

Designing for Migration Among 8-Pin, 8-Bit MCUs

MC9S08QG8, MC9S08QD4, and MC9RS08KA2

by: Inga Harris
8-bit Microcontroller Applications Engineer
East Kilbride, Scotland

Today's designs demand flexibility, thereby adding functionality, improving performance, reducing power consumption, and increasing complexity. As soon as one version of a product is complete, there are already many ideas for features that will improve it in the next version. In many cases, the selected device cannot perform to meet the requirements of the next idea and a trade-off begins between the cost of changing components and the desired functionality.

Portability is an important factor to consider at the outset of a project. The best way to speed the creation of a new project is to re-use the information and experience generated from previous developments. The goal is to find the best way to write an algorithm so that it is portable among devices. Typically, software tools such as compilers and code generation tools dictate the software development phase whereas hardware compatibility is a "nice to have." The microcontroller is often changed when the next generation is designed. The number of pins and the peripherals associated with each

Contents

1	Compatibility Among Low-End Devices.....	2
1.1	Highly Integrated Low-End MCUs.....	2
1.2	Ultra Low-End MCUs.....	3
1.3	Filling the Gap Between Highly Integrated Low-End and Ultra Low-End.....	4
1.4	Feature Comparisons.....	4
1.5	Pin-To-Pin Compatibility.....	12
1.6	Migration Software Tools.....	14
2	Conclusion.....	16

pin also changes. In this case, the same hardware platform cannot be used to test new designs and new hardware is needed for the new product.

The ability to move effortlessly from one hardware platform to another reduces the development, testing, and manufacturing time. Freescale's Controller Continuum Strategy provides new devices that are pin-to-pin compatible with existing microcontrollers, allowing hardware platforms to be re-used and opening new possibilities to use both more peripherals and functionality or simplifying the design to improve the cost of the final product. Migrating between MCUs is easier to plan for at the low end of the controller continuum spectrum when the MCU is an 8-pin, 8-bit micro with small code size (up to 8K bytes of program space) because the pin count limits the number of peripherals a project will use. If designed correctly migrating the software and hardware is much simpler.

1 Compatibility Among Low-End Devices

When designers look for devices in low-end applications they find some manufacturers do not offer compatible microcontrollers across different families. Therefore, switching from one device to another without changing the hardware and software becomes complicated.

Redesigning the hardware or software means a new set of tests must be developed and run on the MCU, and every component the MCU interacts with during the project. This is usually time-consuming and expensive. To quickly migrate from one MCU to another it is important for devices to be pin-to-pin compatible and share the same developing platforms, such as hardware programmers, debugging interfaces, and software tools.

1.1 Highly Integrated Low-End MCUs

The S08 family of microcontrollers focuses on bringing high-performance to low-end applications without sacrificing low-power operation. The MC9S08QG8 is a member of the S08 family that includes numerous peripherals in 8- and 16-pin packages.

The S08 architecture allows developing a design fully programmed in C because the characteristics of the stack handling and its instructions are optimized for data and array handling.

The Background Debug Controller (BDC) provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip Flash and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the S08 (and RS08 family described in [Section 1.2, "Ultra Low-End MCUs"](#)), address and data bus signals are not available on external pins. Debug is done through commands fed into the target MCU via the single-wire background debug interface. The debug module provides a means to selectively trigger and capture bus information so an external development system can reconstruct activity inside the MCU on a cycle-by-cycle basis without having external access to the address and data signals.

[Figure 1](#) shows the MC9S08QG8 block diagram. Of the three MCUs discussed in this document, the MC9S08QG8 is the most feature-rich and is available in 8- and 16-pin packages.

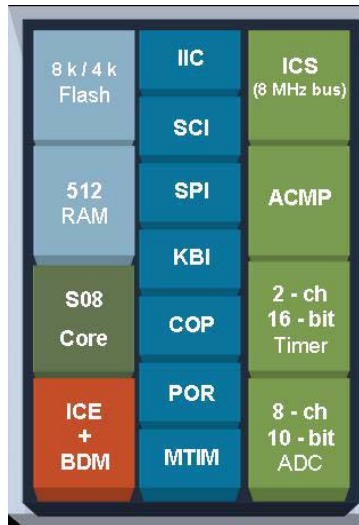


Figure 1. MC9S08QG8

1.2 Ultra Low-End MCUs

The RS08 architecture is optimized for small memory sizes. It includes a tiny addressing mode on which the instruction's address is embedded within the instruction to make the code efficient. Placing the most computation-intensive data in this area helps the user create more optimized systems in both code size and execution speed. The ability to use indexed addressing gives the possibility to access different operand locations depending on the results of earlier program instructions. The BDC uses single-wire communication to program the memory and debug the code in the MCU. A major improvement in the interface is that the code can be debugged in-circuit, using the actual clock source and connected hardware used in the final design, thereby eliminating the possibility of the debug interface causing a problem.

