



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	40MHz
Connectivity	I ² C, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	17
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08sh8mwj



MC9S08SH8 Datasheet Addendum

This addendum describes corrections or updates to the *MC9S08SH8 Datasheet*, file named as MC9S08SH8. Please check our website at <http://www.freescale.com/>, for the latest updates.

The current version available of the *MC9S08SH8 Datasheet* is Revision 3.0.

Table of Contents

1	Addendum for Revision 3.0.	2
2	Revision History	2

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

<http://www.freescale.com/support>

USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
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. 2012. All rights reserved.

MC9S08SH8AD
Rev. 1
05/2012

Revision History

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

<http://freescale.com/>

The following revision history table summarizes changes contained in this document.

Revision Number	Revision Date	Description of Changes
0.01	3/08/2006	Initial review copy
1	11/2007	Updated Electricals and incorporated revisions from Project sync issues: 2394, 2600, 2601, and 2764.
2	3/2008	Corrected SPI module to be version 3. Incorporated fixes for Project Sync issues: 2394, 2600, 2601, 2764, 3237, and 3279; as well as, ADC Temperature Sensor issues 3331 and 3335. Adjusted Features page leading and fixed minor grammatical errors. Added 20-SOIC package option for the C temp only. Corrected package drawing number for 24-QFN.
3	6/2008	Added ICS over Temperature graph to Electricals. Resolved final TBDs.

© Freescale Semiconductor, Inc., 2007-2008. All rights reserved.
This product incorporates SuperFlash® Technology licensed from SST.

1.3 System Clock Distribution

Figure 1-2 shows a simplified clock connection diagram. Some modules in the MCU have selectable clock inputs as shown. The clock inputs to the modules indicate the clock(s) that are used to drive the module function.

The following defines the clocks used in this MCU:

- **BUSCLK** — The frequency of the bus is always half of ICSOUT.
- **ICSOUT** — Primary output of the ICS and is twice the bus frequency.
- **ICSLCLK** — Development tools can select this clock source to speed up BDC communications in systems where the bus clock is configured to run at a very slow frequency.
- **ICSERCLK** — External reference clock can be selected as the RTC clock source and as the alternate clock for the ADC module.
- **ICSIRCLK** — Internal reference clock can be selected as the RTC clock source.
- **ICSFFCLK** — Fixed frequency clock can be selected as clock source for the TPM1, TPM2 and MTIM modules.
- **LPOCLK** — Independent 1-kHz clock source that can be selected as the clock source for the COP and RTC modules.
- **TCLK** — External input clock source for TPM1, TPM2 and MTIM and is referenced as TPMCLK in TPM chapters.

