:

> Processing speed and the impact on inductor size and dimming resolution
> Communication capability with industry standards
> Drive capability for multiple outputs and/or LED strings

We will address solutions for all these limitations in a future article.

Rod Borras is a systems and applications engineer with Freescale's Analog Products Division. He has an MSEE from INSA Toulouse, France. He joined the company in 1989 when it was still part of Motorola, and has held a variety of positions. His expertise has been in power products, microcontrollers, sensors and analog.

Pedro Pachuca is a technical marketer in the MCU division of Freescale Semiconductor. He earned his MS degree in electronic engineering from Instituto Politectnico Nacional in Mexico D.F. In 1996, he joined Freescale when it was a part of Motorola.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Win the Race by Simplifying Development

Here's How with Fast Track™ Services

By Christine Deliance

Companies that win the race to market with new product designs consistently become market leaders in their industries. Freescale Fast Track™ services can help you win that race as you accelerate the development cycle—its convenient services are designed to help you every step of the process. Visit www.freescale.com/fasttrack and open the door to support services that not only help you be the first to market but also help you be the best in market.

Explore

Find the online educational resources that you need to take full advantage of Freescale technologies.

You have free access to a variety of online technical training resources, from general product overviews to in-depth training materials, in the Freescale Technical Learning Center. Make a point to visit the 68HC08MC68HC08 8-bit Flash microcontroller training course and any of 11 training modules that describe the CPU, reset and interrupt processing, Flash memory and several MCU peripherals. Ten additional online product courses in Chinese to support the Asia design engineering community are planned to launch soon.

The Technical Learning Center also features online development tools, known as virtual labs. These full-featured evaluation boards and development software are accessible via a standard Web browser at no cost for 24 hours a day, seven days a week. The latest is our demo board for Freescale MC9S08QG8, which provides access to a fully licensed CodeWarrior™ integrated development environment with Processor Expert™ configuration tools. With these tools, you will be able to familiarize yourself with programming and debugging the QG8 demo board. Visit **www.freescale.com/training** to learn more.

Our worldwide Freescale University Program provides first-class services and architecture solutions to educational institutions, enabling computer science and engineering students to gain experience with widely used Freescale products and architectures.

Select

Whatever you want to know about Freescale and its products and services is just a few clicks away. Access our extensive documentation library using a part number or a parametric search. The first and best destination to find in-depth information about our industry standards activities, alliance partnerships, third-party relationships and our Environmentally Preferred Products initiatives is at **www.freescale.com**.

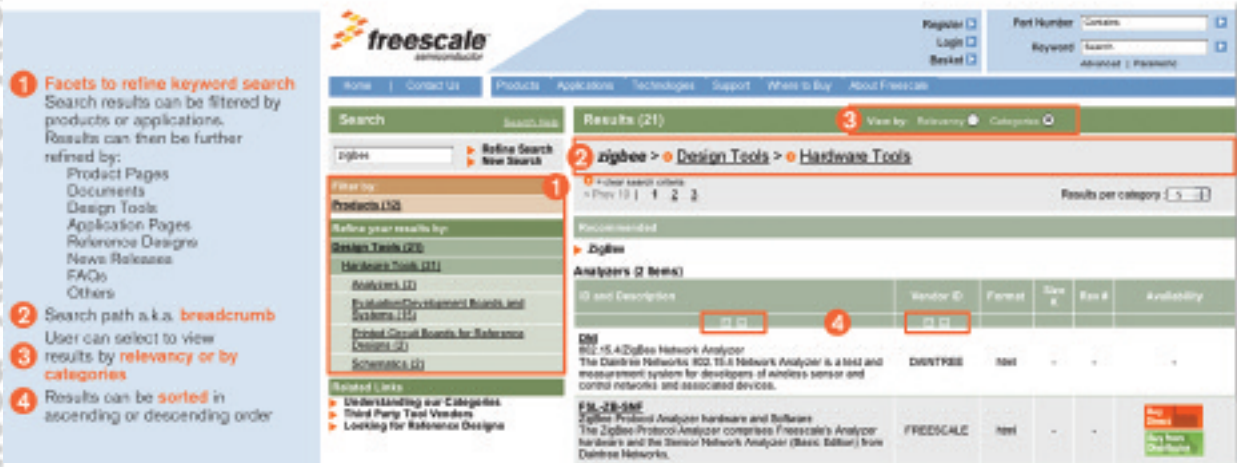
As part of our ongoing commitment to you, we redesigned the search interface on **www.freescale.com** to make it more intuitive and more flexible—with relevant results. Now, instead of getting a long list, users can narrow their results without encountering dead ends. From numerous interviews with our customers, we learned that design engineers want the flexibility to view search results organized by categories and by relevancy. For example, you can not only search by keyword, but also narrow your results to see only certain types of documents, such as data sheets, application notes and reference designs. You can also perform a part number search and filter to see only the RoHS-compliant active parts.

The same simple interface has been implemented for documentation, tool and part number searches to provide a consistent user experience throughout the site.

Develop

Fast Track services provide you with quick access to products and programs that help your development processes work for you—not against you.

ANATOMY OF A KEYWORD SEARCH RESULT PAGE. IN THIS EXAMPLE, THE USER SEARCHED ON "ZIGBEE™" AND REFINED HIS RESULTS BY PRODUCTS AND DESIGN TOOLS.



Sample Program is a fast and easy way to order product samples online. Samples are free and if they are in stock, they will be shipped to you the next day.

DevToolDirect allows developers to quickly find and order online Freescale hardware and software development tools and third-party design tools.

> Orders can be securely placed online by credit card
> Shipping is available via FedEx® Standard or Priority anywhere in the world
> 90-day, "no-hassle" return policy

Technical Information Center gives you the option to seek online technical support and help through worldwide phone assistance from people speaking key languages. Online support includes an innovative system you can use to submit and track your technical questions directly on www.freescale.com.

Design Alliance Program (DAP) is a dynamic worldwide community of design engineers and consultants who provide our customers with resources to accelerate time to market. DAP provides you with access to our global directory of skilled technical alliance associates who are experts in hardware and software development with Freescale products.

We are also enhancing the Freescale online communities and forums so that you can get connected with other developers and discuss your passion for technology and share tips and insights on Freescale products.

Stay up-to-date with Freescale's new developments and events through our first set of RSS (Real Simple Syndication) feeds on **www.freescale.com**. Subscribe to RSS updates and get the stories delivered to you when news that concerns you makes headlines.

Buy/Go to Market

Freescale offers one of the broadest and most comprehensive services-distribution networks in the semiconductor industry. Our Co-Branding Program gives you the opportunity to build on the strength of thousands of patent families and our industry-wide reputation for innovation and leadership.

Working with Us

Freescale's Fast Track services can make a big difference in how quickly and efficiently you get your products to market. Explore, select, develop, buy/go to market and additional innovative services are at your disposal to help make our products work for your designs.

Fast Track services are constantly evolving to meet customer needs. For more details, see **www.freescale.com/fasttrack**.

Christine Deliance leads a team of Web and CRM professionals at Freescale. She has over 10 years of experience in global customer-facing program implementations, including the management of freescale.com.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Friendly Flash and the End of the ICE Age—Flash and Debug Support Continue to Evolve

By Jim Sibigtroth

Embedded microcontrollers continuously evolve to meet the ever-growing expectations of users. This article will explore two areas where recent changes have responded to these pressures. In the area of Flash memory, the Freescale HCS08 and HCS12 devices have added on-chip state machines to manage the programming and erase processes to greatly simplify application software. In the area of in-circuit emulation (ICE) support, these MCUs have introduced on-chip ICE systems that mark the beginning of the end for traditional external ICE systems. We will use this on-chip debug system to examine a novel software routine that enables Flash modification from within the user's application program.

In the ongoing search for inexpensive debugging techniques, HC12, HCS12 and now HCS08 MCUs have integrated a single-wire background debug interface. These systems address the problem of accessing internal information from the MCU, but traditional in-circuit emulators remain the best tool for real-time debugging because they allow bus information to be captured without disturbing the flow of the application program. The newest HCS12 and HCS08 MCUs provide these bus capture and intelligent triggering capabilities in a new on-chip module.

First, we will look at some recent Flash memory improvements and then we will use the on-chip ICE system to examine a Flash programming utility routine in more detail. In the process of debugging this routine, we will more clearly see how the routine works. We will also see how the on-chip ICE system works. It would have been difficult to debug this Flash routine without the on-chip ICE system because earlier debug techniques such as single instruction trace or using a ROM monitor program would have run into problems during the Flash programming operation when the Flash memory is effectively removed from the memory map.

Friendly Flash

The latest improvement to Flash memory is the addition of an on-chip state machine that automates most of the steps needed to erase pages of Flash or program Flash locations. This state machine helps in two main ways. First, it drastically reduces the number of instructions needed to perform Flash operations. Second, it provides better control of the programming sequence and timing. Making the programming time consistent improves the reliability of Flash data and prolongs the write-erase lifetime.

The timing is controlled by an internal Flash clock derived from the bus clock. You set the FCDIV register once during initialization so the Flash clock is between 150 KHz and 200 KHz. The FCDIV register is write-once so you can't accidentally change it after it is initialized.

Flash operations consist of five simple steps.

1. Clear any old error flags by writing a value to the FSTAT register.
2. Write data to an address in Flash to latch address and data.
3. Write one of five commands to FCMD. The allowed commands are Blank Check, Byte Program, Burst Program, Page Erase or Mass Erase.
4. Write a 1 to the FCBEF bit to confirm/start the command.
5. Wait for the command complete flag.

In the HCS08, the last four steps need to be done by instructions outside the Flash because the Flash is removed from the memory map during that time. In the HCS12, the Flash control logic includes extra buffers so only the last two steps need to be located outside Flash. This paper will examine very small routines you can include in your application code in Flash to perform these Flash operations. Once these routines are included in your application program, programming a Flash location is as simple as loading the address and data into registers and calling a subroutine.

The ICE Age

Designers have had a long love-hate relationship with the traditional in-circuit emulator (ICE). On one hand, they provide valuable insight into the internal operations of a microcontroller during the development and debugging process. Unfortunately they also have a few less desirable traits. They are expensive, the interconnection systems tend to be complex and difficult to work with, and as hard as the designers try, they never quite do a perfect job of emulating the real microcontroller. Many application systems are so small that it is impossible to connect an emulator in the real application system. Other applications are embedded into machinery where an emulator connection would interfere with normal operation of the machine.

Traditional in-circuit emulators also imply that the address and data information from the microcontroller is visible on the microcontroller's pins. Typically address, data and a few control signals are provided as alternate functions on several MCU pins. When the device is used in an emulator, the normal I/O functions of these pins are rebuilt with custom hardware in the emulator. Although these rebuilt I/O functions can be made to emulate the function of the real microcontroller pins, they usually cannot exactly duplicate subtle timing and drive characteristics of the original MCU pins. In cases such as the smallest 8-pin microcontrollers, where the microcontroller doesn't have enough pins for address and data, the emulator must use some sort of special bond-out device or completely emulate the function of the small microcontroller.

Cooler Than ICE

The HCS08 family of microcontrollers and the newest HCS12s, greatly reduce or eliminate the need for a traditional in-circuit emulator by building the functionality of an emulator with a bus state analyzer inside the microcontroller. This approach completely eliminates the need for an expensive external emulator box and the nasty interconnect problems. Since none of the pins on the microcontroller are used by this on-chip debug system, there is no need to rebuild anything in external hardware and the emulation is exact, including subtle timing and drive characteristics.

This on-chip system includes address, data and R/W comparators as well as trigger control logic similar to that on a traditional emulator. (See Table 1 for a brief summary of trigger modes supported by the on-chip debug system.) Like almost any ICE, you can capture bus information before or after the trigger event. You can choose to stop execution of the application program at the end of a trace run, or you can keep running after the trace information has been captured. You can even set an independent hardware breakpoint to occur at some point after the trace information is captured. Potential circuit damage can be caused by stopping the application at certain times, such as while the controls to a motor H-bridge driver are being changed.