The new instructions and addressing modes are intended for small memory sizes and all the features and peripherals allow a cost-effective design when working with the RS08.

Figure 2 shows a MC9RS08KA2 block diagram. The MC9RS08KA2 is the lowest end MCU discussed here. It is available in a 6-pin package for applications that are particularly space constrained and in an 8-pin package for applications needing more I/O.

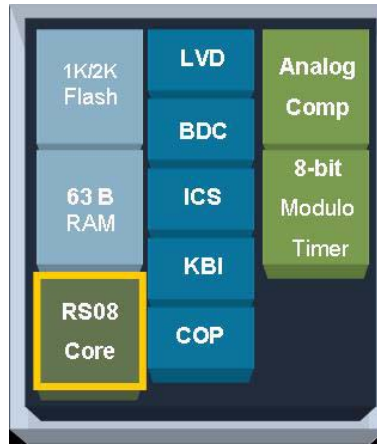


Figure 2. MC9RS08KA2

1.3 Filling the Gap Between Highly Integrated Low-End and Ultra Low-End

The midpoint between the ultra low-end MC9RS08KA2 and the highly integrated low-end device MC9S08QG8 is the MC9S08QD4. It has the same single-wire BDC module as the MC9S08QG8.

Figure 3 shows the block diagram of the MC9S08QD4.

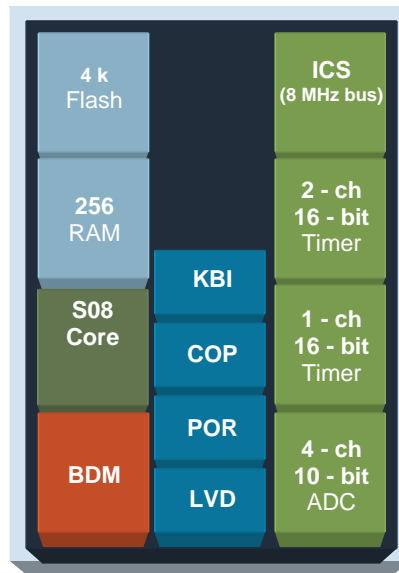


Figure 3. MC9S08QD4

1.4 Feature Comparisons

Table 1 shows the three microcontrollers’ features in a side-by-side comparison. By being aware of the differences before design starts and acknowledging that a change in MCU could occur during your project, you can design your hardware and software to be portable.

Table 1. MCU Feature Comparisons

	MC9S08QG		MC9S08QD	MC9RS08KA	
	QG4	QG8	QD4	KA1	KA2
Core	S08		S08	RS08	
Bus Frequency	10 MHz		8MHz	10MHz	
Development Support	DBDC = DBG		BDC	BDC	
Power Supply	1.8 to 3.6 V		2.7 to 5.5 V	1.8 to 5.5 V	
External Oscillator	8-pin: No		No	No	
	16-pin: Yes				
RAM (bytes)	256	512	256	63	63
Flash (bytes)	4K	8K	4K	1K	2K
Modes	Run, Wait, Stop1, Stop2, Stop3, and Active BDM		Run, Wait, Stop2, Stop3, and Active BDM	Run, Wait, Stop3, and Active BDM	
COP/LVI	Yes		Yes	Yes	
ACMP	Yes		No	Yes	
ADC	8-pin: 4-ch 10-bit		4-ch 10-bit	No	
	16-pin: 8-ch 10-bit				
ICS	Yes		Yes	Yes	
KB1	8-pin: 1-ch		8-pin: 4-ch	6-pin: 3-ch	
	16-pin: 8-ch			8-pin: 5-ch	
MTIM	Yes		No	Yes	
SCI	Yes		No	No	
SPI	Yes		No	No	
IIC	Yes		No	No	
TPM1	8-pin: 1-ch 16-bit		2-ch 16-bit	No	
TPM2	No		1-ch 16-bit	No	
Packages	8-pin: DIP, SOIC, DFN		8-pin: DIP, SOIC	6-pin: DFN	
	16-pin: DIP, TSSOP, QFN			8-pin: DIP, SOIC	

1.4.1 CPU

The first difference between the MCUs is that the RS08 CPU has a reduced instruction set (compared to the S08 CPU), a 14-bit address bus (versus a 16-bit address bus on the S08 devices), and a reduced set of core registers (see [Figure 4](#)). However, CodeWarrior is capable of compiling most of the S08 instructions as pseudo-instruction. The Freescale document, AN3266, “Getting Started with RS08,” is a good reference if a move from S08 to RS08 is forseen.