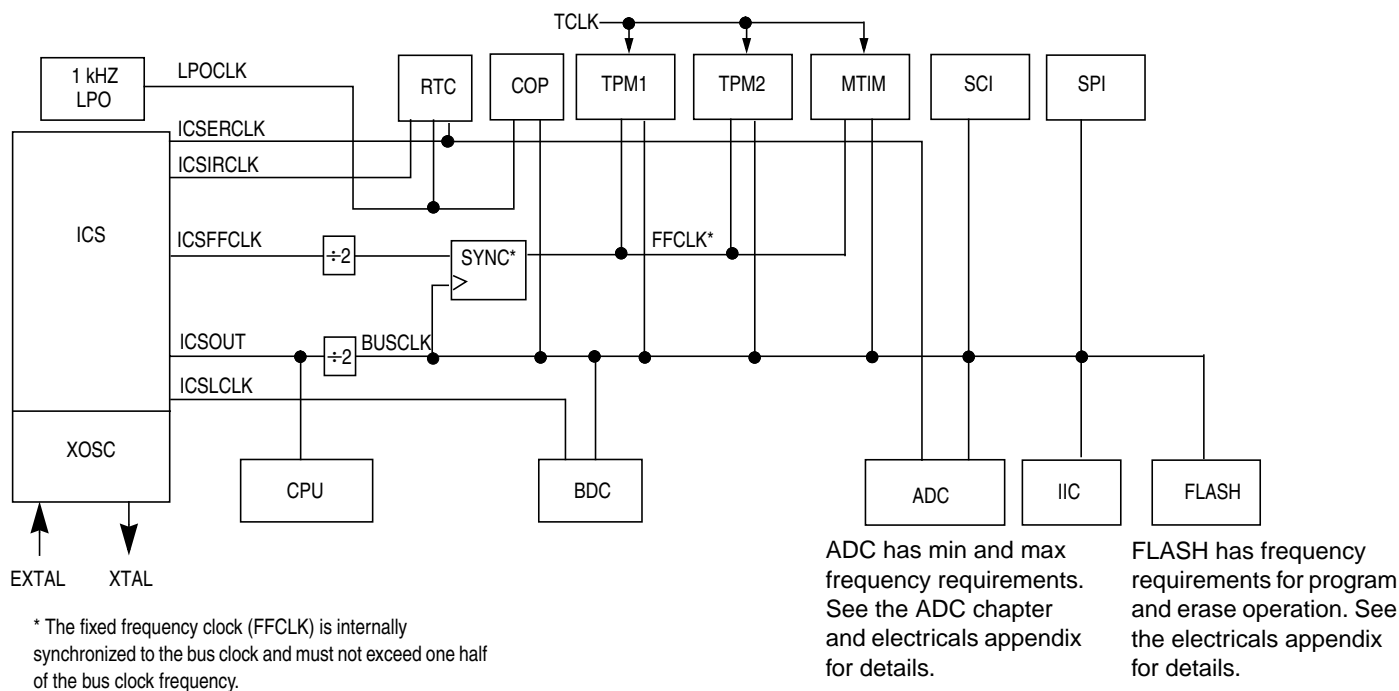


Figure 1-2. System Clock Distribution Diagram

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/MS 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 MC9S08SH8 is shipped from the Freescale Semiconductor 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 CCR is cleared when the CPU enters the wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits the 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 when 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.

3.6 Stop Modes

One of two stop modes is entered upon execution of a STOP instruction when STOPE in SOPT1. In any stop mode, the bus and CPU clocks are halted. The ICS module can be configured to leave the reference clocks running. See [Chapter 10, “Internal Clock Source \(S08ICSV2\),”](#) for more information.

Table 3-1 shows all of the control bits that affect stop mode selection and the mode selected under various conditions. The selected mode is entered following the execution of a STOP instruction.

Table 3-1. Stop Mode Selection

STOPE	ENBDM ¹	LVDE	LVDSE	PPDC	Stop Mode
0	x	x	x	x	Stop modes disabled; illegal opcode reset if STOP instruction executed
1	1	x	x	x	Stop3 with BDM enabled ²
1	0	Both bits must be 1	0	0	Stop3 with voltage regulator active
1	0	Either bit a 0	0	0	Stop3
1	0	Either bit a 0	1	1	Stop2

¹ ENBDM is located in the BDCSCR, which is only accessible through BDC commands, see [Section 17.4.1.1, “BDC Status and Control Register \(BDCSCR\)”](#).

² When in Stop3 mode with BDM enabled, The S_{IDD} will be near R_{IDD} levels because internal clocks are enabled.

3.6.1 Stop3 Mode

Stop3 mode is entered by executing a STOP instruction under the conditions as shown in [Table 3-1](#). The states of all of the internal registers and logic, RAM contents, and I/O pin states are maintained.

Stop3 can be exited by asserting $\overline{\text{RESET}}$ if enabled, or by an interrupt from one of the following sources: the real-time counter (RTC), LVD system, ACMP, ADC, SCI, IRQ, or any pin interrupts.

If stop3 is exited by means of the $\overline{\text{RESET}}$ pin, then the MCU is reset and operation will resume after taking the reset vector. Exit by means of one of the internal interrupt sources results in the MCU taking the appropriate interrupt vector.