TABLE 1. THE ON-CHIP DEBUG SYSTEM SUPPORTS NINE TRIGGER MODES. AN ADDITIONAL ADDRESS COMPARATOR C IS AVAILABLE TO SET AN INDEPENDENT BREAKPOINT. IN THE HCS08, BREAKPOINT C IS IN THE BACKGROUND DEBUG MODULE SO IT IS ONLY ACCESSIBLE THROUGH A BDM POD. IN THE HCS12, BREAKPOINT C IS ACCESSIBLE THROUGH BDM OR THE SERIAL MONITOR.

Mode	Comparator A Matches…	Comparator B Matches…	FIFO Captures…
A-only	address	—	COF
A OR B	address	address	COF
A Then B	address	address	COF
A AND B Data	address	data	COF
A AND NOT B Data	address	data	COF
Inside Range A to B	address	address	COF
Outside Range A to B	address	address	COF
Event Only B	—	address	data
A Then Event Only B	address	address	data

The bus capture buffer is a little different than that on a traditional emulator because buffer memory is still relatively expensive in terms of silicon area. Because of this, the on-chip system is more selective about what gets stored into this memory. Traditional emulators take the blunt approach to capture the address and data for every bus cycle and then worry about sorting out what is useful after the trace run is finished. In these new on-chip systems, most trace runs capture only the address information from change-of-flow (COF) events. The host debug software can easily fill in the gaps between these COF entries.

One benefit of having the trigger logic on chip is that it has access to internal CPU control signals. This allows a user to configure a trigger to occur when an instruction has propagated through the instruction queue and is about to be executed. This is called a tag-type trigger as opposed to a force-type trigger that fires as soon as a comparator match occurs. In another case, the newest HCS12s use CPU signals to allow the capture mechanism to ignore "free" cycles in order to avoid filling up the capture buffer with unimportant information.

The best thing about this on-chip debug system is that the hardware is built into the MCU and you can download a special edition of CodeWarrior™ v5.0 to get full access to the applicable debug features. If you use an available serial monitor program in the target MCU, you can connect the target system to your PC with a simple serial cable. Or you can buy an inexpensive BDM pod for the connection from the host PC to the target MCU. Both connection methods provide the same debugging capabilities but the BDM pod is completely non-intrusive.

You use the same techniques to set up triggers as you would use to set simple breakpoints. The user interface is smart enough to automatically select some trigger settings based on context. For example, if you right-click on an instruction line in a source code window, the debugger knows that you want to set a tag-type trigger on an instruction. If you right-click on a data value in a memory display window or a variable label in a software variables' window, the debugger knows you want to set a force-type trigger at the selected memory address. After setting the trigger address, additional pull-down menus allow you to specify the conditions for R/W and whether you want to trigger on a specific data value at the selected address. Some trigger modes require two trigger addresses and the debugger recognizes that you are trying to set up one of these triggers when you right-click on another instruction or another memory location.

A Practical Example

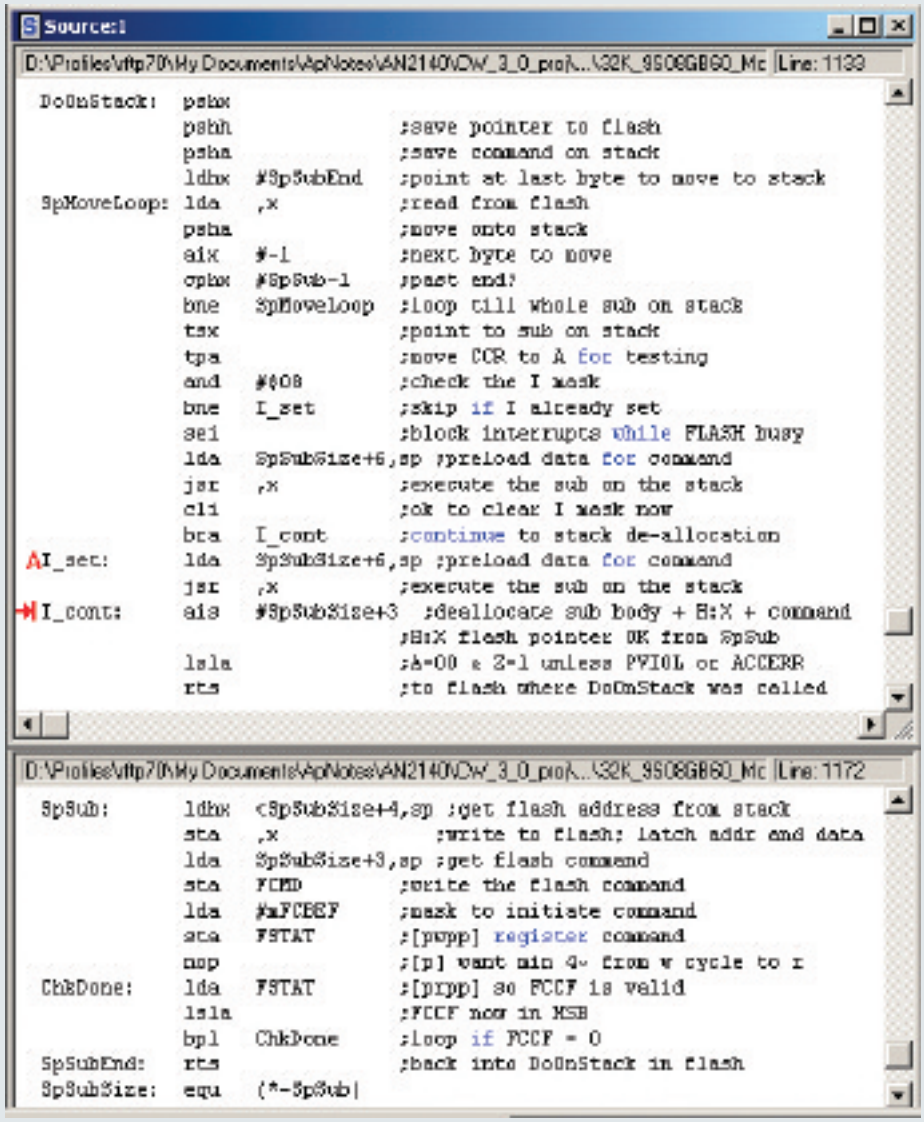
Now let's use the advanced debug functions to examine a novel software routine that programs or erases Flash memory. Typical trace techniques would not work well for this routine because the Flash memory is disabled during the actual program or erase operation and this would interfere with debugging. When the debugger tries to read Flash during the operation, invalid data is returned. Also, any attempt to read from Flash in the middle of a programming or erase operation would cause an error and abort the operation.

The on-chip debug system will allow us to trace this routine because it non-intrusively captures bus information during the trace run. We will set a trigger just before the start of the programming operation. It will be configured to continue running after the trace buffer is full. An independent breakpoint will be set just after the operation is complete. This allows us to see what happened without disturbing the operation.

We will examine a routine called DoOnStack that is shown in Figure 1. This routine resides in the Flash and allows you to program or erase another area of the same Flash even though the Flash memory is removed from the memory map during the Flash operation. It does this by copying a small subroutine called SpSub onto the stack, and calling that subroutine to perform the critical steps of the Flash operation. When the Flash operation is finished, the program returns to DoOnStack in Flash and de-allocates the stack space used by SpSub. The subroutine that actually controls the operation is executing from stack RAM during the Flash programming operation.

With older MCU Flash memories, the code to perform the Flash operation would use so much stack space that this approach probably wouldn't be desirable. In the HCS08, the stack subroutine uses 24 bytes and in the HCS12 it only takes about 12 bytes, which is about the same as a normal interrupt stack frame. Interrupts are blocked just before calling the small SpSub routine on the stack and are unblocked as soon as it is finished. The execution time to copy this small routine onto the stack is small enough that it easily fits in line with normal application code. Programming a single byte of Flash only requires about 45 microseconds so there are no problems with extended

FIGURE 1. DoOnStack AND SpSub SOURCE CODE. THE RED A IN THE LEFT COLUMN NEAR THE BOTTOM OF THE DoOnStack ROUTINE INDICATES TRIGGER POINT A HAS BEEN SET AT THIS ADDRESS. THE RED ARROW, TWO LINES BELOW, INDICATES A BREAKPOINT IS SET AT THIS ADDRESS.



interrupt latency due to Flash programming. If you program several Flash locations in a series, interrupts are unblocked between operations so interrupts only need to wait for the current byte to finish.

Figure 1 shows two source code windows from the CodeWarrior IDE containing the source code for the DoOnStack and SpSub routines. Both of these routines are located in the application program in Flash memory. These routines are designed to allow you to erase a 512-byte page of Flash or program a byte of Flash in the same Flash memory where your application program is located. With these routines, you can easily use one or more pages of the Flash memory as if it was EEPROM.

Refer to the DoOnStack subroutine in the Source:1 window at the top of Figure 1. After copying entry values onto the stack, the SpMoveLoop copies the SpSub routine from Flash onto the stack (RAM). After this move operation, the stack pointer points at the start of the relocated SpSub routine. The tsx instruction copies the stack pointer into X so we can use a JSR ,X to call the relocated subroutine in stack RAM. We want to block interrupts during the Flash operation and restore the I mask after we return so there are two slightly different blocks of

code depending on whether the I bit was already set or not. After returning from the JSR ,X subroutine call, we de-allocate the stack space we used, adjust the position of error flags in A and return.

The JSR ,X calls SpSub to complete the last four steps of the Flash operation and return to DoOnStack. Now refer to the SpSub routine in the bottom half of Figure 1. The Flash address that will be operated on was pushed onto the stack before SpSub was copied onto the stack. When SpSub is executed, <SpSubSize+4,sp provides the appropriate offset where the Flash address can be accessed. The < before SpSubSize+4 indicates to the assembler that you want this value to be treated as an 8-bit offset even though SpSubSize is a 16-bit value in the assembler.

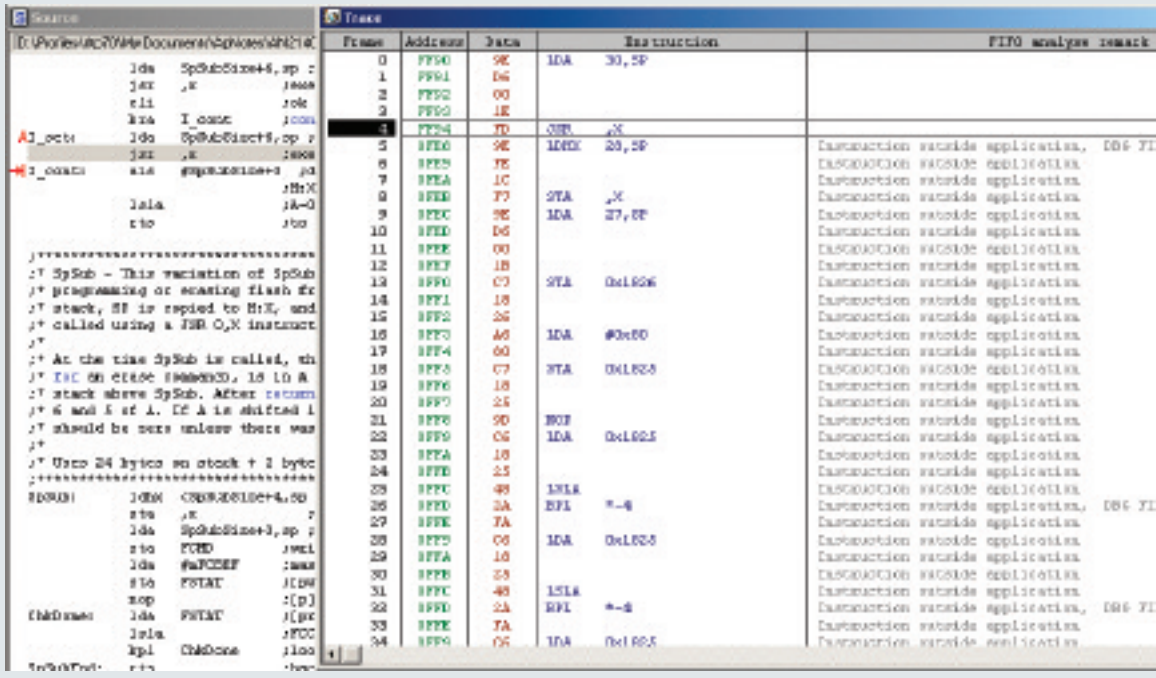
Look Ma, No Emulator

A typical in-circuit emulator has an external box connected to the target system with some sort of wiring harness, including dozens of wires. With this on-chip debug system, the capture memory is inside the target MCU; you just need a way to set controls and read results. There are at least two ways to do this. The preferred way to connect the target system to the host PC is through a background debug pod because it allows completely non-intrusive access to the target system through a single dedicated pin on the target MCU. USB versions of BDM pods are available.