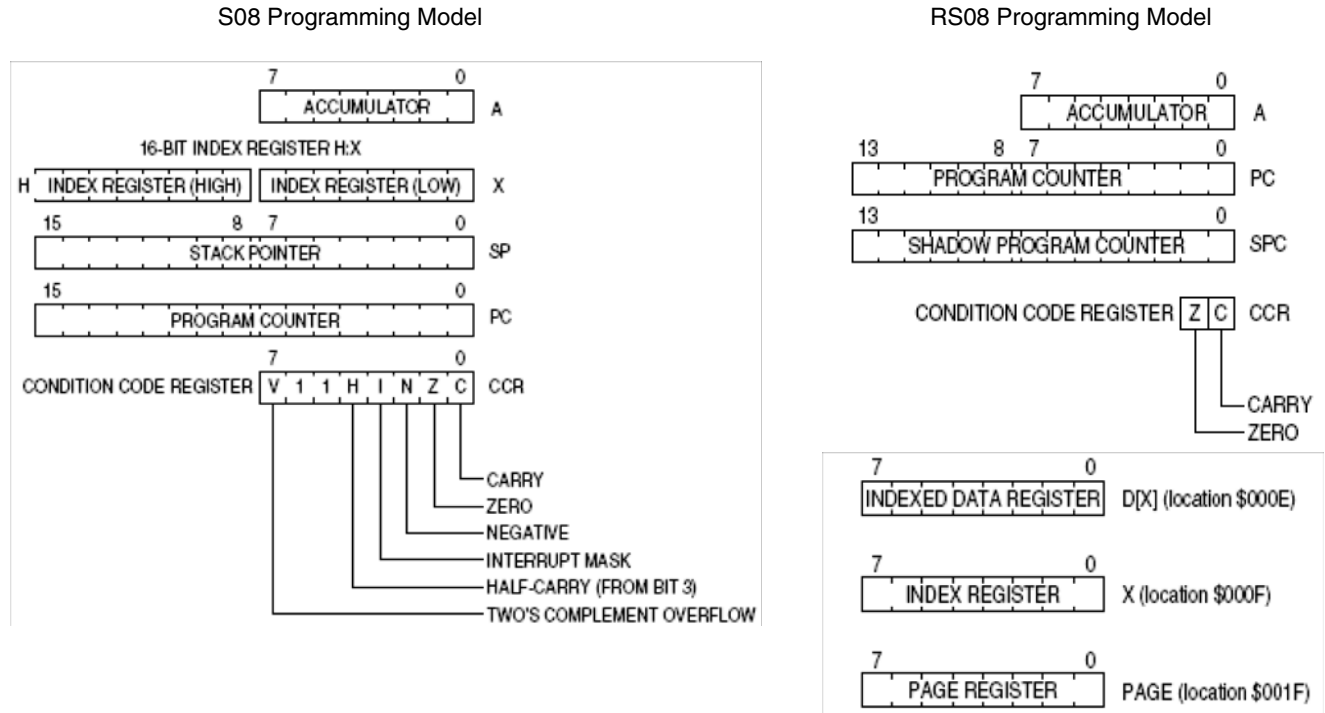


Figure 4. S08 and RS08 Programming Models

If project development dictates a move from the MC9RS08KA2 to one of the S08 devices you should be aware that any SHA and SLA instructions will not compile, but are also unnecessary as the stack is present. The designer may wish to use the full S08 instruction set for efficient code usage. All commands that deal with the RS08 paging can also be removed.

1.4.1.1 Addressing Modes

The MC9RS08KA2 does not support extended, stack, or offset indexed addressing modes that are supported on the S08 devices. By avoiding these modes when developing with the S08 device the migration to the RS08 will be smoother; however, if you cannot avoid using these modes it can be worked around. By adding in these modes when moving to an S08 device the designer can made more efficient use of the CPU.

Extended addressing is recovered on the RS08 by using paging;

```
HCS08 — LDA extended
RS08  — MOV#HIGH_6_13(extended),PAGESEL
        LDA    MAP_ADDR_6(extended)
```

The stack can be emulated to recover the stack addressing mode with the addition of the shadow program counter and the SHA and SLA instructions.

The same functions can be performed by using indirect addressing in place of offset indexed addressing (see Figure 5).

