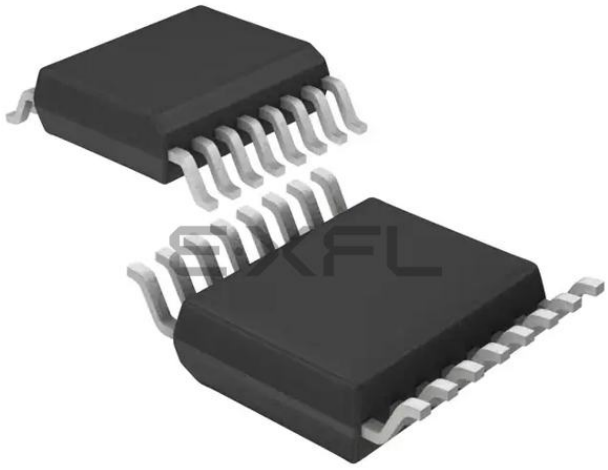

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	S08
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	12
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	16-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	16-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08qg84cdter

This product incorporates SuperFlash® Technology licensed from SST.

Chapter 6
Parallel Input/Output Control

Chapter 7
Central Processor Unit (S08CPUV2)

Chapter 8
Analog Comparator (S08ACMPV2)

After entering active background mode, the CPU is held in a suspended state waiting for serial background commands rather than executing instructions from the user application program.

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:
  - Memory access commands
  - Memory-access-with-status commands
  - BDC register access commands
  - The BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode. Active background commands include commands to:
  - Read or write CPU registers
  - Trace one user program instruction at a time
  - Leave active background mode to return to the user application program (GO)

The active background mode is used to program a bootloader or user application program into the FLASH program memory before the MCU is operated in run mode for the first time. When the MC9S08QG8/4 is shipped from the Freescale factory, the FLASH program memory is erased by default unless specifically noted, so there is no program that could be executed in run mode until the FLASH memory is initially programmed. The active background mode can also be used to erase and reprogram the FLASH memory after it has been previously programmed.

For additional information about the active background mode, refer to the Development Support chapter.

3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The I bit in the condition code register (CCR) is cleared when the CPU enters wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

While the MCU is in wait mode, there are some restrictions on which background debug commands can be used. Only the BACKGROUND command and memory-access-with-status commands are available while the MCU is in wait mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from wait mode and enter active background mode.

Chapter 4
Memory Map and Register Definition

4.1 MC9S08QG8/4 Memory Map

As shown in Figure 4-1, on-chip memory in the MC9S08QG8/4 series of MCUs consists of RAM, FLASH program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into these groups:

- Direct-page registers (0x0000 through 0x005F)
- High-page registers (0x1800 through 0x184F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