The second way is through a simple RS232 serial cable from the PC to the target system. To use this option, you need a small 1 KB to 2 KB serial monitor program in the target MCU. Evaluation boards for the MC9S08GB60, MC9S12E128 and MC9S12C32 all support the serial monitor option. The CodeWarrior debugger looks and works the same regardless of which connection option is used.

Figure 2 shows the results of a trace run using the on-chip bus state analyzer. Before running the program you want to debug, you open a trace window in the CodeWarrior tool. You can see the source code partly hidden by the trace window toward the left side of Figure 2. The red A in the left column just above the JSR ,X line is the trigger point for starting the bus capture. We set this by right-clicking on that source line. In the two lines below, there is a red arrow in the left column indicating that we have set a breakpoint that will fire after we finish the JSR to SpSub that has been relocated to the stack. When we run this program, the bus capture starts when we reach trigger

FIGURE 2. SOURCE AND TRACE WINDOWS SHOWING THE RESULTS OF A TRACE RUN. AS THE CURSOR IS MOVED OVER LINES IN THE TRACE WINDOW THE LINE, 4 IN THIS FIGURE, IS HIGHLIGHTED AND THE CORRESPONDING LINE IN THE SOURCE CODE IS ALSO HIGHLIGHTED GRAY.



point A and stops when the buffer gets full. A little later, the program will stop when it reaches the breakpoint after the JSR.

After the program stops, the CodeWarrior tool reads the contents of the trace buffer and fills in the trace window. This window includes frame (line) numbers for reference, program addresses, program data, re-constructed instructions and comments. Lines that include the comment DBG FIFO are the only bus cycles that were actually captured into the on-chip bus capture FIFO buffer. All other lines correspond to cycles that the host debugger has reconstructed based on knowledge of your source code and the change of flow events that were in the FIFO.

In the portion of the trace that we can see in Figure 2, there are only three change-of-flow events. Frame 4 is an indexed JSR so the FIFO captured the destination address that corresponds to frame 5. Frames 26 and 32 are both branches that were captured as change-of-flow events because the branch was taken. Frame 0 is a special case. The CodeWarrior tool knows this address was executed because the trigger was set at the instruction at this address.

The comment “Instruction outside application” indicates that the addresses in frames 5 through 34 do not correspond to addresses in the source program. Usually this would indicate an error, but in this case it corresponds to when the CPU is executing the SpSub routine that we copied onto the stack before calling it. Note the addresses are in the area around 0x0FE8 through 0x0FFE. This area is near the top of the stack RAM.

As you move the PC mouse over lines in the trace window, the corresponding line in the source code is highlighted gray so you can associate events in the trace with the source code. In Figure 2, Frame 4 is highlighted in the trace window and the JSR ,X line is highlighted in the source window.

The reconstructed instruction sequence in the trace window provides much useful information to developers as they debug their program. We can see that the LDA at the trigger address and the highlighted JSR ,X were executed as expected in trace frames 0 through 4. In frame 5, we see that the indexed JSR took us to address 0x0FE8 in stack RAM and executed a LDHX 28,SP instruction that corresponds to the ldhx <SpSubSize+4,sp instruction at SpSub: in our source code. The instructions executed in

frames 5 through 34 are in the relocated copy of the SpSub routine, not the original routine from the source code that is located at a different address in the Flash memory of the MCU. When the program reaches the branch (BPL) instruction in frame 26 and 27, we see from the comment “DBG FIFO” and the address 0x0FF9 in frame 28 that the branch was taken. This loop near the end of the SpSub routine is waiting for the Flash programming operation, which is being automatically controlled by an on-chip state machine, to finish.

Close and Postscript

While these improvements to Flash memory and debugging support are a significant advance from earlier MCUs, they are really just another step along a continuing evolution. Other non-volatile memory technologies such as MRAM are bound to eventually replace even this Flash memory technology. At some point, writing or re-writing non-volatile memory may be as easy as writing to a static RAM. For now, automating the program and erase operations make it possible to develop very efficient routines to simplify the use of Flash memory.

The on-chip bus analysis capability is a more significant evolutionary jump. The problems of cabling and emulation speed are becoming downright unmanageable using the traditional ICE approach. On-chip debug offers a practical solution to the impending technical barriers of speed, pin density, and more. The HCS08 demonstrates that this on-chip approach works. Future generations of MCUs could increase the size of the capture buffer, add trigger comparators and add the ability to trigger on CPU register contents.

The MC9S12E128, MC9S12C32, MC9S12NE64 and MC9S12UF32 have already gone a little further than the HCS08. They increased the depth of the capture FIFO from eight words to 64 words and added two new capture modes called “Loop1” and “Detail.” Loop1 inhibits multiple captures of the most recent change of flow event as in delay loops and loops where you are waiting for a flag to set. This increases the effective depth of the capture buffer. The detail mode uses pairs of FIFO words to capture the address and data for every significant bus cycle around the trigger rather than just change of flow addresses. This allows you to see values that are loaded into registers, and data values read from or written to memory. In this detail mode, selective capture is used to ignore “free” cycles where internal CPU operations are being performed and the address and data bus are not used for any meaningful data transfer.

References

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

Eduardo Montañez, “Using the HCS08 On-Chip Debug System,” *Freescale Application Note*, AN2596/D.

Motorola, *HCS08 Family Reference Manual Volume 1*, 2003.

J. Sibigtroth, “Serial Monitor for MC9S08GB60,” *Freescale Application Note*, AN2140/D.

J. Williams, “Serial Monitor Program for HCS12 MCUs,” *Freescale Application Note*, AN2548/D.

Jim Sibigtroth has worked for Freescale Semiconductor (formerly the Semiconductor Products Sector of Motorola, Inc.) for more than 27 years and is currently a senior systems engineer in the 8/16-Bit MCU Division of the Transportation and Standard Products Group. Jim defined the original MC68HC11 and wrote the M68HC11 Reference Manual, commonly known as the “Pink Book.” More recently he defined the CPU12 instruction set and the single-wire background debug interface that is on all HCS08 and HCS12 MCUs.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Design Alliance Program

Steps You Through Concept to Production

By Gary Streber

Freescale can help bring your 8-bit product concept to market quicker and more efficiently than ever before. For more than a decade, Freescale has been improving its processes and adding value to better support the always-changing customer model.

In years past, customers came to Freescale strictly for silicon. Over time, we have seen hardware and software technology, integration and the supplier base grow at phenomenal rates, offering a plethora of potential silicon enablers. At the same time, the end product complexity, user demands, system requirements, software and logistics go beyond what a single company can hope to master and manage internally.

Freescale is a leader in a number of key markets. We serve the automotive, industrial, medical, networking, wireless and merchant markets. It is because of our product breadth and broad customer base that we look for additional support methods to meet the needs of our customers.

That's why we offer access to support resources through Fast Track™ services. Customers can pick and choose internal and external resources to help make a product concept a reality. The Fast Track services include many resources such as data sheets, samples and a directory of external independent third-party design houses that have a variety of expertise to help customers with the product development cycle. Learn more about Fast Track services and resources at **www.freescale.com/fasttrack**.

The Freescale third-party independent design house membership, the Design Alliance Program (DAP), is made up of about 400 hardware and software companies globally. Although the DAP members vary in size, location, applications and product expertise, they all offer some level of custom design work as a part of their core business, utilizing Freescale's 8-bit through 32-bit technologies. In addition, to broaden their offerings and time-to-market enhancements, DAPs have developed reference designs, operating systems, application firmware, software stacks and even manufacturing capabilities to support varying customer needs.

All DAP members have technical experience with various Freescale products and technologies, and their experiences are validated by a Freescale representative. Through a special collaborative relationship, we keep the members up to speed to better enable them to support customers. The members are provided with early access to product information and given technical training to help them prepare in advance of customers' needs. Many Freescale new product introductions and collateral are co-developed with DAP members. This effort helps bring

members up to speed, reduces the learning curve and helps reduce risk and improves time to market.

Many DAP members showcase our relationship and their utilization and support of Freescale technology by proudly displaying the Freescale Alliance logo.

freescale™
Alliance Member

Most DAP members support customers, from concept, product release and through any number of steps in the process. Depending on the customer need, the engagement may be initiated at any point during the process.

DAP support is designed to be very complementary to a customer's internal organization. Many times, DAPs work closely with a customer's internal resources, and each focuses on their core competencies to help get products to market faster. This is true for both hardware and software development. A particular portion of the product, such as wireless communications, may be critical for the application but is not a long-term key requirement. You may call upon the DAP member to provide a wireless module, subsystem, or other solutions to meet your needs.

About 30 percent of our current DAP membership has experience with products utilizing each of our featured technologies. More than 230 members have experience with our microcontroller families, and 182 of the members have Freescale 8-bit product experience.

The member directory and additional information about Freescale's DAP program, including our certification criteria, can be found at: **www.freescale.com/webapp/dap.search.framework?NEXT_SCREEN=SEARCH**.

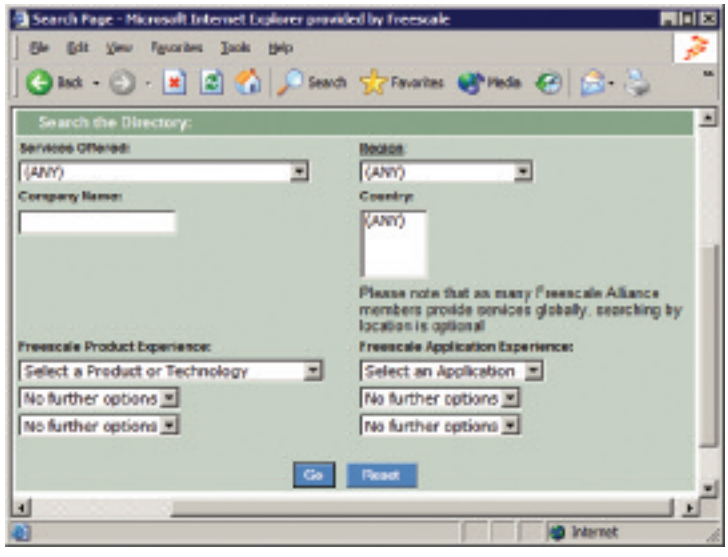
As shown in the figure below, the directory allows you to search for the right member through device or product experience, application experience, location as well as some other criteria.

Once your requested list is displayed, you can get more details on the members, a direct link to the member's Web site and contact information.

In addition to their vast product experience, Freescale Alliance members deal with a large number of customers, applications and multiple markets. The customer base ranges from Fortune 500 companies to small start-ups. This variety allows the DAPs to draw from a broad knowledge base.

With the broad customer reach, up-to-date training, early product access, an open door to Freescale experts and a wide variety of application experience, Freescale and our Design Alliance Program members can supplement customer design resources from concept to production.

FIGURE 1. SEARCH THE DAP MEMBER DIRECTORY



For more than 25 years, Gary Streber has worked with customers and their needs in the semiconductor industry. He manages the Design Alliance Program at Freescale.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Add It Up

How Value Is Added Throughout the Product Development Lifecycle

By Javier Gutiérrez

Introduction

Service cannot be seen or touched, but you immediately notice when it is not there. Freescale makes an effort to provide great service to help you stay ahead and at ease. Great support is not only helping when you call and ask for help—it is being there during the whole development process. Freescale is accessible, useful, resourceful and committed to help you win.

In this article, we will show you how Freescale can provide services that add value to your process and help you get your product to market fast.

New Product Development

Freescale helps accelerate the development cycle and provides convenience at every step of the new product development process. Freescale offers support services designed to help you stay a step ahead of the rest.