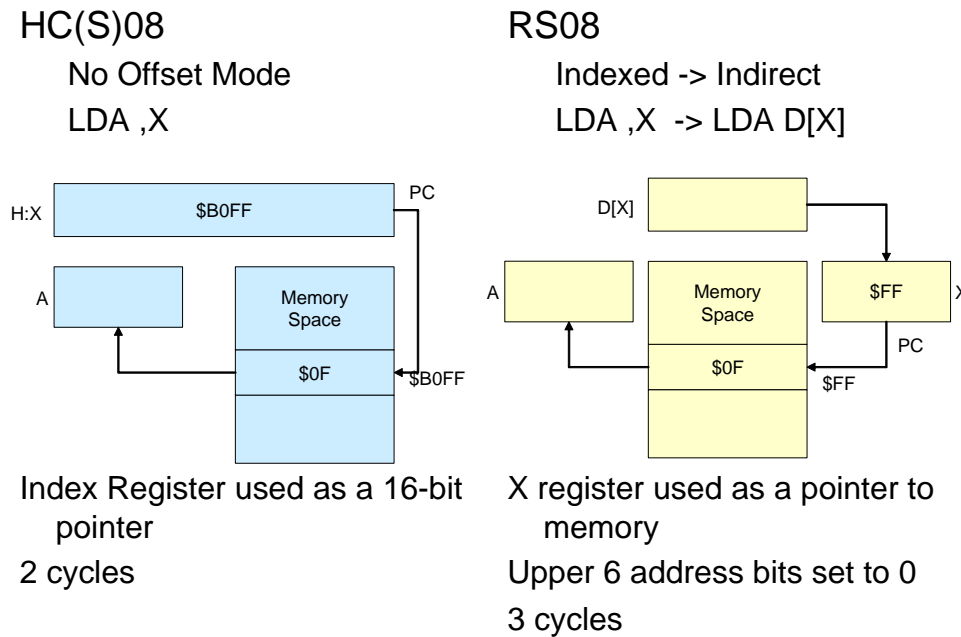


Figure 5. Indexed Addressing on the HCS08 and RS08

1.4.1.2 Instructions Not Supported

Table 2 shows the non-addressing mode restricted S08 instructions that are not supported on the RS08 architecture.

Table 2. Non-supported instructions

Type	Instructions
Interrupt	BIL, BIH, BMC, BMS, CLI, SEI, RTI, SWI
BCD	BHCC, BHCS, DAA, NSA
Maths	ASR, DIV, MUL
2's Compliment	BLT, BMI, BGE, BGT, BLE, BPL, NEG
Condition Code	TAP, TPA

1.4.2 Development Support

The MC9S08QD4 and MC9RS08KA2 do not have the on-chip DGB module. This means that these devices have only 21 commands versus 30 on the MC9S08QG8, and only one hardware breakpoint. More details on the differences between debugging with and without the DBG module can be found in Freescale document AN2497, “HCS08/RS08 Background Debug Mode versus HC08 Monitor Mode.”

1.4.3 Power Supply

The V_{DD} ratings of the three devices are a key factor when selecting an MCU. All devices will work with a 3 V supply. The MC9S08QG8 and MC9RS08KA2 will operate down to 2 V and the MC9S08QD4 and MC9RS08KA2 will operate at 5 V.

1.4.4 IRQ

The MC9S08QG8 and MC9S08QD4 have the IRQ (interrupt request) pin which manages external interrupts with the IRQ status and control register, IRQSC.

As the RS08 core has reduced interrupt handling capability due to the removal of the vector lookup mechanism, stack and index register, the IRQ function is not supported. Interrupts must be handled by polling in software. How to do so is described in the Freescale document, RS08QRUG, “RS08 Peripheral Module Quick Reference.” If migrating from the MC9RS08KA2 to an S08 device, any reference to the system interrupts pending register (SIP1) used in the polling must be adjusted to reference the corresponding interrupt flags on each modules register set and vice versa.

1.4.5 System Options

The write-once system option register (SOPT) is different between the devices. If reducing features it will not harm the MCU to write to unimplemented bits, but if increasing features the designer must write all required bits. The designer will also find that the MC9RS08KA2 has only one SOPT register whereas the S08 devices have SOPT1 and SOPT2.

1.4.6 Oscillator

Only the 16-pin device of the MC9S08QG8 family has the ability to work with an external oscillator. As the QD and KA are limited to 8-pin, this feature can only be used when moving up the feature set. The internal oscillators are controlled by the ICS module.

1.4.7 ICS

The ICS module on the MC9S08QG8 is different from that on the MC9S08QD4 and MC9RS08KA2 as it has the added function of incorporating the external oscillator component ability mentioned in [Section 1.4.6, “Oscillator.”](#) However, the modules have been designed to be compatible. If the project moves from the MC9S08QG8 using the internal reference clock, the CLKS bits in ICSC1 take care of themselves because bit 7 will ignore any attempt at writing and bit 6 has the same function. Any write to RDIV, IREFS, or IRCLKEN will also be ignored by the ICS on the MC9S08QD4 and MC9RS08KA2. The IREFSTEN has the same functionality and position so the previously written code will run. The same can be said for ICSC2, ICSSC, and the trim register. If the project is moving from MC9S08QD4 or MC9RS08KA2 to MC9S08QG8, the designer must ensure that all register bits are initialized.

[Table 3](#) shows the ICS registers for the three MCUs.