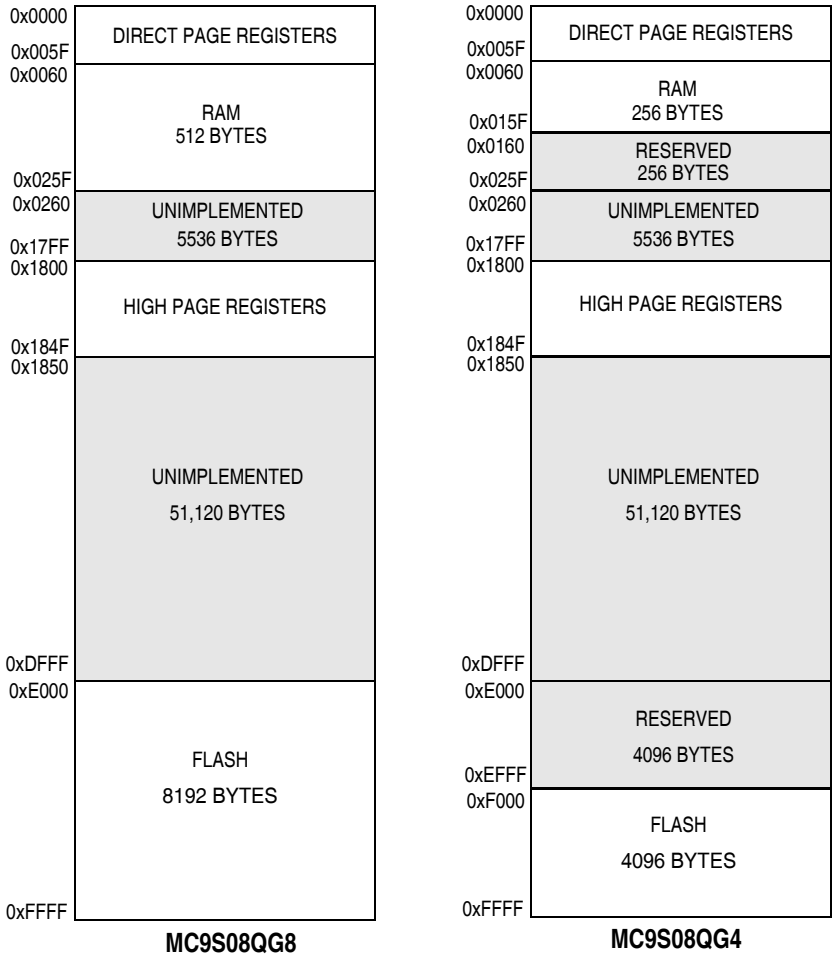

Figure 4-1. MC9S08QG8/4 Memory Map

Table 4-2. Direct-Page Register Summary (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x003F	MTIMMOD	MOD							
0x0040	TPMSC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0041	TPMCNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0042	TPMCNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0043	TPMMODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0044	TPMMODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0045	TPMC0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0046	TPMC0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0047	TPMC0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0048	TPMC1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0049	TPMC1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x004A	TPMC1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x004B– 0x005F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —

High-page registers, shown in Table 4-3, are accessed much less often than other I/O and control registers so they have been located outside the direct addressable memory space, starting at 0x1800.

Table 4-3. High-Page Register Summary

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x1801	SBDFR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPE	COPT	STOPE	—	0	0	BKGDPE	RSTPE
0x1803	SOPT2	COPCLKS	0	0	0	0	0	IICPS	ACIC
0x1804	Reserved	—	—	—	—	—	—	—	—
0x1805	Reserved	—	—	—	—	—	—	—	—
0x1806	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	SRTISC	RTIF	RTIACK	RTICLKS	RTIE	0	RTIS		
0x1809	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
0x180A	SPMSC2	0	0	0	PDF	PPDF	PPDACK	PDC	PPDC
0x180B	Reserved	—	—	—	—	—	—	—	—
0x180C	SPMSC3	LVWF	LVWACK	LVDV	LVWV	—	—	—	—
0x180D– 0x180F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0

Table 4-4. Nonvolatile Register Summary

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFAE	Reserved for Storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0xFFAF	Reserved for Storage of ICSTRM	TRIM							
0xFFB0 – 0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8 – 0xFFBC	Unused	— —	— —	— —	— —	— —	— —	— —	— —
0xFFBD	NVPROT	FPS							FPDIS
0xFFBE	Unused	—	—	—	—	—	—	—	—
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00

Provided the key enable (KEYEN) bit is 1, the 8-byte comparison key can be used to temporarily disengage memory security. This key mechanism can be accessed only through user code running in secure memory. (A security key cannot be entered directly through background debug commands.) This security key can be disabled completely by programming the KEYEN bit to 0. If the security key is disabled, the only way to disengage security is by mass erasing the FLASH if needed (normally through the background debug interface) and verifying that FLASH is blank. To avoid returning to secure mode after the next reset, program the security bits (SEC01:SEC00) to the unsecured state (1:0).

4.4 RAM

The MC9S08QG8/4 includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power wait, stop2, or stop3 mode. At power-on or after wakeup from stop1, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention (V_{RAM}).

For compatibility with M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08QG8/4, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale Semiconductor-provided equate file).

```
LDHX    #RamLast+1    ;point one past RAM
TXS                   ;SP<-(H:X-1)
```

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or through code executing from non-secure memory. See Section 4.6, “Security,” for a detailed description of the security feature.

The RAM array is not automatically initialized out of reset.

4.5.3 Program and Erase Command Execution

The steps for executing any of the commands are listed below. The FCDIV register must be initialized and any error flags cleared before beginning command execution. The command execution steps are:

1. Write a data value to an address in the FLASH array. The address and data information from this write is latched into the FLASH interface. This write is a required first step in any command sequence. For erase and blank check commands, the value of the data is not important. For page erase commands, the address may be any address in the 512-byte page of FLASH to be erased. For mass erase and blank check commands, the address can be any address in the FLASH memory. Whole pages of 512 bytes are the smallest block of FLASH that may be erased.