Companies handle new product development in different ways. However, most of them coincide with eight different phases in the process. Freescale can work with you to add significant value during the first four stages and beyond. We are committed to help you with:

1. Idea generation—this is when the new product development process begins. It's searching and generating novel ideas that satisfy customer needs. We are committed to communicating closely with you, understanding your needs and reflecting them in devices to help ease your design process. Speak up, we listen!
2. Idea screening—this is the effort many companies undertake to encourage new ideas from their employees. We strongly believe this is a great path to innovation. Freescale assists your talented people with new device and technology training. We offer demonstrations and reference designs that are focused on helping your talent stay at the vanguard of the technology and help motivate them toward new ideas.
3. Concept development and testing—this is a very important stage of the whole process. Consumers do not buy product ideas. Instead, they buy product concepts. We can help you create the best concept.
   - > Take advantage of virtual tools where you can develop and test your concepts without the expense of hardware tools.
   - > Use software tools, such as Processor Expert™, to help you make quick prototypes. You can use all available resources to help prove your idea is attainable.
   - > Use Freescale's highly qualified technical support to help you get your product to market fast.
   - > Hire our engineering services to help you develop your "dream" so you can transform it from idea into reality.

At this stage, efficient and accurate support is the advantage designers need to position their products ahead of the competition.

4. Marketing strategy development—this is the key for the success of your product. We understand it—that is one of the reasons we make the effort to understand your market and what you need to position your product to achieve consumer preference. We provide you with compatibility across technologies, different choices within a technology and scalability throughout a family of products. This means that you have the flexibility to create products that better suit the consumer needs in the regions or segments in which you want your product to fit.
5. Business analysis
6. Product development—this is when your prototype is working and approved. This is the time to make feature enhancements and look for cost efficiencies. You can work with our technical marketing and sales teams to find the right approach to fit your customer needs.
7. Market testing
8. Commercialization

Support Services

Support goes throughout the whole development process. Freescale strives to provide you with the information you need. Our Web site is designed to provide detailed information on our current devices and system solutions at any stage of your design.

Visit **www.freescale.com** and conduct parametric searches for all of our architectures and access reference designs, a rich library of technical documents, online training, real boards through a virtual lab and an easy-to-use interface to order free samples and cost-effective tools directly from Freescale. All this helps you make the right product decisions, understand our products and use them properly in your design to speed up the cycle.

We want to give you the necessary information at the right time. However, we also know that there are always questions you need answered, issues you need help resolving and statements in documentation that need clarification. This is when personal support becomes indispensable.

There are several types of support which differ according to the complexity of the project, the expertise of the team, the applicable technology, the innovation of the project, the market segment and other different needs that you face each day.

Some of the support services we provide include:

Technical documentation—this is a very critical part for every application design. We provide you with useful data sheets and user manuals. We always show qualified specifications that are carefully tested.

In terms of applications, we have quick reference notes with basic examples of how to configure our modules with code written in embedded C. These quick reference notes help developers avoid from having to start their applications from scratch.

We also have application notes that work as a guideline to help designers accelerate the development process, from basic to complex application examples.

As a final aid to speed up the go-to-market process, we offer reference designs for key applications that exhibit the capabilities of our devices. Most importantly, we provide a hardware tool that fully proves a concept and, with basic adaptations, could make your application ready

for production. The reference designs come with a full document, including source code, gerber files, bill of materials and a detailed explanation of how the application works.

Virtual tools—this is a different and innovative level of support. This is a virtual capability for our customers who need to prove a concept, but are not ready to invest in tools. You can test your code virtually, via the Internet, and debug in a remote development board placed just about anywhere in the world. In some cases it is even possible to view what is happening through a webcam while running your code. If necessary, we can arrange a virtual meeting with a support engineer to assist you.

It is important to emphasize that not all technologies support this capability, but we are working to include all of them soon.

Hardware and software tools—this is fundamental for the success of product developments. The best way to prove a concept, or test a new technology or device, is to use a development board and targets supplied by the silicon manufacturer. We understand your needs, which is one of many reasons we are committed to creating the best hardware and software tools for our devices.

Among our hardware tools, we offer cost-effective kits, development boards and emulation boards to more complex systems, such as real-time emulation boards that allow you to get the best out of our devices while easing the task of debugging. We also rely on third-party partners who help us develop tools to aid you during the design process, such as serial, BDM and gang programmers, serial interfaces and more.

In the software arena, we are well-positioned for 8- and 16-bit micro and hybrid controllers. Using the CodeWarrior™ integrated development environment together with Processor Expert technology and FreeMaster, we have a winning platform with a comprehensive set of tools and features.

The CodeWarrior tool suite provides code-size-limited licenses for testing purposes with the necessary features enabled, allowing the developer to use the powerful capabilities of the system without having to spend money. This limited license is available at no cost* for all 8- and 16-bit technologies.

Processor Expert technology is an extra feature integrated into the CodeWarrior tool suite, which is also available with the license at no cost*. Processor Expert technology provides a set of easy-to-use beans that can help you develop more easily and faster than you could if you created all the drivers on your own. More complex drivers from the CodeWarrior tool suite are available with the purchase of a license. For the prototyping stage, Processor Expert technology is designed to help you quickly create a working application. So after you have proven the concept and sold the project, you will have the opportunity to do some code size enhancements.

FreeMaster is included in the CodeWarrior tool suite, which can work as an HTML user interface, allowing you to check and modify variables in real time. It also might work as a debugging tool, virtual oscilloscope, and in many other uses as well.

Global online support—this is a very important section of our technical services. The support page **(www.freescale.com/support)** offers answers to thousands of frequently asked questions and is available around the clock, every day of the week.

Also working live 24 hours, five days a week, the Technical Information Center (TIC) provides fast and professional technical support services around the world via web, mail and phone. We answer simple questions—where to find the data sheet or how to order a sample—and very technical questions that require product specialists or engineers with expertise in applications.

The process to contact us through the web is very simple. Just log on to our support page and place your question, explain your problem or share your needs. Requests generated from this Internet interface are automatically fed into a worldwide customer database as a technical service request. You immediately receive a service request reference number. This reference number lets you view the request status, deliver additional information directly to our specialists and re-open or cancel a request.

Through our worldwide network, one of our specialists can immediately start working on your service request. We are located in the Americas, Europe and Asia regions—we cover every time zone. Our goal is to answer your request within two days. Depending on the complexity of the issue, most of our customers receive an immediate or overnight

response, though others may take several days to receive a satisfactory solution to the problem.

You can always use our phone system, where a representative will be there to assist you. If your question is highly technical, we encourage you to use our Web system.

Worldwide numbers can be found on our support page **www.freescale.com/support**.

Field technical support—this is the most direct support service we offer. It can be provided by a field applications engineer (FAE) or technical sales engineer (TSE) from any of our authorized distributors, or directly through Freescale by an FAE, field technical marketer (FTM), or product specialist (PS). This kind of support is needed when the problem cannot be solved remotely and requires a visit from our specialists. You have the support on site, where you need it.

Quality support—we count on quality engineers around the world to help resolve the issues our customers report. These engineers work together with you to provide solutions to your problems in a timely manner.

Quality support is provided in different ways. One way is to report the issue through the TIC where the issue will be researched and assigned to the appropriate specialist. Another way is to directly contact your quality engineer if one has been assigned.

At times, problems may be traced to the application's design (software, hardware compatibility, PCB layout and more) or in the manufacturing process. The job of our quality group is to help you find those flaws through different testing procedures. If the issue is not traced to an outside source, we will proceed with internal research.

Engineering services—this is a support service that helps you develop a part of your entire application when you hire Freescale services. While you are the expert of those applications in your respective market, sometimes you may not have the time, the specific knowledge, or even the resources to work on a key project. Freescale offers you a cost-effective alternative. Most of our applications laboratories are strategically located in cost-effective regions.

We offer numerous kinds of services:

> Specialized training on key technologies
> VIP support
> Code migrations
> Technology migrations
> Intellectual property development
> System cost reductions
> Application design
> Outsourcing

Different teams will lead the tasks, which are based on the requirements.

The applications laboratories are in charge of the execution of code and technology migrations, IP development and system cost reductions. However, a multi-skilled team is formed when outsourcing services are provided.

Services are based on our online technical support duty—the portal to increase our technical knowledge and understand what the customer needs.

*Subject to license agreement and registration.

Javier Gutiérrez is committed to technical support and customer satisfaction as part of the worldwide support network and value-added services Freescale offers to its customers.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Partner Up for Superior Products

Why Partnering with Distributors Is Smart

By Pattye Brown

Freescale Semiconductor has established long-term relationships with several distributors who can provide you the service level and product access you require, from design conception through mature production. For further details on specific distributor contacts, visit the Freescale Web site at www.freescale.com.

Freescale partners with several types of distributors and they all effectively support their markets. The categories of distributors include catalog, online, sampling, full service distributors and global distributors. Depending on your level of business and the support you require, one distributor may satisfy all of your needs, or you may require a combination of distributors. This article provides an overview of the services (in alphabetical sequence) that these channel partners provide to help make your efforts cost-effective, fast, easy, productive and successful.

Board Level Assembly Services

Many distributors have close relationships with assembly services subcontractors or they provide these services as part of their distribution strategy. Utilizing the distributor who has worked with you to identify the components of your system-level design offers more benefits than just access to products. The distributors can work in partnership with a subcontractor by providing just in time (JIT) access to Freescale components and all the devices necessary to cost-effectively build your board and to meet your specific system production schedules. Through-hole and surface mount methodologies can be utilized by the subcontractors who make the investment in capital to reduce your manufacturing costs by spreading the costs of their equipment across many customers' production runs. This can help you move to production more quickly and provide access to the most up-to-date manufacturing equipment without the intensive investment of your own manufacturing line. As an example, transitioning board builds to RoHS-compliant materials or environmentally preferred products can be supported by these broad-based subcontractors. By utilizing board level assembly services, you can quickly move from your specifications to a fully completed board that meets your defined requirements for components, dimensioning, tolerances, thermal requirements and quality.

Many subcontractors can also provide board-level test capabilities based on your design specifications. Some even offer qualification or burn-in services for a start-to-finish assembly and test and qualification process flow. With the high cost of fuel and shipping, any reduction in the transport time and costs associated with moving the boards from one location to another can be extremely beneficial to your end product designs and production budgets.

Broad Line Card

In addition to offering a broad range of Freescale products, distributors carry a wide range of passive components, switches, connectors and more that can aid you in fulfilling your board requirements. Use your distributor sales and engineering teams to keep up-to-date on the current offering of Freescale products that meet your design project needs, and use them, as well, to keep abreast of the new, innovative product offerings that can give you the performance, pricing, packaging, tools and services to help you improve your systems solutions in the future. Distributor teams may be able to offer a system price when a customer chooses multiple Freescale semiconductors and other passive components from their line card. Consider taking advantage of bundled price savings when choosing your system solution by selecting the Freescale semiconductors, passive components, switches, connectors and more from a single distributor.

Design Services

Some distributors provide a range of system design services to support system solutions that use Freescale products. These design services may range from telephone or online technical support to full design engineering, prototyping and beyond. Distributors offer hardware and software design services and work with customers to establish specifications, statements of work, terms and conditions of board or system designs—all to allow the customer to move fast into the design phase and help reduce the engineering resources, overhead and investment typically required by the system.

During design phases and often into production, the distributor partner acts as liaison between the design

services function and the customer, ensuring a smooth effective flow of information, confirmation of critical dates, requirements and achievement of schedules. The distributor design services may also be valuable in redesign efforts driven by product obsolescence. Many customers don't have the additional engineering staff required to support current projects and backfill needs for product obsolescence redesigns. The distributor can fulfill these temporary design engineering service needs and allow the customer to keep their emphasis and efforts on revenue growth-generating designs.

Design Tools

The distributors can offer quick access and support for a broad range of Freescale hardware and software design tools. These tools include everything from background debug connectors to full-blown system emulators with prices as low as $50.* Distributor support includes engineering and technical support of hardware and software design tools from Freescale. (See Field Applications Engineering Services.) Some distributors offer overnight shipment of design tools so you can get started on your design within 24 hours of choosing your best-fit Freescale solution.

A few distributors even design and sell promotional quick-start kits for Freescale components. These kits offer easy load and start-up of software design environments and hardware. In most cases, you can be up and running with your design process within minutes. Ask your distributor which of Freescale's cost-effective, quick-start kits can help you quickly initiate your design process.