Table 3. ICS Registers

MC9S08QG8 ICS Registers										
ICSC1	R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN	
	W									
	RESET	0	0	0	0	0	1	0	0	
ICSC2	R	BDIV		RANGE	HG0	LP	EREFS	ERCLKEN	EREFSTEN	
	W									
	RESET	0	1	0	0	0	0	0	0	
ICSTRM	R	TRIM								
	W									
	POR	1	0	0	0	0	0	0	0	
	RESET	U	U	U	U	U	U	U	U	
ICSSC	R	0	0	0	0	CLKST		OSCINIT	FTRIM	
	W									
	POR	0	0	0	0	0	0	0	0	
	RESET	0	0	0	0	0	0	0	U	
MC9S08QD4 ICS Registers										
ICSC1	R	0	CLKS	0	0	0	1	1	IREFSTEN	
	W									
	RESET	0	0	0	0	0	1	1	0	
ICSC2	R	BDIV		0	0	LP	0	0	0	
	W									
	RESET	0	1	0	0	0	0	0	0	
ICSTRM	R	TRIM								
	W									
	POR	1	0	0	0	0	0	0	0	
	RESET	U	U	U	U	U	U	U	U	
ICSSC	R	0	0	0	0	0	CLKST	0	FTRIM	
	W									
	POR	0	0	0	0	0	0	0	0	
	RESET	0	0	0	0	0	0	0	U	
MC9RS08KA2 ICS Registers										
ICSC1	R	0	CLKS	0	0	0	0	0	IREFSTEN	
	W									
	RESET	0	0	0	0	0	0	0	0	

Table 3. ICS Registers (continued)

ICSC2	R	BDIV		0	0	LP	0	0	0
	W								
	RESET	0	1	0	0	0	0	0	0
ICSTRM	R	TRIM							
	W								
	POR	1	0	0	0	0	0	0	0
	RESET	U	U	U	U	U	U	U	U
ICSSC	R	0	0	0	0	0	CLKST	0	FTRIM
	W								
	POR	0	0	0	0	0	0	0	0
	RESET	0	0	0	0	0	0	0	U

1.4.8 Modes of Operation

All three microcontrollers have Run, Wait, Stop3, and Active BDM modes, but the MC9S08QD4 does not have Stop1 and the MC9RS08KA2 does not have Stop1 or Stop2 available.

Stop1 is a complete power down. The MCU exits with only IRQ or reset pins and the MCU is always reset after stop recovery. All RAM contents are lost, all register contents are reset, and I/O pins are in reset state.

Stop2 is a partial power down. The MCU exists with IRQ, reset, or internal RTI and the MCU is always reset after stop recovery. RAM contents are maintained with I/O states latched and the RTI can be used to wake up the MCU without an external event and register values are reset, but values can be saved to and restored from RAM.

Stop3 is a more flexible stop mode but with higher power consumption. The MCU exits with any active interrupt: IRQ, KBI, LVD, RTI, or reset. RAM and register retain their values which means that it does not require re-initialization of peripherals and the design can use an external clock as a highly accurate input into the RTI. The MC9RS08KA2 STOP and WAIT instructions do behave slightly differently to compensate for the lack of stacking process as the code resumes from the next instruction upon exit.

NOTE

Typical stop3 current consumption values for the MC9S08QG8, MC9S08QD4 and MC9RS08KA2 are 750 nA, 900 nA and 2.5 μ A at 3 V and 85°C, respectively.

1.4.9 Analog Comparator

The ACMP module functions the same in the MC9S08QG8 and the MC9RS08KA2. The registers are identical and the electrical characteristics are the same. The MC9S08QD4 does not have an on-chip ACMP but it is likely this function can be initiated by the ADC, which shares the same pins.

1.4.10 Analog to Digital Converter

The only difference between the ADC on the MC9S08QG8 and the MC9S08QD4 is the option for four extra channels on the 16-pin MC9S08QG8. The MC9RS08KA2 does not have this module, but because the ACMP shares the same pins an ADC can be implemented using this module and the MTIM. Freescale document, “RS08 Peripheral Module Quick Reference,” explains how to recreate this function.

NOTE

MC9S08QG8 and MC9S08QD4 devices have slightly different module names in the CodeWarrior header files. ADC is named ADC1 on the MC9S08QD4.

1.4.11 IIC, SCI and SPI

The MC9S08QG8 is a serial communication-rich microcontroller whereas the MC9S08QD4 and MC9RS08KA2 do not have these modules on chip. These serial communication features can be simulated to various degrees in software and Freescale has published numerous application notes showing how to emulate these types of modules. Please refer to www.freescale.com for the latest versions of these application notes and quick reference guides.

1.4.12 KBI

The keyboard interrupt modules on the three devices differ only in the number of channels.

1.4.13 MTIM and TPM

The MTIM module is available on the MC9S08QG8 and the MC9RS08KA2. This module is a simple 8-bit counter with four selectable clock sources and nine prescaler values.