3.6.1.1 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. For configuring the LVD system for interrupt or reset, refer to [5.6, “Low-Voltage Detect \(LVD\) System”](#). If the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 both set) at the time the CPU executes a STOP instruction, then the voltage regulator remains active during stop mode.

For the ADC to operate in stop mode, the LVD must be enabled when entering stop3.

For the ACMP to operate in stop mode with compare to internal bandgap option, the LVD must be enabled when entering stop3.

3.6.1.2 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in [Chapter 17, “Development Support.”](#) If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation.

Chapter 4

Memory

4.1 MC9S08SH8 Memory Map

As shown in [Figure 4-1](#), on-chip memory in the MC9S08SH8 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 three groups:

- Direct-page registers (0x0000 through 0x007F)
- High-page registers (0x1800 through 0x185F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

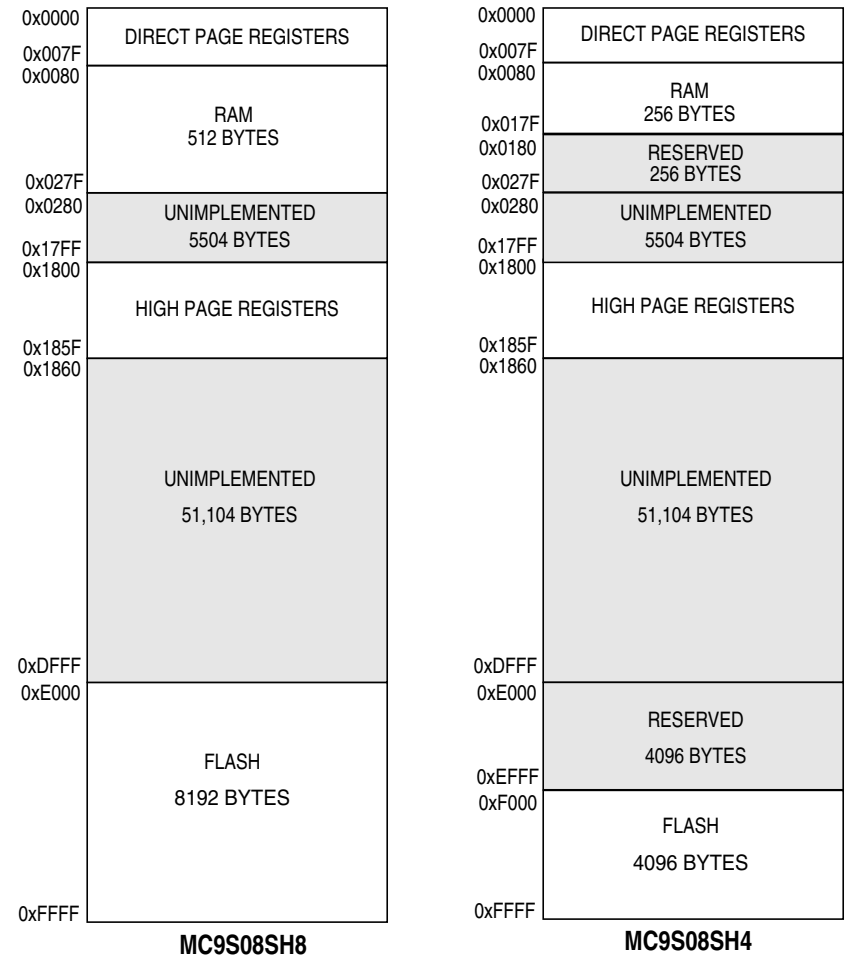


Figure 4-1. MC9S08SH8 Memory Map

Table 4-2. Direct-Page Register Summary (Sheet 1 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	PTAD	0	0	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	0	0	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x0002	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0004	PTCD	0	0	0	0	PTCD3	PTCD2	PTCD1	PTCD0
0x0005	PTCDD	0	0	0	0	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x0006– 0x000D	Reserved	—	—	—	—	—	—	—	—
0x000E	ACMPSC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0
0x000F	Reserved	—	—	—	—	—	—	—	—
0x0010	ADSC1	COCO	AIEN	ADCO	ADCH				
0x0011	ADSC2	ADACT	ADTRG	ACFE	ACFGT	—	—	—	—
0x0012	ADRH	0	0	0	0	0	0	ADR9	ADR8
0x0013	ADRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
0x0014	ADCVH	0	0	0	0	0	0	ADCV9	ADCV8
0x0015	ADCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x0016	ADCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x0018	APCTL2	0	0	0	0	ADPC11	ADPC10	ADPC9	ADPC8
0x0019	Reserved	—	—	—	—	—	—	—	—
0x001A	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x001B	Reserved	—	—	—	—	—	—	—	—
0x001C	MTIMSC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x001D	MTIMCLK	0	0	CLKS		PS			
0x001E	MTIMCNT	CNT							
0x001F	MTIMMOD	MOD							