Environmental Initiatives

Freescale and our distributor partners have a serious commitment to environmentally preferred RoHS-compliant products. Distributors offer services to help you minimize your current inventory impact and build adequate inventories of RoHS compliant product to help you achieve the objectives required by June 2006. For further information see www.freescale.com/pbfree or your preferred Freescale distributor's website.

Field Applications Engineering Services

Most distributors offer some type of Freescale product or applications engineering support. Typically, a catalog or online distributor can offer telephone or online applications engineering support, while a full-service distributor can provide in-person field applications engineering services based on the volume of purchases or project sizes of an individual customer. Applications engineering services may include provision of technical information, answers to brief technical questions, design review and component recommendations, design issue resolution, complete system review, assistance in writing software, qualification recommendations and many other functions. Regardless of the service level or the experience level of the design engineer, it is likely that customers will find a need for applications engineering services at some point during their design process.

Flash Programming

With the continuing value recognition and growth of Flash-based semiconductor products, many distributors offer services to support Flash-based programming of semiconductor devices. The service saves the customer the cost of purchasing and maintaining volume Flash programming equipment and software and provides a cost-effective way for a product to be programmed to the customer's specification in the volumes required when they need the product. The service also allows stocking commonly used products (and often returnable products under specified terms and conditions) that can be programmed in various ways for different end-product requirements. This flexibility can often reduce what might be the cost of a custom part to the level of a service fee for a Flash-programmed part—typically a small per unit adder.

In-House Stores

Depending on a customer's volume of business with distribution partners, there can be an opportunity to arrange for in-house or consignment stores. In-house stores provide JIT access to product. Often, the material held secure in in-house stores remains an asset of the distributor until it is transacted out of the in-house store. The benefits for the customer are quick access to product and management of the variations of manufacturing volumes often experienced in high-volume customer facilities.

Smaller customers may achieve results similar to the in-house store by working closely with the distributor sales

person to provide periodic advance forecasts (outside of lead time). These forecasts can help the distributor pipeline adequate product for smaller customer orders, while providing a comparable level of service.

Kitting Programs

Smaller customers can achieve economies of scale by identifying the full bill of materials (BOM) required for their electronic design and offering the BOM for quoting to their distributor of choice. Distributors may offer kitting services that gather all the listed BOM items into a kit before shipping to the customer or subcontractor for board build. The kitting process provides a JIT inventory flow and relieves the customer of the cost of carrying components in their inventory until the full set of materials for the board build comes together.

On-Line Ordering

Many distributors provide online ordering capabilities, which can reduce the paperwork and cost associated with non-EDI (electronic data interchange) methods. Larger customers may manage online ordering via open purchase orders and complex EDI inventory system monitors; smaller customers may use these online ordering capabilities to purchase by credit card. Most distributor partners offer access to Freescale products through online ordering as well as traditional ordering systems.

Pricing Management

Distributors typically work closely with customers to provide the best pricing package for single or multiple Freescale components or a BOM. If the distributor has the opportunity to provide a quote on a BOM, there is often the opportunity to work with the various vendors of active and passive components to reach the distributor's BOM pricing targets.

Distributors also help customers review their board solutions for potential cost-effective alternatives. Cost reduction may relate to component quality or selection or to a product obsolescence that introduces a potential for other pricing considerations. When working with full-service distributors, consider regular reviews of your board and component selection throughout the design process to ensure a well-matched cost solution to meet your end-customer needs.

Product Access

Freescale product access through distributors can be made through many different avenues, from online to traditional ordering methods. Some distributors offer immediate shipment for 24-hour delivery of in-stock materials. Current product lead time information may be provided via telephone or during a conversation with your inside or outside distributor salesperson. The key to product access is planning, particularly where large or varying volumes are the norm. Distributors work closely with customers to determine needs, forecasted six months to a year, and they manage their assets accordingly to build inventories that minimize lead-time impacts to customers. Stocks may be held in many forms by Freescale; i.e., wafers, die and packaged units, but they are typically held only by distributors in packaged unit form. Some distributors sell smaller quantities while other distributors sell products in minimum tray or tape and reel configurations.

Product Obsolescence Management

Distributors are committed to supporting customers in minimizing the impact of product obsolescence on new and future designs. Distributors manage product change notifications and are the key communication link to their customers, and they also help manage the transition to newer products. They assist and lead the component selection for redesign, review current designs and make recommendations for an optimum component replacement solution. They can also show a cost-effective system or subsystem solution with return on investment (ROI) for further end-product savings.

In addition to the design services segment of product obsolescence management, often the remaining life cycle of the end product will not justify the cost of any redesign. Distributors can help mitigate product obsolescence by working with customers to identify product volume requirements through the discontinuance of the end product. Distributors can stock adequate products to support the remaining end product needs and any supporting replacement part requirements after the end product is no longer offered. Distributors' outside sales personnel work with customers to weigh all the key variables and help customers make optimal decisions based on their financial requirements and the life cycle of the end products.

Sales and Customer Service Support

The outside and inside sales teams are often the first contact for most customers in purchasing and engineering departments. Inside sales teams offer services such as telesales for smaller orders, price quoting for individual components or BOM, lead time quoting, and other service offering arrangements and scheduling, such as kitting, Flash programming, tape and more.

The outside sales team often is the first face-to-face contact with a customer. The outside sales team gathers key information on the customer's design and presents alternatives for solution. An outside sales team member may be the first to present a new distributor service or product to the customer. Should further technical support be required, the outside sales person is responsible for arranging the next step. An outside sales member also takes orders and provides the customer a high quality of follow up and service to ensure ongoing sales. Take advantage of your outside sales personnel by requesting regular updates on current and new Freescale products and solutions.

Samples

Distributors offer many avenues of access to samples of Freescale products. Catalog and online distributors may offer immediate shipment of samples for receipt within 24 hours. Full-service distributors may offer a similar cycle time on samples and may also schedule a visit for the sample delivery. Samples are typically ordered through an inside sales contact. Because samples are the key to moving into the prototype build, planning for these essential start-up units is important. For full-service distributors, consider letting your inside or outside sales person know the date you need samples at the first conversation. For online or catalog distributors, enter your sample order well in advance of the protobuild. The evaluation board or evaluation module is the first design requirement for embedded semiconductor component design. However, the need for samples fast approaches once the design process is complete. Planning ahead allows your distributor to have those early prototype samples to you in advance of your build schedules.

Stocking Programs

Stocking programs vary from distributor to distributor. Minimally, distributors work to identify fast-moving Freescale products, and they stock these products for quick turnover to customers. A large percentage of distributors' business falls into the quick-turn category. When lead time might impact customers, similar strategies are followed and fast moving products are stocked. Customers are also encouraged to provide forecasts to full-service distributors to allow them to pipeline product to meet the run rates of high volume or volumes that cover a wide range over time. In some instances, distributors will hold specific stock for customers. Consider providing regular forecasts outside of quoted lead times to your distributor of choice to ensure access to required product.

Tape and Reel

Some distributors offer tape and reel services. Die and finished packaged components may be suitable for tape and reel services. Distributors may require minimum quantities for tape and reel service provision. Some online and catalog distributors are willing to provide tape and reel for small quantity purchases as well.

Training

Freescale distributor partners offer several training alternatives. They may direct customers to the online training, including virtual labs with access to real, working development tools. They may offer workshops-in-a-box where a field applications engineer visits a customer for an appointment to provide a hands-on introduction to a product. Distributors may review documentation with customers to guide them in quickly finding key information essential to the design process.

Distributors can join Freescale to provide seminars at the branch location. In this situation, Freescale and the distributor partner provides a brief training session and offers instructional hands-on labs. Customers usually take home a development board for a minimal seminar fee. Seminars cover products or focus on specific applications. In either case, the goal is to provide the customer some design experience with Freescale products.

*Prices subject to change.

Pattye Brown is a channel marketing manager for Freescale's 8-bit microcontrollers. She also teaches university-level undergraduate marketing courses.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Green Is Smart Business

Freescale's Environmental Product Program Helps Ensure Your Requirements Are Met

By Pattye Brown

Freescale is committed to environmentally conscious products and works to support you with relevant information, processes and products that you require to meet Restriction on Hazardous Substances (RoHS) compliance objectives.

Research and Development from an Industry Leader

Freescale wants to support you with the information and products you need to transition as smoothly as possible into compliance with the new RoHS environmental regulations. Today, many Freescale products that are shipping in high volumes meet RoHS requirements for all of the listed substances. By the end of the first quarter of 2006, Freescale intends to have RoHS-compliant alternatives for all products in the portfolio.

A key aspect of RoHS compliance and high temperature attach-capable products is the qualification of these products using established industry standards.

> All products intended for high temperature attach meet Package Peak Temperatures (PPTs) per JEDEC standard J-STD-020C.

> Moisture Sensitivity Levels (MSLs) are fully characterized for increased PPTs.

> Freescale tin whisker stress conditions are currently based on JEDEC (JESD22-A121) and iNEMI: Freescale intends to use JESD201 upon its release.

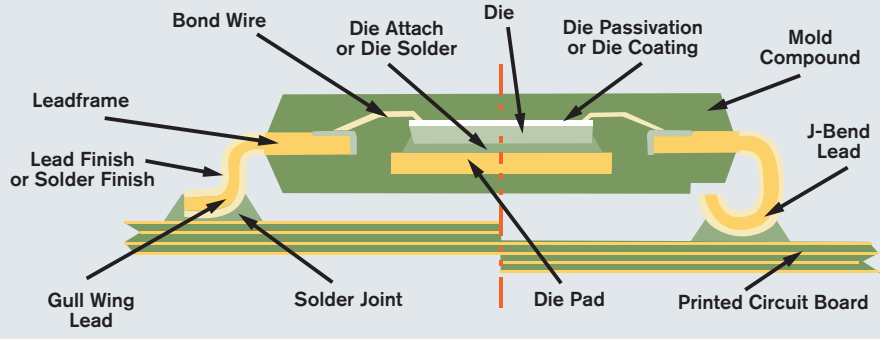
> Established industry environmental stress testing was performed on qualification vehicles per package family. These tests may include:

- Temperature Humidity Bias (THB)
- Highly Accelerated Stress Test (HAST)
- Autoclave (AC)
- Temperature Cycling (TC)
- High Temperature Operating Life (HTOL)
- High Temperature Storage (HTS)

Differentiation and Flexibility for Supply Chain Management

Product identification is a key concern of customers with regard to RoHS compliance and managing inventories. By providing unique part numbers for RoHS-compliant

FIGURE 1. LEAD FRAME COMPONENT CROSS SECTION



Find Compliance Data on Your Parts

Visit the Freescale Web site at **www.freescale.com/pbfree** to view detailed compliance information associated with the European Union's Restriction of Hazardous Substances.

Freescale delivers the information you need for RoHS compliance:

> Lead-free and RoHS data
> RoHS strategy
> RoHS implementation
> Supplier compliance
> Shipment packaging
> Technology papers
> Definitions and FAQs
> Online service requests

Freescale has defined strategies to provide solutions to meet the strategic needs of our customers. The information to transition into compliance with the European RoHS Directive, which is effective July 1, 2006, can also be accessed from Freescale's Web site.

With our strong research and development foundation and our demonstrated leadership in materials research, we provide a dependable source of information for understanding the materials impact on solderability, tin whisker susceptibility and solder fatigue.

Pre-qualification efforts help build assurance that customers receive low-risk and reliable products. With predominant industry solutions-tin-silver-copper (SnAgCu) spheres for ball grid arrays and matte tin (Sn) plating for leaded devices—customers can be confident in Freescale's product quality.

products, Freescale offers a differentiator that many suppliers have overlooked. The transition to RoHS-compliant products will be eased by dual availability of both lead (Pb) containing products and RoHS products over a specified period of time. Packaging labels will specify RoHS-compliant status, MSL, PPT and second level interconnect to provide customer container level visibility of product compliance.

Specific actions are being taken by Freescale Semiconductor to make the customer's transition to RoHS compliance as seamless as possible. Freescale's present portfolio of products meets RoHS compliance for hexavalent chromium or PBB and PBDE, cadmium and mercury. Freescale continues to migrate production to lead (Pb) free packaging solutions.