The MC9S08QG8 and the MC9S08QD4 have an on-chip TPM. This module is a more sophisticated 16-bit timer than the MTIM.

The MC9S08QG8 has a single two-channel TPM. Two additional configuration options that are not available on the TPM of the MC9S08QD4 were added to this device. The ACMP module connects its output to the TPM input capture channel 0 by setting ACIC in SOPT2. With ACIC set, the TPMCH0 pin is not externally available regardless of the configuration of the TPM module. The MTIM and the TPM can be clocked simultaneously. When the external clock for the TPM module, TPMCLK, is selected by setting $CLKS[B:A] = 1:1$ in TPMSC the TCLK input on PTA5 can be enabled as external clock inputs to both the MTIM and TPM.

The MC9S08QD4 has two independent TPM modules: one 1 channel and one 2 channel.

Each TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on all channels. Clock sources are independently selectable per TPM on a multiple TPM device such as the MC9S08QD4. It has selectable clock sources and clock prescaler taps for division by 1, 2, 4, 8, 16, 32, 64, or 128. It can be configured as 16-bit free-running or up/down (CPWM) count operation or 16-bit modulus register to control counter range. Each channel has an individual interrupt and a terminal count interrupt for each TPM module on a multiple TPM device such as the MC9S08QD4.

Each channel may be input capture, output compare, or buffered edge-aligned PWM with rising-edge, falling-edge, or any-edge input capture trigger. Each channel has set, clear, or toggle output compare action and have selectable polarity on PWM outputs.

1.5 Pin-To-Pin Compatibility

These three devices are pin-to-pin compatible. You can replace one device for any of the other two without creating electrical problems in the connection on the new device so long as you perform the necessary I/O configuration. Table 4 shows the pin assignment and priority for the 8-pin package versions of the MC9S08QG8, MC9S08QD4, and the MC9RS08KA2.

Table 4. Pin Descriptions

Pin #	MC9S08QG8					MC9S08QD4				MC9RS08KA2				
	8 DFN, 8 NB SOIC, 8 PDIP					8 PDIP, 8 NB SOIC				8 PDIP, 8 NB SOIC				
Priority	Lowest				Highest	Lowest			Highest	Lowest				Highest
1	PTA5	IRQ	TCLK	RESET		PTA5	TPM2CH1	IRQ	RESET	PTA2	KBIP2	TCLK	RESET	VPP
2	PTA4	ACMPO	BKGD	MS		PTA4	TPM2CH0	BKGD	MS	PTA3	ACMPO	BKGD	MS	
3	VDD					VDD				VDD				
4	VSS					VSS				VSS				
5	PTA3	KBIP3	SCL	ADP3		PTA3	KBIP3	TCLK2	ADP3	PTA5	KBIP5			
6	PTA2	KBIP2	SDA	ADP2		PTA2	KBIP2	TCLK1	ADP2	PTA4	KBIP4			
7	PTA1	KBIP1	ADP1	ACMP-		PTA1	KBIP1	TPM1CH1	ADP1	PTA1	KBIP1	ACMP-		
8	PTA0	KBIP0	TPMCH0	ADP0	ACMP+	PTA0	KBIP0	TPM1CH0	ADP0	PTA0	KBIP0	ACMP+		

V_{DD}, Ground (V_{SS}), Reset, and Background connection for programming and debugging code are on the same pins. Peripherals that exist on all the devices, such as keyboard interrupt module, are located on the same pins and peripherals, such the analog comparator, can be multiplexed with analog-to-digital converter. Pins share more than one peripheral, allowing migration to a different device and using the new device's peripheral to improve functionality. For example, when controlling a motor with a general purpose output you can move to a different device with PWM capability and use the peripheral of the MCU without changing the hardware. The pins on all the devices also have the same current limit of 25 mA.

1.5.1 Pin 1

Pin 1 has the reset signal and a timer channel (either from MTIM or TPM) on all of the products. MC9S08QG8 and MC9S08QD4 also have the IRQ pin function located on this pin. The MC9RS08KA2 device has an additional KBI which can be used to emulate an interrupt request in software. The MC9RS08KA2 also has V_{PP} on this pin. This is the highest priority signal but it is used only during programming of the Flash memory. Because the voltage required to program is 12 V, it is unlikely to be built into the functionality of the application board and it is only used during development and programming during the manufacturing process.

1.5.2 Pin 2

Pin 2 has the BKGD/MS function common to all products. The MC9S08QG8 and MC9RS08KA2, which have the ACMP module in common, have the output of the module on this pin. The MC9S08QD4, which does not have a ACMP, has one of its second TPM channels available in this pin.

1.5.3 Pins 3 and Pin 4 – Power

All the 8-pin package options have V_{DD} and V_{SS} on pins 3 and 4, respectively. The three products have different ranges of operating voltage (V_{DD}), which can be supplied to the MCU, but the correct choice of MCU will mitigate any problems.