NOTE

Do not program any byte in the FLASH more than once after a successful erase operation. Reprogramming bits to a byte that is already programmed is not allowed without first erasing the page in which the byte resides or mass erasing the entire FLASH memory. Programming without first erasing may disturb data stored in the FLASH.

2. Write the command code for the desired command to FCMD. The five valid commands are blank check (0x05), byte program (0x20), burst program (0x25), page erase (0x40), and mass erase (0x41). The command code is latched into the command buffer.
3. Write a 1 to the FCBEF bit in FSTAT to clear FCBEF and launch the command (including its address and data information).

A partial command sequence can be aborted manually by writing a 0 to FCBEF any time after the write to the memory array and before writing the 1 that clears FCBEF and launches the complete command. Aborting a command in this way sets the FACCERR access error flag, which must be cleared before starting a new command.

A strictly monitored procedure must be obeyed or the command will not be accepted. This minimizes the possibility of any unintended changes to the FLASH memory contents. The command complete flag (FCCF) indicates when a command is complete. The command sequence must be completed by clearing FCBEF to launch the command. Figure 4-2 is a flowchart for executing all of the commands except for burst programming. The FCDIV register must be initialized before using any FLASH commands. This must be done only once following a reset.

must be programmed to logic 0 to enable block protection. Therefore the value 0xF8 must be programmed into NVPROT to protect addresses 0xFA00 through 0xFFFF.

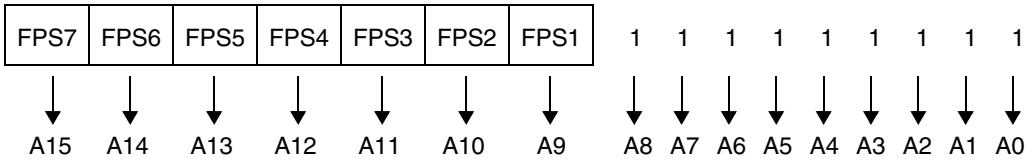

Figure 4-4. Block Protection Mechanism

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

4.5.7 Vector Redirection

Whenever any block protection is enabled, the reset and interrupt vectors will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register located at address 0xFFBF to zero. For redirection to occur, at least some portion but not all of the FLASH memory must be block protected by programming the NVPROT register located at address 0xFFBD. All of the interrupt vectors (memory locations 0xFFC0–0xFFFD) are redirected, though the reset vector (0xFFFE:FFFF) is not.

For example, if 512 bytes of FLASH are protected, the protected address region is from 0xFE00 through 0xFFFF. The interrupt vectors (0xFFC0–0xFFFD) are redirected to the locations 0xFDC0–0xFDFD. Now, if an SPI interrupt is taken for instance, the values in the locations 0xFDE0:FDE1 are used for the vector instead of the values in the locations 0xFFE0:FFE1. This allows the user to reprogram the unprotected portion of the FLASH with new program code including new interrupt vector values while leaving the protected area, which includes the default vector locations, unchanged.

4.6 Security

The MC9S08QG8/4 includes circuitry to prevent unauthorized access to the contents of FLASH and RAM memory. When security is engaged, FLASH and RAM are considered secure resources. 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 MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two nonvolatile register bits (SEC01:SEC00) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from FLASH into the working FOPT register in high-page register space. A user engages security by programming the NVOPT location which can be done at the same time the FLASH memory is programmed. The 1:0 state disengages security and the other three combinations engage security. Notice the erased state (1:1) makes