RoHS-compliant products' distinct part numbers allow customers to manage inventories and conversions without monitoring date codes. RoHS-exempt applications (e.g., automotive and networking infrastructure) and customers not subject to RoHS compliance will be supported by RoHS-exempt products for a limited time. All new products are being introduced in RoHS-compliant, high-temperature, attach qualified packaging in January 2006. Availability schedules may be reviewed at **www.freescale.com/pbfree**. Last ship date plans for RoHS exempt leadframe products is scheduled for December 2007.

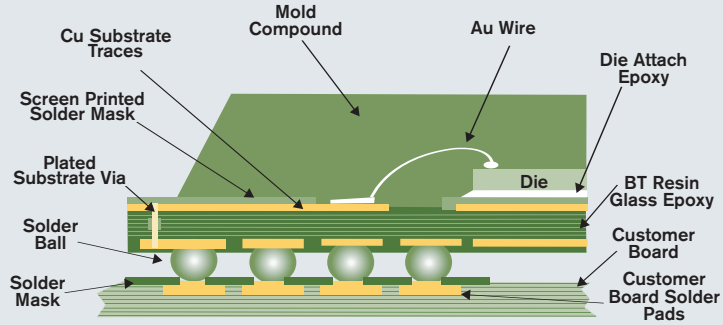
FIGURE 2. PRODUCT LABEL

EU Directive Requirements

Freescale continues its focus primarily on these European directives:

> RoHS [Restriction of Hazardous Substances] 2002/95/EC and ELV [End of life vehicles] 2000/53/EC
  - Sets restrictions on levels of designated hazardous substances: lead, cadmium, mercury, hexavalent chromium, polybrominated biphenyls (PBBs) and polybrominated diphenyl ethers (PBDEs).
> WEEE [Waste of Electrical and Electronic Equipment] 2002/96/EC
  - Mandates recovery of certain electrical/electronic equipment and sets recycling requirements.
> Packaging and Packaging Waste, ECM/2002/02
  - Controls the content of shipping materials and encourages reuse and recycling.

FIGURE 3. BALL GRID ARRAY CROSS SECTION



Our Commitment

Freescale is committed to our customers' success. We will continue to offer the same high-quality products and the high level of customer support our customers have come to expect throughout our 50 years of technology innovation.

Reference

Environmentally Preferred Products for RoHS Compliance (Brochure), November 2005.

Pattye Brown is a channel marketing manager for Freescale's 8-bit microcontrollers. She also teaches university-level undergraduate marketing courses.

Comments about this article?
e-Mail **MCUideas@freescale.com**.

Addendum

Full Peripheral Matrix

Serial Communications

Acronym	Peripheral Name	Functional Description/Features	Highlights	Common Uses/ Applications	MCUs	
SCI	Serial Communications Interface	Full-duplex, standard non-return-to-zero (NRZ) format Double-buffered transmitter and receiver with separate enables Programmable baud rates (13-bit modulo divider—S08 only) Interrupt-driven or polled operatio n Transmit data register empty and transmission complete Receive data register full Receive overrun, parity error, framing error and noise error Idle receiver detect Hardware parity generation and checking Programmable 8-bit or 9-bit character length Receiver wake-up by idle line or address mark Optional 13-bit break character (S08 only) Selectable transmitter output polarity Two-wire (pin) interface	Two-wire full-duplex, asynchronous communication	High-speed, asynchronous communications with peripheral devices and other MCUs Standard universal asynchronous receiver transmitter (UART) communications Conversion to RS-232 connection for serial communication with PC	**HC08s** **MC68HC908Axxx** **MC68HC908GR8/4** **MC68HC908GP32** **MC68HC908JK/JL8** **MC68HC908JL16** **MC68HC908KXx** **MC68HC908Lxxx** **MC68HC908MRxx** **MC68HC908SR12**	**S08s** **MC9S08Gxxx** **MC9S08RDxx** **MC9S08RExx** **MC9S08RGxx** **MC9S08QGx** **MC9S08AWxx**
ESCI	Enhanced Serial Communications Interface	Full-duplex operation Standard mark/space non-return-to-zero (NRZ) format Programmable baud rates Programmable 8-bit or 9-bit character length Separately enabled transmitter and receiver Separate receiver and transmitter interrupt requests Programmable transmitter output polarity Idle-line or address-mark receiver wake-up methods Interrupt-driven operation with eight interrupt flags Receiver framing error detection Hardware parity checking 1/16 bit-time noise detection Two-wire (pin) interface	Two-wire full-duplex, asynchronous communication Adds 13-bit break detect for software LIN emulation	High-speed, asynchronous communications with peripheral devices and other MCUs Standard universal asynchronous receiver transmitter (UART) communications Conversion to RS-232 connection for serial communication with PC Allows for software emulation of LIN protocol	**HC08s** **MC68HC908EYxx** **MC68HC908GR60/32/16** **MC68HC908GT60/32/16** **MC68HC908GZxx** **MC68HC908QBx**	
SPI	Serial Peripheral Interface	Master and slave mode operation Cost-effective communications module Full-duplex or single-wire bidirectional option Programmable transmit bit rate Double-buffered transmit and receive Serial clock phase and polarity options Selectable MSB-first or LSB-first shifting Maximum master mode frequency = bus frequency/2 Maximum slave mode frequency = bus frequency/4 Four-wire (pin) interface	Master and slave mode operation Four-wire, full-duplex or single-wire bidirectional operation Maximum master mode frequency = bus frequency/2 Maximum slave mode frequency = bus frequency/4	Full-duplex, synchronous serial communication with external peripherals Ex: high-resolution analog-to-digital converters (ADCs), serial EEPROM, chip-on-glass LCD screens and other MCUs Communication within PCB board at relative high speeds	**HC08s** **MC68HC908Axxx** **MC68HC908EYxx** **MC68HC908GRxx** **MC68HC908GTxx** **MC68HC08GZxx** **MC68HC908GP32** **MC68HC908JK/JL8** **MC68HC908JW32** **MC68HC908LJ/Lkxx** **MC68HC908MRxx** **MM908E625/624**	**S08s** **MC9S08Gxxx** **MC9S08RGxx** **MC9S08QGx** **MC9S08AWxx**
MMI_2C	Multi-Master Inter-Integrated Circuit	Compatibility with multimaster I_2C bus standard Software-controllable acknowledge bit generation Interrupt-driven byte-by-byte data transfer Calling address identification interrupt Auto-detection of R/W bit and switching of transmit or receive mode Detection of START, repeated START and STOP signals Auto generation of START and STOP condition in master mode Arbitration loss detection and No-ACK awareness in master mode Eight selectable baud rate master clocks Automatic recognition of the received acknowledge bit	Same features as I_2C, but allows system to have multiple Master I_2C nodes	Full-duplex, synchronous serial communication with external peripherals Ex: high-resolution analog-to-digital converters (ADCs), serial EEPROM, chip-on-glass LCD screens and other MCUs Communication within PCB board at relative high speeds (slightly slower than SPI) Can be used in I_2C system with multiple master devices	**HC08s** **MC68HC908APxx** **MC68HC908BDxx** **MC68HC908LDxx** **MC68HC908LJ/LKxx** **MC68HC908SR12**	

Acronym	Peripheral Name	Functional Description/Features	Highlights	Common Uses/ Applications	MCUs	
I_2C	Inter-Integrated Circuit	Master and slave mode operation Provides a method of communication between a number of devices Designed to operate at up to 100 kbps with maximum bus loading and timing Capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF Two-wire (pin) interface Individual I_2C modules can only transmit or receive at any given time (i.e., single-wire, bidirectional data transfer) Multiple I_2C modules can simultaneously receive a message that is transmitted on a bus	Master and slave mode operation Designed to operate at up to 100 kbps with maximum bus loading and timing Capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading Two-wire (pin) interface, half-duplex operation Individual I_2C modules can only transmit or receive at any given time (i.e., single-wire, bidirectional data transfer) Multiple I_2C modules can simultaneously receive a message that is transmitted on a bus	Synchronous serial communication with external peripherals Ex: high-resolution analog-to-digital converters (ADCs), serial EEPROM, chip-on-glass LCD screens and other MCUs Communication within PCB board at relative high speeds (slightly slower than SPI) Can be used in multinodal, serial bus communications		**S08s** **MC9S08Gxxx** **MC9S08QGx** **MC9S08AWxx**
SLIC	Slave LIN Interface Controller	Full LIN messaging buffering of Identifier and 8 data bytes Automatic baud rate and LIN message frame synchronization: – No prior programming of bit rate required, 1–20 kbps LIN bus-speed operation – All LIN messages will be received (no message loss due to synchronization process) – Input clock tolerance as high as ± 50 percent, allowing internal oscillator to remain untrimmed – Incoming break symbols allowed to be 10–20 bit times without message loss – Supports automatic software trimming of internal oscillator using LIN synchronization data Automatic processing an verification of LIN SYNCH BREAK and SYNCH BYTE Automatic checksum calculation and verification with error reporting Maximum of two interrupts per LIN message frame Full LIN error checking and reporting High-speed LIN capability up to 120 kbps at 8 MHz Configurable digital receive filter	Automates many LIN functions for lower CPU overhead when communicating on LIN bus Full LIN error checking and reporting High-speed LIN capability up to 120 kbps at 8 MHz	Designed to provide slave node connectivity on a local interconnect network (LIN) sub-bus LIN is an open-standard serial protocol developed for the automotive industry to connect sensors, motors and actuators	**HC08s** **MC908QLx**	
USB 1.1	Universal Serial Bus 1.0/1.1	Full universal serial bus specification 1.1 low-speed functions 1.5 Mbps data rate On-chip 3.3-volt regulator Endpoint 0 with 8-byte transmit buffer and 8-byte receive buffer Endpoint 1 with 8-byte transmit buffer Endpoint 2 with 8-byte transmit buffer and 8-byte receive buffer USB data control logic: – Control endpoint 0 and interrupt endpoints 1 and 2 – Packet decoding/generation – CRC generation and checking – NRZI (non-return-to zero inserted) encoding/decoding – Bit-stuffing USB reset options: – Internal MCU reset generation – CPU interrupt request generation Suspend and resume operations, with remote wakeup support USB-generated interrupts: – Transaction interrupt driven – Resume interrupt – End-of-packet interrupt – USB reset STALL, NAK and ACK handshake generation	Full universal serial bus specification 1.1 low-speed functions 1.5 Mbps data rate	Serial communication with common PC serial interface Low-speed communication (1.5 Mbps) Targeted to various PC peripheral applications (i.e., wireless mice dongles)	**HC08s** **MC68HC908BD24** **MC68HC908BD48** **MC68HC908JB8** **MC68HC908JB12** **MC68HC908JB16** **MC68HC908LD60** **MC68HC908LD64**	

USB 2.0	Universal Serial Bus 2.0	Full universal serial bus specification 2.0 full-speed functions 12 Mbps data rate On-chip 3.3-volt regulator Endpoint 0 with 8-byte transmit buffer and 8-byte receive buffer 64 bytes programmable buffer to share with four data endpoints Four data endpoints supports USB device controller with protocol control supports single configuration, two interfaces and no alternate settings for each interface Programmable endpoint type for four independent endpoints—interrupt or bulk USB data control logic: – Packet identification and decoding/generation – CRC generation and checking – NRZI (non-return-to zero inserted) encoding/decoding – Bit-stuffing – Sync detection – End-of-packet detection USB reset options: – Internal MCU reset generation – CPU interrupt request generation Suspend and resume operations, with remote wake-up support	Full universal serial bus specification 2.0 full-speed functions 12 Mbps data rate	Serial communication with common PC serial interface Full-speed communication (12 Mbps) Targeted to various PC peripheral applications (i.e., wireless mice dongles)	**HC08s** **MC68HC908JW32**	
DDC12AB	Display Data Channel 1 and Display Data Channel 2AB	DDC1 hardware Compatibility with multimaster I_2C bus standard Software-controllable acknowledge bit generation Interrupt-driven byte-by-byte data transfer Calling address identification interrupt Auto detection of R/W bit and switching of transmit or receive mode Detection of START, repeated START and STOP signals Auto-generation of START and STOP condition in master mode Arbitration loss detection and No-ACK awareness in master mode Eight selectable baud rate master clocks Automatic recognition of the received acknowledge bit	DDC1 hardware; also compliant with DDC2AB protocol Compatibility with multimaster I_2C bus standard	Digital monitor applications requiring VESA-certified DDC1 and DDC2AB communication standards	**HC08s** **MC68HC908BDxx** **MC68HC908LDxx**	