1.5.4 Pins 5 and 6

A KBI function and an ADC channel are bonded to these pins. The MC9S08QG8 has an SCI clock and data signal on these pins. The MC9S08QD4 has its MTIM channels available.

1.5.5 Pin 7

Pin 7 has a KBI function common to all parts. It is also host to an analog function, either ACMP, ADC, or both depending on the MCU. The MC9S08QD4 has one of the additional TPM channels bonded to this pin.

1.5.6 Pin 8

Pin 8 also has KBI as well as some analog functionality. A TPM channel is available on this pin when either MC9S08QG8 or MC9S08QD4 are selected.

1.5.7 Examples

Two scenarios lead to the migration from one device to another pin-compatible device. The first is when the option of a new peripheral includes another MCU that could simplify the development of the code. The second, opposite case, is when a feature can be performed in software to emulate a peripheral and therefore the project can use a cheaper device.

For the first case, an example of PWM generation based on analog voltage is used. This operation can be easily solved using the smaller device on the migration path, the MCRS08KA2. Even though the RS08 does not have a timer that can automatically generate PWM or an analog-to-digital converter, both functionalities can be emulated with the 8-bit modulo timer and the analog comparator of the RS08. However, generating more than one PWM of different characteristics can be complicated to solve in code because we are using the timer to generate the first PWM and the ADC emulation. Migrating to the next device on the path, the MC9S08QD4, can quickly solve the problem because the ADC is an included peripheral can use two or even the three timer channels in the device to generate the PWM. Moving up to the MC9S08QG8 can add capabilities to use the included I²C peripheral to communicate data or to store it into an external memory component.

The second could be found at the beginning of the development phase when the complexity of the design is bigger than it really is. During the design of the hardware, in the middle of the code development it is found that the problem can be solved easier than anticipated and the design can change to a smaller device. In this case, the design can move from a MC9S08QG8 to an MC9S08QD4 or to a MC9RS08KA2 without any change in the hardware. The pin-to-pin compatibility is of use because hardware testing of the device can continue. The missing peripherals can be emulated on the new part with the existing ones and at the end of the development, the functionality will be that required and the cost of the design will be lower than the initial calculation.

1.6 Migration Software Tools

It is important to have software tools that span a migration from one device to another. A single tool that works with all the devices is available. It is only necessary to familiarize yourself with a single tool and all the functionality associated with it.

A software tool that generates code to initialize the peripherals of each device is included. The Device Initialization Tool in CodeWarrior helps designers concentrate only on the routines that have the functionality of the design thereby leaving the initialization of the peripheral to the tool. Because this tool, CodeWarrior for HC(R/S)08 V5.1, exists for these three devices, the peripheral initialization can be done automatically and designers must care only of the routines.

Changing the target MCU in CodeWarrior versions 5.0 and above is simple. From the project menu, select Change MCU/Connection (see [Figure 6](#)).

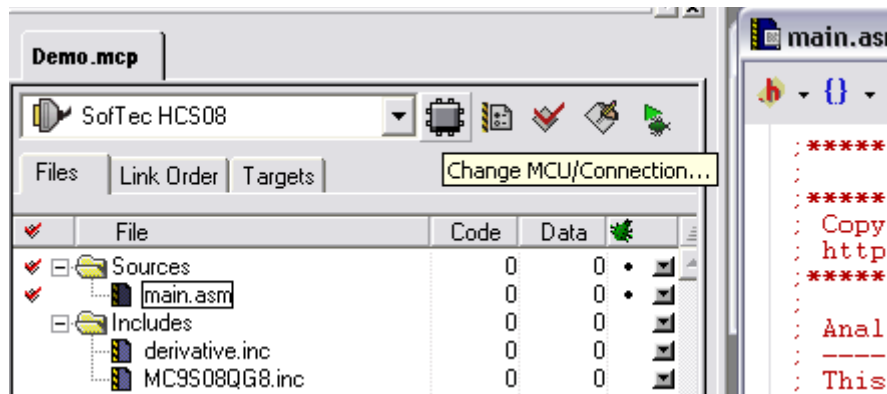


Figure 6. Change MCU/Connection

You can change the connection and the target microcontroller from a comprehensive list of parts. CodeWarrior will substitute the previous MCU files with the newly-selected one, which will update the memory map and make available all declarations of the new modules. By compiling the project again, any potential problems in the migration appear as errors or warnings, which can be individually viewed. Application code will easily convert if written with the CodeWarrior header files, for example, PTA5 is named PTAD_PTAD5 across all 8-bit MCUs.