0x0020	TPM1SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0021	TPM1CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0022	TPM1CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0023	TPM1MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0024	TPM1MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0025	TPM1C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0026	TPM1C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0027	TPM1C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	TPM1C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0029	TPM1C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x002A	TPM1C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x002B– 0x0037	Reserved	—	—	—	—	—	—	—	—

5.7.2 System Reset Status Register (SRS)

This high page register includes read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by writing 1 to BDFR in the SBDIFR register, none of the status bits in SRS will be set. Writing any value to this register address causes a COP reset when the COP is enabled except the values 0x55 and 0xAA. Writing a 0x55-0xAA sequence to this address clears the COP watchdog timer without affecting the contents of this register. The reset state of these bits depends on what caused the MCU to reset.

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	0	LVD	0
W	Writing 0x55, 0xAA 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.

Figure 5-3. System Reset Status (SRS)

Table 5-4. SRS Register Field Descriptions

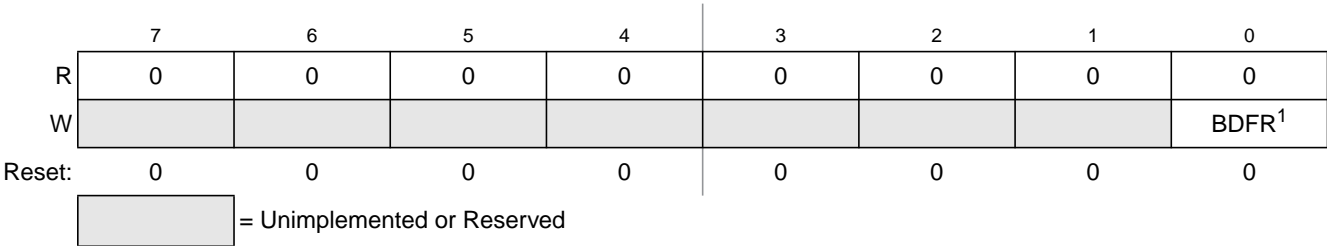
Field	Description
7 POR	Power-On Reset — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	External Reset Pin — Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
5 COP	Computer Operating Properly (COP) Watchdog — Reset was caused by the COP watchdog timer timing out. This reset source can be blocked by COPT bits = 0:0.. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.
4 ILOP	Illegal Opcode — Reset was caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by STOPE = 0 in the SOPT register. The BGND instruction is considered illegal if active background mode is disabled by ENBDM = 0 in the BDCSC register. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.

Table 5-4. SRS Register Field Descriptions

Field	Description
3 ILAD	Illegal Address — Reset was caused by an attempt to access either data or an instruction at an unimplemented memory address. 0 Reset not caused by an illegal address 1 Reset caused by an illegal address
1 LVD	Low Voltage Detect — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset will occur. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.

5.7.3 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.



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

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

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.