Table 5-2. Vector Summary

Vector Priority	Vector Number	Address (High:Low)	Vector Name	Module	Source	Enable	Description
Lower	31 through 24	0xFFC0:FFC1 through 0xFFCE:FFCF	Unused Vector Space (available for user program)				
	23	0xFFD0:FFD1	Vrti	System control	RTIF	RTIE	Real-time interrupt
	22	0xFFD2:FFD3	—	—	—	—	—
	21	0xFFD4:FFD5	—	—	—	—	—
	20	0xFFD6:FFD7	Vacmp	ACMP	ACF	ACIE	ACMP
	19	0xFFD8:FFD9	Vadc	ADC	COCO	AIEN	ADC
	18	0xFFDA:FFDB	Vkeyboard	KBI	KBF	KBIE	Keyboard pins
	17	0xFFDC:FFDD	Viic	IIC	IICIF	IICIE	IIC control
	16	0xFFDE:FFDF	Vscitx	SCI	TDRE TC	TIE TCIE	SCI transmit
	15	0xFFE0:FFE1	Vscirx	SCI	IDLE RDRF	ILIE RIE	SCI receive
	14	0xFFE2:FFE3	Vscierr	SCI	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI error
	13	0xFFE4:FFE5	Vspi	SPI	SPIF MODF SPTEF	SPIE SPIE SPTIE	SPI
	12	0xFFE6:FFE7	Vmtim	MTIM	TOF	TOIE	MTIM
	11	0xFFE8:FFE9	—	—	—	—	—
	10	0xFFEA:FFEB	—	—	—	—	—
	9	0xFFEC:FFED	—	—	—	—	—
	8	0xFFEE:FFEF	—	—	—	—	—
	7	0xFFF0:FFF1	Vtpmovf	TPM	TOF	TOIE	TPM overflow
	6	0xFFF2:FFF3	Vtpmch1	TPM	CH1F	CH1IE	TPM channel 1
	5	0xFFF4:FFF5	Vtpmch0	TPM	CH0F	CH0IE	TPM channel 0
	4	0xFFF6:FFF7	—	—	—	—	—
	3	0xFFF8:FFF9	Vlvd	System control	LVDF	LVDIE	Low-voltage detect
	2	0xFFFA:FFFB	Virq	IRQ	IRQF	IRQIE	$\overline{\text{IRQ}}$ pin
	1	0xFFFC:FFFD	Vswi	CPU	SWI Instruction	—	Software interrupt
Higher	0	0xFFFE:FFFF	Vreset	System control	COP LVD $\overline{\text{RESET}}$ pin Illegal opcode Illegal address POR	COPE LVDRE RSTPE — — —	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address power-on-reset

5.8.3 System Background Debug Force Reset Register (SBDFR)

This high page register contains a single write-only control bit. A serial background command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								BDFR[1]
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-4. System Background Debug Force Reset Register (SBDFR)

[1] BDFR is writable only through serial background debug commands, not from user programs.

Table 5-5. SBDFR Register Field Descriptions

Field	Description
0 BDFR	**Background Debug Force Reset** — A serial background command such as WRITE_BYTE can be used to allow an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program. To enter user mode, PTA4/ACMPO/BKGD/MS must be high immediately after issuing WRITE_BYTE command. To enter BDM, PTA4/ACMPO/BKGD/MS must be low immediately after issuing WRITE_BYTE command. See Table A-9., “Control Timing,” for more information.

6.4.2.3 Port A Drive Strength Select (PTADS)

An output pin can be selected to have high output drive strength by setting the corresponding bit in the drive strength select register (PTADS). When high drive is selected, a pin is capable of sourcing and sinking greater current. Even though every I/O pin can be selected as high drive, the user must ensure that the total current source and sink limits for the chip are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive a greater load with the same switching speed as a low drive enabled pin into a smaller load. Because of this the EMC emissions may be affected by enabling pins as high drive.

6.4.2.4 Port A Drive Strength Select (PTADS)

	7	6	5	4	3	2	1	0
R	0	0	PTADS5[1]	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-8. Drive Strength Selection for Port A Register (PTADS)

1 PTADS5 has no effect on the input-only PTA5 pin.

Table 6-5. PTADS Register Field Descriptions

Field	Description
5:0 PTADS[5:0]	**Output Drive Strength Selection for Port A Bits** — Each of these control bits selects between low and high output drive for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port A bit n. 1 High output drive strength selected for port A bit n.

10.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL to be disabled and thus conserve power when it is not being used. However, in some applications it may be desirable to enable the FLL and allow it to lock for maximum accuracy before switching to an FLL engaged mode. Do this by writing the LP bit to 0.