Analog Interfaces

Acronym	Peripheral Name	Functional Description/Features	Highlights	Common Uses/ Applications	MCUs	
ACMP	Analog Comparator	Full rail-to-rail supply operation Less than 40 mV of input offset Less than 15 mV of hysteresis Selectable interrupt on rising edge, falling edge and either rising or falling edges of comparator output Option to compare with fixed internal bandgap reference voltage Option to allow comparator output to be visible on a pin, ACMPO (S08QGx only)	Full rail-to-rail supply operation Selectable interrupt on rising edge, falling edge and either rising or falling edges of comparator output Option to compare with fixed internal bandgap reference voltage Option to allow comparator output to be visible on a pin, ACMPO (S08QGx only)	Emulate single-slope analog-to-digital conversion Combined with TPM and external RC circuit for temperature measurement Comparing external analog signal to internal reference When output is brought out to pin (not available on all parts), it may be possible to create an op-amp circuit	**HC08s** **MC68HC908LB8**	**S08s** **MC9S08RGxx** **MC9S08RCxx** **MC9S08RExx** **MC9S08QGx**
ATD	Analog-to-Digital Converter	8-/10-bit resolution 14.0 _sec, 10-bit single conversion time at a conversion frequency of 2 MHz Left-/right-justified result data Left-justified signed data mode Conversion-complete flag or conversion-complete interrupt generation Analog input multiplexer for up to eight analog input channels Single or continuous conversion mode	8- or 10-bit resolution Single or continuous conversion	Conversion of analog signal to digital Systems that need to measure temperature, voltage, pressure and motor speed Interface to sensor inputs		**S08s** **MC9S08Gxxx**

ADC (8-bit)	Analog-to-Digital Converter (8-bit)	Linear successive approximation with monotonicity 8-bit resolution Single or continuous conversion Conversion-complete flag or conversion-complete interrupt Selectable ADC clock frequency	Linear successive approximation with monotonicity 8-bit resolution Single or continuous conversion	Conversion of analog signal to digital Systems that need to measure temperature, voltage, pressure and motor speed Interface to sensor inputs	**HC08s** **MC68HC908ABxx** **MC68HC908ASxx** **MC68HC908AZxx** **MC68HC908BDxx** **MC68HC908GP/GTxx** **MC68HC908GR8/4** **MC68HC908KXx** **MC68HC908JK/JLxx** **MC68HC908LB8** **MC68HC908LDxx** **MC908Qxx**	
ADC (10-bit)	Analog-to-Digital Converter (10-bit)	Linear successive approximation with monotonicity 10-bit resolution Single or continuous conversion Conversion-complete flag or conversion-complete interrupt Selectable ADC clock Left- or right-justified result Left-justified sign data mode	Linear successive approximation with monotonicity 10-bit resolution Single or continuous conversion	Conversion of analog signal to digital Systems that need to measure temperature, voltage, pressure and motor speed Interface to sensor inputs	**HC08s** **MC68HC908APxx** **MC68HC908EYxx** **MC68HC908GR60/48/32/16** **MC68HC908GZxx** **MC68HC908LK/LJxx** **MC68HC908MRxx**	
ADC10	Analog-to-Digital Converter (10-bit)	Linear successive approximation algorithm with 10-bit resolution Output formatted in 10- or 8-bit right-justified format Single or continuous conversion (automatic power-down in single conversion mode) Configurable sample time and conversion speed (to save power) Conversion-complete flag and interrupt Input clock selectable from up to three sources Operation in WAIT and STOP modes for lower noise operation Selectable asynchronous hardware conversion trigger Ability to run in STOP3 and wake up MCU from STOP3 mode (S08s only)	Linear successive approximation algorithm with 10-bit resolution Output formatted in 10- or 8-bit right-justified format Single or continuous conversion (automatic power-down in single conversion mode) Operation in WAIT and STOP modes for lower noise operation Selectable asynchronous hardware conversion trigger Ability to run in STOP3 and wake up MCU from STOP3 mode (S08s only)	Conversion of analog signal to digital Systems that need to measure temperature, voltage, pressure and motor speed Interface to sensor inputs	**HC08s** **MC908QBx** **MC908QY8** **MC908QxxA**	**S08s** **MC9S08QGx** **MC9S08AWxx**

Other On-chip Peripherals

Acronym	Peripheral Name	Functional Description/Features	Highlights	Common Uses/ Applications	MCUs	
MTIM	Modulo Timer	Simple 8-bit timer with several software-selectable clock sources and a programmable interrupt Central component of the MTIM is the 8-bit counter, which can operate as a free-running counter or as a modulo counter A timer overflow interrupt can be enabled to generate periodic interrupts for time-based software loops	Simple 8-bit timer with several software-selectable clock sources and a programmable interrupt	Separate timebase to use for software interrupts		**S08s** **MCS08QGx**
TIM	Timer Interface Module	Two input capture/output compare channels – Rising-edge, falling-edge or any-edge input capture trigger – Set, clear or toggle output compare action Buffered and unbuffered output compare pulse-width modulation (PWM) signal generation Programmable clock input – Seven frequency internal bus clock prescaler selection – External clock input pin if available Free-running or modulo up-count operation Toggle any channel pin on overflow Counter stop and reset bits	Two input-capture/output-compare channels Buffered and unbuffered output compare pulse-width modulation (PWM) signal generation Free-running or modulo up-count operation Toggle any channel pin on overflow	Input capture, output compare and PWM functionality Motor/motion control Combine with RC filter for cheap digital-to-analog converter Driving piezos or LEDs Light dimming Battery charging	**All HC08s**	

TPM	Timer/Pulse-Width Modulator 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 prescaler for each TPM is independently selectable as bus clock, fixed system clock or an external pin (on some devices) Prescale taps for divide by 1, 2, 4, 8, 16, 32, 64, 128 Fixed system clock (XCLK) and pin paths are synchronized 16-bit free-running or up/down (CPWM) count operation 16-bit modulus register to control counter range Timer system enable One interrupt per channel plus terminal count interrupt	Each channel may be input capture, output compare or buffered edge-aligned PWM 16-bit free-running or up/down (CPWM) count operation	Input capture, output compare and PWM functionality Motor/motion control Combine with RC filter for cheap digital-to-analog converter Driving piezos or LEDs Light dimming Battery charging		**All S08s**
PWMMC	Pulse-Width Modulator—Motor Control	Three complementary PWM pairs or six independent PWM signals Edge-aligned PWM signals or center-aligned PWM signals PWM signal polarity control 20 mA current sink capability on PWM pins Manual PWM output control through software Programmable fault protection Complementary mode featuring: – Dead-time insertion – Separate top/bottom pulse width correction via current sensing or programmable software bits	Three complementary PWM pairs or six independent PWM signals 20 mA current sink capability on PWM pins Programmable fault protection Complementary mode featuring: – Dead-time insertion – Separate top/bottom pulse-width correction via current sensing or programmable software bits	12-bit PWM for motor control applications (AC motor control, in particular)	**HC08s** **MC68HC908MRxx**	
RTC	Real-Time Clock	Real-time clock (RTC) with clock, calendar, alarm and chronograph functions. Selectable periodic interrupt requests for seconds, minutes, hours, days, 2 Hz, 4 Hz, 8 Hz, 16 Hz, 128 Hz	Real-time clock (RTC) with clock, calendar, alarm and chronograph functions Selectable periodic interrupt requests for seconds, minutes, hours, days, 2 Hz, 4 Hz, 8 Hz, 16 Hz, 128 Hz	Used for applications that require keeping track of time/date Calendar, alarm and chronograph functions	**HC08s** **MC68HC908LK/LJxx**	
XIRC	Crystal or Internal Resonator Circuit	The bus clock frequency is one fourth of any of these clock source options: 1. Internal oscillator: An internally generated, fixed frequency clock, trimmable to ± 0.4 percent. There are three choices for the internal oscillator: 12.8 MHz, 8 MHz or 4 MHz; the 12.8 MHz internal oscillator is the default option out of reset 2. External oscillator: An external clock that can be driven directly into OSC1 3. External RC: A built-in oscillator module (RC oscillator) that requires an external R connection only; the capacitor is internal to the chip 4. External crystal: A built-in XTAL oscillator that requires an external crystal or ceramic-resonator; there are three crystal frequency ranges supported: 8–32 MHz, 1–8 MHz and 32–100 kHz.	The bus clock frequency is one fourth of any of these clock source options: 1. Internal oscillator (trimmable) 2. External oscillator 3. External RC 4. External crystal	Allows for use of external clock source or trimmable internal clock source	**HC08s** **MC68HC908Qxx** **MC68HC908LB8**	

Acronym	Peripheral Name	Functional Description/Features	Highlights	Common Uses/ Applications	MCUs	
ICG	Internal Clock Generator	Provides multiple options for clock sources, which offers a user great flexibility when making choices between cost, precision, current draw, and performance Consists of four functional blocks: - Oscillator block - Internal reference generator - Frequency-locked loop (FLL) - Clock select block	Provides multiple options for clock sources; consists of four blocks: - Oscillator block - Internal reference generator - Frequency-locked loop - Clock select block	Internal clock source that allow pins to be used as I/O Eliminates need and cost of external clock components	**HC08s** **MC68HC908GTxx** **MC68HC908KXx** **MC68HC908EYxx**	**S08s** **MC9S08Gxxx** **MC9S08AWxx**
ICS	Internal Clock Source	Provides clock source choices for the MCU Contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock Can provide this FLL clock or either of the internal or external reference clocks as a source for the MCU system clock There are also signals provided to control a low-power oscillator (XOSC) module to allow the use of an external crystal/resonator as the external reference clock	Provides clock source choices for the MCU Contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock There are also signals provided to control a low-power oscillator (XOSC) module to allow the use of an external crystal/resonator as the external reference clock	Low-power internal clock source that allow pins to be used as I/O Eliminates need and cost of external clock components		**S08s** **MC9S08QGx**
LCD	Liquid Crystal Display Driver (segment-based)	Software-programmable driver segment configurations LCD bias voltages generated by internal resistor ladder Software programmable contrast control	Software-programmable driver segment configurations LCD bias voltages generated by internal resistor ladder Software-programmable contrast control	Segment-based display applications	**HC08s** **MC68HC908LJ/LKxx**	
CMT	Carrier Modulator Timer			Remote-control applications		**S08s** **MC9S08Rxxx**
CAN	Controller Area Network Interface Module	Modular architecture Implementation of the CAN Protocol—version 2.0A/B – Standard and extended data frames – 0–8 bytes data length. – Programmable bit rate up to 1 Mbps, depending on the actual bit timing and the clock jitter of the phase-locked loop (PLL) Support for remote frames Double-buffered receive storage scheme Triple-buffered transmit storage scheme with internal prioritization using a "local priority" concept Flexible maskable identifier filter supports alternatively one full-size extended identifier filter or two 16-bit filters or four 8-bit filters Programmable wake-up functionality with integrated low-pass filter Programmable loop-back mode supports self-test operation Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus off) Programmable MSCAN08 clock source either CPU bus clock or crystal oscillator output Programmable link to timer interface module 2 channel 0 for time-stamping and network synchronization Low-power sleep mode	Modular architecture Implementation of the CAN Protocol—version 2.0A/B	Automotive/Industrial applications that uses the CAN network protocol	**HC08s** **MC68HC908AZxx** **MC68HC908GZxx**	

HR-PWM	High-Resolution Pulse-Width Modulator	One complementary output pair for driving a half bridge Dithering between two frequencies or duty cycles for increased output resolution Automatic calculation of second frequency or duty cycle for output dithering Variable frequency mode with automatic 50-percent duty cycle calculation Variable duty cycle mode Programmable deadtime insertion Shutdown input for fast disabling of outputs	One complementary output pair for driving a half bridge Dithering between two frequencies or duty cycles for increased output resolution Programmable deadtime insertion Shutdown input for fast disabling of outputs	Targeted for applications that require power factor correction Motor control	**HC08s** **MC68HC908LB8**	
LVI	Low-Voltage Inhibit	Programmable LVI reset Selectable LVI trip voltage Programmable STOP mode operation	Programmable LVI reset	Overall system safety; prevents micro from operating when power supply is too low (allowing corruption of code execution)	**All HC08s**	
LVD	Low-Voltage Detect	Selectable RESET or interrupt off LVD trip point	Selectable RESET or interrupt off LVD trip point	Overall system safety; prevents micro from operating when power supply is too low (allowing corruption of code execution) Generating an interrupt instead of resetting the part; allows battery-operated systems to alert user of decaying batteries		**All S08s**