5.7.4 System Options Register 1 (SOPT1)

This high page register is a write-once register so only the first write after reset is honored. It can be read at any time. Any subsequent attempt to write to SOPT1 (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT1 should be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

	7	6	5	4 ¹	3	2	1	0
R	COPT		STOPE		0	IICPS	BKGDPE	RSTPE
W								
Reset:	1	1	0	0	0	0	1	u ²
POR:	1	1	0	0	0	0	1	0
LVR:	1	1	0	0	0	0	1	u

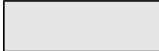
 = Unimplemented or Reserved

Figure 5-5. System Options Register 1 (SOPT1)

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

² u = unaffected

Table 5-6. SOPT1 Register Field Descriptions

Field	Description
7:6 COPT[1:0]	COP Watchdog Timeout — These write-once bits select the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period. See Table 5-1 .
5 STOPE	Stop Mode Enable — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.
2 IICPS	IIC Pin Select — This bit selects the location of the SDA and SCL pins of the IIC module. 0 SDA on PTA2, SCL on PTA3. 1 SDA on PTB6, SCL on PTB7.
1 BKGDPE	Background Debug Mode Pin Enable — This write-once bit when set enables the PTA4/ACMPO/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output-only alternative functions. This pin defaults to the BKGD/MS function following any MCU reset. 0 PTA4/ACMPO/BKGD/MS pin functions as PTA4 or ACMPO. 1 PTA4/ACMPO/BKGD/MS pin functions as BKGD/MS.
0 RSTPE	RESET Pin Enable — This write-once bit when set enables the PTA5/IRQ/TCLK/RESET pin to function as RESET. When clear, the pin functions as one of its alternative functions. This pin defaults to a general-purpose input port function following a POR reset. When configured as RESET, the pin will be unaffected by LVR or other internal resets. When RSTPE is set, an internal pullup device is enabled on RESET. 0 PTA5/IRQ/TCLK/RESET pin functions as PTA5, IRQ or TCLK. 1 PTA5/IRQ/TCLK/RESET pin functions as RESET.

7.3.6.7 SP-Relative, 16-Bit Offset (SP2)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.4 Special Operations

The CPU performs a few special operations that are similar to instructions but do not have opcodes like other CPU instructions. In addition, a few instructions such as STOP and WAIT directly affect other MCU circuitry. This section provides additional information about these operations.

7.4.1 Reset Sequence

Reset can be caused by a power-on-reset (POR) event, internal conditions such as the COP (computer operating properly) watchdog, or by assertion of an external active-low reset pin. When a reset event occurs, the CPU immediately stops whatever it is doing (the MCU does not wait for an instruction boundary before responding to a reset event). For a more detailed discussion about how the MCU recognizes resets and determines the source, refer to the [Resets, Interrupts, and System Configuration](#) chapter.

The reset event is considered concluded when the sequence to determine whether the reset came from an internal source is done and when the reset pin is no longer asserted. At the conclusion of a reset event, the CPU performs a 6-cycle sequence to fetch the reset vector from 0xFFFFE and 0xFFFF and to fill the instruction queue in preparation for execution of the first program instruction.

7.4.2 Interrupt Sequence

When an interrupt is requested, the CPU completes the current instruction before responding to the interrupt. At this point, the program counter is pointing at the start of the next instruction, which is where the CPU should return after servicing the interrupt. The CPU responds to an interrupt by performing the same sequence of operations as for a software interrupt (SWI) instruction, except the address used for the vector fetch is determined by the highest priority interrupt that is pending when the interrupt sequence started.

The CPU sequence for an interrupt is:

1. Store the contents of PCL, PCH, X, A, and CCR on the stack, in that order.
2. Set the I bit in the CCR.
3. Fetch the high-order half of the interrupt vector.
4. Fetch the low-order half of the interrupt vector.
5. Delay for one free bus cycle.
6. Fetch three bytes of program information starting at the address indicated by the interrupt vector to fill the instruction queue in preparation for execution of the first instruction in the interrupt service routine.