10.4.5 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal will be presented as ICSIRCLK, which can be used as an additional clock source. The ICSIRCLK frequency can be re-targeted by trimming the period of the internal reference clock. This can be done by writing a new value to the TRIM bits in the ICSTRM register. Writing a larger value will slow down the ICSIRCLK frequency, and writing a smaller value to the ICSTRM register will speed up the ICSIRCLK frequency. The TRIM bits will effect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low power (FBILP) mode. The TRIM and FTRIM value will not be affected by a reset.

Until ICSIRCLK is trimmed, programming low reference divider (RDIV) factors may result in ICSOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the Device Overview chapter).

If IREFSTEN is set and the IRCLKEN bit is written to 1, the internal reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value can be copied to the ICSTRM register during reset initialization. The factory trim value does not include the FTRIM bit. For finer precision, the user can trim the internal oscillator in the application and set the FTRIM bit accordingly.

10.4.6 Optional External Reference Clock

The ICS module can support an external reference clock with frequencies between 31.25 kHz to 5 MHz in all modes. When the ERCLKEN is set, the external reference clock signal will be presented as ICSERCLK, which can be used as an additional clock source. When IREFS = 1, the external reference clock will not be used by the FLL and will only be used as ICSERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support (see the Device Overview chapter).

If EREFSTEN is set and the ERCLKEN bit is written to 1, the external reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

10.4.7 Fixed Frequency Clock

The ICS provides the divided FLL reference clock as ICSFFCLK for use as an additional clock source for peripheral modules. The ICS provides an output signal (ICSFFE) which indicates when the ICS is providing ICSOUT frequencies four times or greater than the divided FLL reference clock (ICSFFCLK). In FLL engaged mode (FEI and FEE), this is always true and ICSFFE is always high. In ICS Bypass modes, ICSFFE will get asserted for the following combinations of BDIV and RDIV values:

Figure 13-3. MTIM **Register Summary**

Name		7	6	5	4	3	2	1	0
MTIMSC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIMCLK	R	0	0	CLKS		PS			
	W								
MTIMCNT	R	COUNT							
	W								
MTIMMOD	R	MOD							
	W								

Each MTIM includes four registers:

- An 8-bit status and control register
- An 8-bit clock configuration register
- An 8-bit counter register
- An 8-bit modulo register

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all MTIM registers.This section refers to registers and control bits only by their names and relative address offsets.

Some MCUs may have more than one MTIM, so register names include placeholder characters to identify which MTIM is being referenced.

13.4.1 MTIM Operation Example

This section shows an example of the MTIM operation as the counter reaches a matching value from the modulo register.

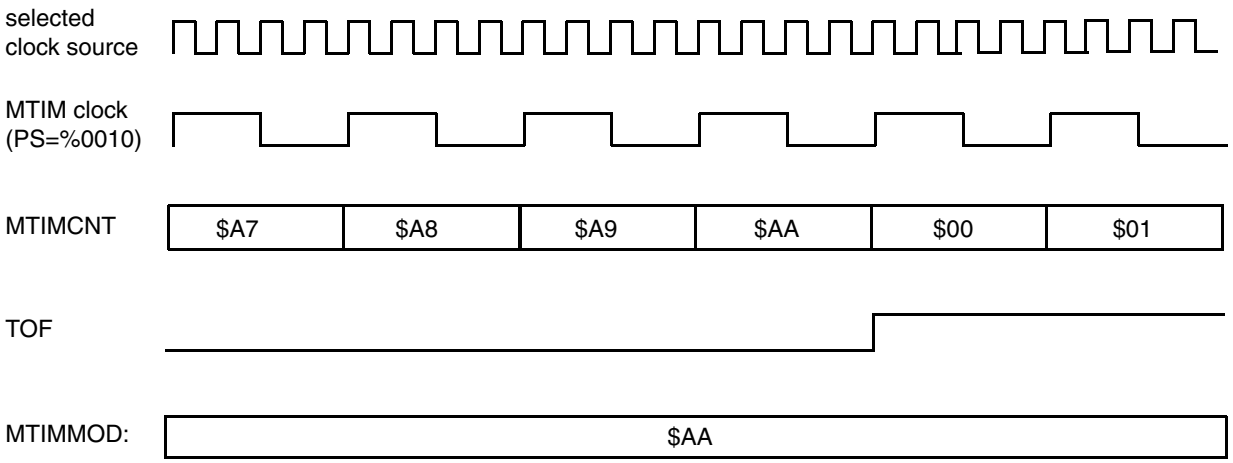

Figure 13-8. MTIM counter overflow example

In the example of Figure 13-8, the selected clock source could be any of the five possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMMOD register is set to $AA. When the counter, MTIMCNT, reaches the modulo value of $AA, the counter overflows to $00 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from $AA to $00. An MTIM overflow interrupt is generated when TOF is set, if TOIE = 1.