HCS08 supports C code development in CodeWarrior but the C compiler for the RS08 is under development. Until the RS08 C compiler is available, code developers can use the CodeWarrior disassemble function to convert their C code into assembler (see [Figure 7](#)). In the source window

CodeWarrior can disassemble a C program and show the assembly equivalent code. Figure 8 is a snapshot of the CodeWarrior display.

```

/* MCU_init(): */

SOPT1 = 0x20; /* disable COP enable stop */
SOPT2_MCSEL = 1; /* enable MCLK. MCLK = BUSCLK/2 */

SPMSC1 = 0x10;
SPMSC2 = 0x01; /* Enable

Ports_Init();

EnableInterrupts; /* ena

/* include your code here */

/* Configure MCG to produce a 2.0MHz bus clock from a 4.0MHz externa
MCGC2_RANGE = 1;
MCGC2_ERCLKEN=1;
MCGC2_EREFS = 1;
MCGC2_EREFSSTEN =1;
-----

```

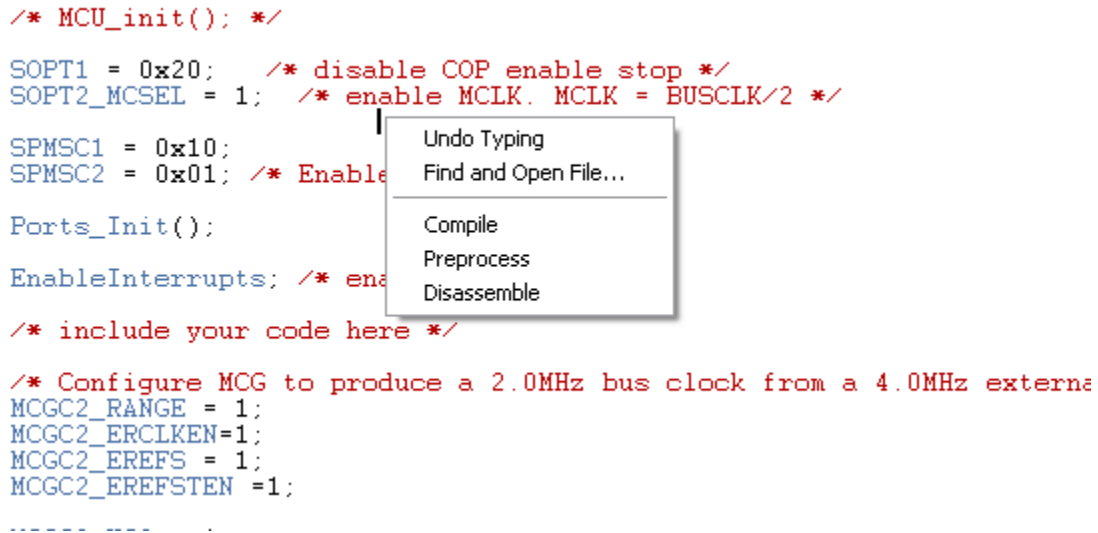


Figure 7. How to Disassemble

```

62:      SOPT1 = 0x20; /* disable COP enable stop */
63:      SOPT2_MCSEL = 1; /* enable MCLK. MCLK = BUSCLK/2 */
64:      SPMSC2_PPDACK = 1;
000f 450000 [3]      LDHX  @_SPMSC2
0012 f6 [3]      LDA  ,X
0013 aa04 [2]      ORA  #4
0015 f7 [2]      STA  ,X
0016 [5]      L16:
65:      while (1) PTDD_PTDD7 ^=1;
0016 b600 [3]      LDA  _PTDD
0018 a880 [2]      EOR  #-128
001a b700 [3]      STA  _PTDD
001c 20f8 [3]      BRA  L16 ;abs = 0016
001e [5]      L1E:
001e ad59 [5]      BSR  L79 ;abs = 0079
66:      }

```

Figure 8. Example of Disassembled Code

The last key consideration of low-end design is the debugging interface. A single-wire programming and debugging interface exists on the RS08 family and in the S08 devices. The same software will provide debugging interface for the design and designers can use the same tool to program all of them. A full design can be migrated to a different MCU without changing the hardware or the software and the same programming tools can be used for these devices. This is the BDC module.

2 Conclusion

Portability is in designers' minds when developing a project. Software portability and hardware compatibility are key factors in speeding design and getting the most out of an MCU. The use of devices that offer an effortless move from one device to another with different peripherals and features expands the use of the MCU in each design and give designers confidence to change the device if the design is required to do so.

The use of the same tools to develop projects is also very important; it would be very inefficient to switch the software or the programmer to finish the migration and that is why having software and hardware tools that are also compatible for the devices is a very important component to quickly change the MCU used. Having the same programmer and debugger also helps to efficiently use the tools and make easier the migration process.



THIS PAGE IS INTENTIONALLY BLANK



THIS PAGE IS INTENTIONALLY BLANK



THIS PAGE IS INTENTIONALLY BLANK

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:

www.freescale.com

E-mail:

support@freescale.com

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2006. All rights reserved.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Document Number: AN3325
Rev. 0
09/2006