Freescale's 8-bit Products Summary

Device	Flash	RAM	USB	ADC Channels 10-bit	ADC Channels 8-bit	SCI	ESCI	SPI	IIO	ACMP	Timer	Clock Type	Package DFN/QFN	Package QFP/LQFP	Package TSSOP	Package SOIC	Package DIP	Dev Tools DEMO	Dev Tools EVB	Dev Tools PSICE	Applications/Additional Features
General Purpose Products																					
HCSO8 & RSO8 Families																					
MC9S08AW32	60 KB	2 KB		16		2		√	√		6+2ch	ICG w/FLL	48	64, 44				√			High integration, Flash programmable to 5V
MC9S08AW32	32 KB	2 KB		16		2		√	√		6+2ch	ICG w/FLL	48					√			High integration, Flash programmable to 5V
MC9S08AW16	16 KB	1 KB		16		2		√	√		4+2ch	ICG w/FLL	48					√			High integration, Flash programmable to 5V
MC9S08GB60A	60 KB	4 KB		8		√		√	√		3+5ch	ICG		64				√	√		High performance, Flash programmable down to 1.8V
MC9S08GT60A	60 KB	4 KB		8		√		√	√		2+2ch	ICG	48	44				√	√		High performance, Flash programmable down to 1.8V
MC9S08GB32A	32 KB	2 KB		8		√		√	√		3+5ch	ICG		64				√	√		High performance, Flash programmable down to 1.8V
MC9S08GT32A	32 KB	2 KB		8		√		√	√		2+2ch	ICG	48	44				√	√		High performance, Flash programmable down to 1.8V
MC9S08GT16	16 KB	2 KB		8		√		√	√		2+2ch	ICG	48	44			42	√	√		High performance, low voltage
MC9S08QG8	8 KB	512B		8		√		√	√	√	2 ch	ICS	8, 16		16	8	16	√			High performance, low voltage, small package
MC9S08QG4	4 KB	256B		8		√		√	√	√	2 ch	ICS	8, 16		16	8	8	√			High performance, low voltage, small package
MC9RS08KA2 *New!*	**2 KB**	**62 B**								**√**	**MTIM**	**ICS**	**6**			**8**	**8**	**√**			**Ultra-low end, new RS08 core for small MCUs**
MC9RS08KA1 *New!*	**1 KB**	**62 B**								**√**	**MTIM**	**ICS**	**6**			**8**	**8**	**√**			**Ultra-low end, new RS08 core for small MCUs**
HC08 Family																					
MC908AP64	60 KB	2 KB		8		√		√	√		2+2ch	PLL		48, 44			42	√		√	Pin compatible from 8 to 62 KB
MC908GR60A	60 KB	2 KB		24			√	√			2+6ch	PLL		64, 48, 32				√		√	24 analog inputs & increased RAM
MC908GR48A	48 KB	1.5 KB		24			√	√			2+6ch	PLL		64, 48, 33				√		√	24 analog inputs & increased RAM
MC908AP32	32 KB	2 KB		8		√		√	√		2+2ch	PLL		48, 44			42	√		√	Pin compatible from 8 to 62 KB
MC908GP32	32 KB	512B			8	√		√			2+2ch	PLL		44			40, 42		√	√	2, 2-ch timers
MC908GR32A	32 KB	1.5 KB		24			√	√			2+6ch	PLL		64, 48, 32				√		√	24 analog inputs & increased RAM
MC908AB32	32 KB	1 KB			8	√		√			4+4ch	PLL		64						√	Embedded EEPROM (512B) & add'l timer channels
MC908AP16	16 KB	1 KB		8		√		√	√		2+2ch	PLL		48, 44			42	√		√	Pin compatible from 8 to 62 KB
MC908GT16	16 KB	512B			8		√	√			2+2ch	ICG		44			42			√	Internal clock
MC908GR16	16 KB	1 KB		8			√	√			2+2ch	PLL		48, 32				√		√	High resolution ADC; supporting 32-kHz to 100-kHz crystals
MC908GR16A	16 KB	1 KB		8			√	√			2+2ch	PLL		48, 32				√		√	High resolution ADC; supporting 1-MHz to 8-MHz crystals
MC908JL16 *New!*	**16 KB**	**512 B**		**13**		**√**			**√**		**2+2ch**	**OSC**		**32**		**28**	**28, 32**	**√**		**√**	**Add Memory and IIC**
MC908QC16	16KB	512B		10			√	√			4+2ch	OSC			28, 20, 16	28, 20, 16		√		√	High pin count with add'l timer ch
MC908JK8	8 KB	256B			13	√					2+2ch	OSC				20	20			√	Low pin count
MC908JL8	8 KB	256B			13	√					2+2ch	OSC		32		32, 28	28			√	Low pin count, more analog channels
MC908GT8	8 KB	512B			8		√	√			2+2ch	ICG		44			42			√	Internal clock
MC908QB8	8 KB	256B		10			√	√			4ch	OSC			16	16	16	√		√	Analog resolution, extra timers, small packages
MC908QC8	8 KB	384B		10			√	√			4+2ch	OSC			28, 20, 16	28,20, 16		√		√	High pin count with add'l timer ch
MC908QY8	8 KB	256B		10							2ch	OSC			16	16	16	√		√	Small packages
MC908QB4	4 KB	128B		4			√	√			4ch	OSC			16	16	16	√		√	Small packages, extra timers
MC908JL3E	4 KB	128B			12						2ch	OSC		48		28	28			√	Low pin count
MC908JK3E	4 KB	128B			12						2ch	OSC				20	20			√	Low pin count
MC908QT4A	4 KB	128B		6							2ch	OSC	8			8	8	√		√	Small packages
MC908QY4A	4 KB	128B		6							2ch	OSC			16	16	16	√		√	Small packages
MC68HC908JK1E	1.5 KB	128B			12						2ch	OSC				20	20			√	Low pin count
MC908QT2A	1.5 KB	128B		6							2ch	OSC	8			8	8	√		√	Small packages
MC908QY2A	1.5 KB	128B		6							2ch	OSC			16	16	16	√		√	Small packages
MC908QT1A	1.5 KB	128B									2ch	OSC	8			8	8	√		√	Small packages
MC908QY1A	1.5 KB	128B									2ch	OSC			16	16	16	√		√	Small packages

*All HC08 and08 include COP, LVI, POR and KBI

Part Number	Flash	RAM	USB	ADC Channels 10-bit	ADC Channels 8-bit	SCI	ESCI	SPI	IIO	ACMP	Timer	Clock Type	Package DFN/QFN	Package QFP/LQFP	Package TSSOP	Package SOIC	Package DIP	Dev Tools DEMO	Dev Tools EVB	Dev Tools PSICE	Applications/Additional Features
Application-Specific Products																					
HCS08 Family																					
MC9S08RC60	60 KB	2 KB								√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RD60	60 KB	2 KB				√					2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RE60	60 KB	2 KB				√				√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RG60	60 KB	2 KB				√		√		√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RC32	32 KB	2 KB								√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RD32	32 KB	2 KB				√					2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RE32	32 KB	2 KB				√				√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RG32	32 KB	2 KB				√		√		√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RC16	16 KB	1 KB								√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RD16	16 KB	1 KB				√					2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RE16	16 KB	1 KB				√				√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08JR1 ***New!***	**12 KB**	**256 B**									**1+1ch**	**OSC**		**48**							**Integrated RF, 27 MHz FSK (generator & transmitter)**
MC9S08RC8	8 KB	1 KB								√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RD8	8 KB	1 KB				√					2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
MC9S08RE8	8 KB	1 KB				√				√	2ch	OSC		44, 32		28	28	√			Remote Control, Carrier Modulator Timer
HC08 Family																					
MC908AZ60A	60 KB	2 KB			15	√		√			2+6ch	PLL		64						√	Auto/Industrial Communication, CAN, EEPROM (1KB)
MC908AS60A	60 KB	2 KB			15	√		√			6ch	PLL		64						√	Auto/Industrial Communication, EEPROM (1KB)
MC908GZ60	60 KB	2 KB		24			√	√			2+6ch	PLL		64, 48, 32				√		√	Auto/Industrial Communication, CAN
MC908GZ48	48 KB	1.5 KB		24			√	√			2+6ch	PLL		64, 48, 32				√		√	Auto/Industrial Communication, CAN
MC908AZ32A	32 KB	1 KB			15	√		√			6ch	PLL		64						√	Auto/Industrial Communication, CAN, EEPROM (512B)
MC908AS32A	32 KB	1 KB			15	√		√			6ch	PLL		64						√	Auto/Industrial Communication, EEPROM (512B)
MC908GZ32	32 KB	1.5 KB		24			√	√			2+6ch	PLL		64, 48, 32				√		√	Auto/Industrial Communication, CAN
MCHC908JW32	32 KB	1 KB	2.0					√			2ch	PLL		48						√	USB
MC908MR32	32 KB	768B		10		√		√			2+4ch	PLL		64			56			√	Motor Control, 6-ch 12-bit PWM
MC908LJ24	24 KB	768B		6		√		√	√		2ch	PLL		80, 64						√	LCD
MC908LK24	24 KB	768B		6		√		√	√		2ch	PLL		80, 64						√	LCD
MC908GZ16	16 KB	1 KB		8			√	√			2+2ch	PLL		48, 32				√		√	Auto/Industrial Communication, CAN
MC908EY16	16 KB	512B		8			√	√			2+2ch	PLL		32						√	Auto/Industrial Communication
MC908JB16	16 KB	384B	1.0, 1.1			√					2+2ch	PLL		32		28, 20				√	USB
MC908MR16	16 KB	768B				√		√			2+4ch	PLL		64			56			√	Motor Control, 6-ch 12-bit PWM
MC908LJ12	12 KB	512B		6		√		√			2ch	PLL		64, 52						√	LCD
MC908JB12	12 KB	384B	1.0, 1.1			√					2+2ch	PLL				28, 20				√	USB
MC908JB8	8 KB	256B	1.1								2+2ch	OSC		44		28, 20	20			√	USB, ROM available
MC908LB8	8 KB	128B			7						2ch	OSC				20	20	√		√	Lighting, High-Resolution PWM
MC908GZ8	8 KB	512B		8			√	√			2+2ch	PLL		48, 32				√		√	Auto/Industrial Communication, CAN
MC908EY8	8 KB	384B		8			√	√			2+2ch	PLL		32						√	Auto/Industrial Communication
MC908MR8	8 KB	256B		7		√					2+2ch	PLL		32		28	28			√	Motor Control, 6-ch 12-bit PWM
MC908LV8	8 KB	512B		6							2ch	OSC		52						√	LCD
MC908QL4	4 KB	128B		6							2ch	OSC			16	16			√	√	Auto/Industrial Communication, SLIC (LIN)
MC908QL3	4 KB	128B									2ch	OSC			16	16			√	√	Auto/Industrial Communication, SLIC (LIN)
MC908QL2	2 KB	128B		2							2ch	OSC			16	16			√	√	Auto/Industrial Communication, SLIC (LIN)
MM908E626	16 KB	512B		8			√	√			2+2ch	ICG				54					Stepper Moter, Integrated Vreg, LIN, PHY, 4 Half-Bridge
MM908E625	16 KB	512B		8			√	√			2+2ch	ICG		32							Lighting, Integrated VReg and LIN PHY, KBI
MM908E624	16 KB	512B		8			√	√			2+2ch	ICG		32							Motor Control, Integrated VReg and LIN PHY, KBI
MM908E621	16 KB	512B		8			√	√			2+2ch	ICG				54					Integrated Quad Half-Bridge & Triple High-Side, LIN

*All HC08 and08 include COP, LVI, POR and KBI