Chapter 16
Timer/Pulse-Width Modulator (S08TPMV2)

16.1 Introduction

Figure 16-1 shows the MC9S08QG8/4 block diagram with the TPM highlighted.

16.1.1 ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to TPM input capture channel 0 by setting ACIC in SOPT2. With ACIC set, the TPMCH0 pin is not available externally regardless of the configuration of the TPM module.

16.1.2 MTIM/TPM Configuration Information

The external clock for the TPM module, TPMCLK, is selected by setting CLKS[B:A] = 1:1 in TPMSC, which selects the TCLK pin input. The TCLK input on PTA5 can be enabled as external clock inputs to both the MTIM and TPM modules simultaneously.

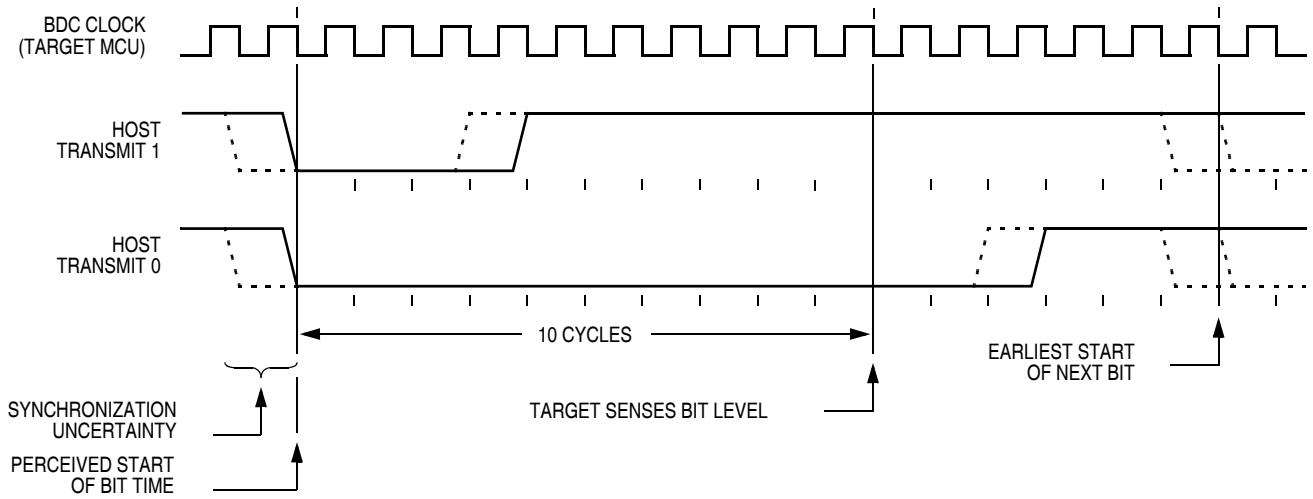

Figure 17-2. BDC Host-to-Target Serial Bit Timing

Figure 17-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

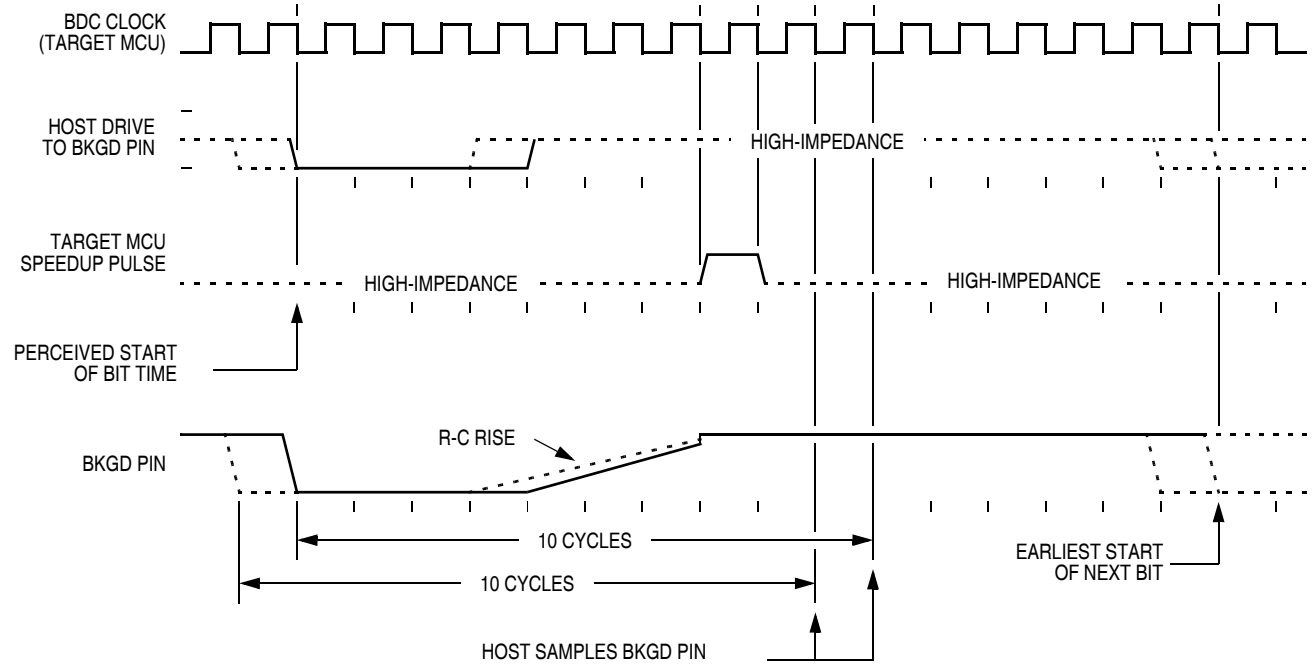

Figure 17-3. BDC Target-to-Host Serial Bit Timing (Logic 1)

the host must perform ((8 – CNT) – 1) dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see Section 17.3.5, “Trigger Modes”), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When ARM = 0, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

17.3.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

17.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGT register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

17.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGT register selects one of nine trigger modes. When TRGSEL = 1 in the DBGT register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGT chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGS. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGC.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

A-Only — Trigger when the address matches the value in comparator A

A OR B — Trigger when the address matches either the value in comparator A or the value in comparator B

A.8 AC Characteristics

This section describes timing characteristics for each peripheral system.

A.8.1 Control Timing

Table A-9. Control Timing

Parameter	Symbol	Min	Typ[1]	Max	Unit
Bus frequency (t_{cyc} = 1/f_{Bus})	f_{Bus}	0	—	10	MHz
Real-time interrupt internal oscillator period	t_{RTI}	700	1000	1300	µs
External reset pulse width[2]	t_{extrst}	100	—	—	ns
$\overline{IRQ}$ pulse width Asynchronous path[2] Synchronous path[3]	t_{ILIH}	 100 1.5 t_{cyc}	—	—	ns
KBIPx pulse width Asynchronous path[2] Synchronous path[3]	t_{ILIH}, t_{IHIL}	 100 1.5 t_{cyc}	—	—	ns
Port rise and fall time (load = 50 pF)[4] Slew rate control disabled (PTxSE = 0) Slew rate control enabled (PTxSE = 1)	t_{Rise}, t_{Fall}	 — —	 3 30	 — —	ns
BKGD/MS setup time after issuing background debug force reset to enter user or BDM modes	t_{MSSU}	500	—	—	ns
BKGD/MS hold time after issuing background debug force reset to enter user or BDM modes [5]	t_{MSH}	100	—	—	µs

1 Data in Typical column was characterized at 3.0 V, 25°C.

2 This is the shortest pulse that is guaranteed to be recognized.

3 This is the minimum pulse width that is guaranteed to pass through the pin synchronization circuitry. Shorter pulses may or may not be recognized. In stop mode, the synchronizer is bypassed so shorter pulses can be recognized in that case.

4 Timing is shown with respect to 20% V_{DD} and 80% V_{DD} levels. Temperature range –40°C to 85°C.

5 To enter BDM mode following a POR, BKGD/MS should be held low during the power-up and for a hold time of t_{MSH} after V_{DD} rises above V_{LVD}.