After the CCR contents are pushed onto the stack, the I bit in the CCR is set to prevent other interrupts while in the interrupt service routine. Although it is possible to clear the I bit with an instruction in the



Table 11-9. IICC2 Field Descriptions

Field	Description
7 GCAEN	General Call Address Enable. The GCAEN bit enables or disables general call address. 0 General call address is disabled 1 General call address is enabled
6 ADEXT	Address Extension. The ADEXT bit controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
2–0 AD[10:8]	Slave Address. The AD[10:8] field contains the upper three bits of the slave address in the 10-bit address scheme. This field is only valid when the ADEXT bit is set.

11.4 Functional Description

This section provides a complete functional description of the IIC module.

11.4.1 IIC Protocol

The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts:

- Start signal
- Slave address transmission
- Data transfer
- Stop signal

The stop signal should not be confused with the CPU stop instruction. The IIC bus system communication is described briefly in the following sections and illustrated in [Figure 11-9](#).

11.4.2 10-bit Address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

11.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 11-10). When a 10-bit address follows a start condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/\overline{W} direction bit) is 0. More than one device can find a match and generate an acknowledge (A1). Then, each slave that finds a match compares the eight bits of the second byte of the slave address with its own address. Only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Data	A	...	Data	A/A	P
---	--	----------	----	-----------------------------------	----	------	---	-----	------	-----	---

Table 11-10. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

11.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/\overline{W} bit (see Table 11-11). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated start condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the start condition (S) and tests whether the eighth (R/\overline{W}) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

After a repeated start condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/\overline{W}) bit. However, none of them are addressed because $R/\overline{W} = 1$ (for 10-bit devices) or the 11110XX slave address (for 7-bit devices) does not match.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Sr	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 1	A3	Data	A	...	Data	A	P
---	--	----------	----	-----------------------------------	----	----	--	----------	----	------	---	-----	------	---	---

Table 11-11. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

12.3.2 MTIM Clock Configuration Register (MTIMCLK)

MTIMCLK contains the clock select bits (CLKS) and the prescaler select bits (PS).

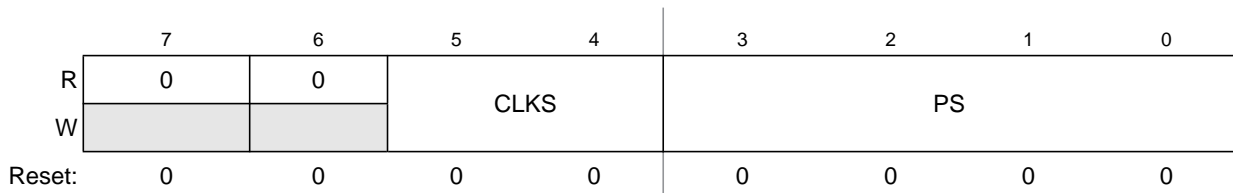


Figure 12-5. MTIM Clock Configuration Register

Table 12-3. MTIM Clock Configuration Register Field Descriptio

Field	Description
7:6	Unused register bits, always read 0.
5:4 CLKS	Clock Source Select — These two read/write bits select one of four different clock sources as the input to the MTIM prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 000. 00 Encoding 0. Bus clock (BUSCLK) 01 Encoding 1. Fixed-frequency clock (XCLK) 10 Encoding 3. External source (TCLK pin), falling edge 11 Encoding 4. External source (TCLK pin), rising edge All other encodings default to the bus clock (BUSCLK).
3:0 PS	Clock Source Prescaler — These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000. 0000 Encoding 0. MTIM clock source ÷ 1 0001 Encoding 1. MTIM clock source ÷ 2 0010 Encoding 2. MTIM clock source ÷ 4 0011 Encoding 3. MTIM clock source ÷ 8 0100 Encoding 4. MTIM clock source ÷ 16 0101 Encoding 5. MTIM clock source ÷ 32 0110 Encoding 6. MTIM clock source ÷ 64 0111 Encoding 7. MTIM clock source ÷ 128 1000 Encoding 8. MTIM clock source ÷ 256 All other encodings default to MTIM clock source ÷ 256.

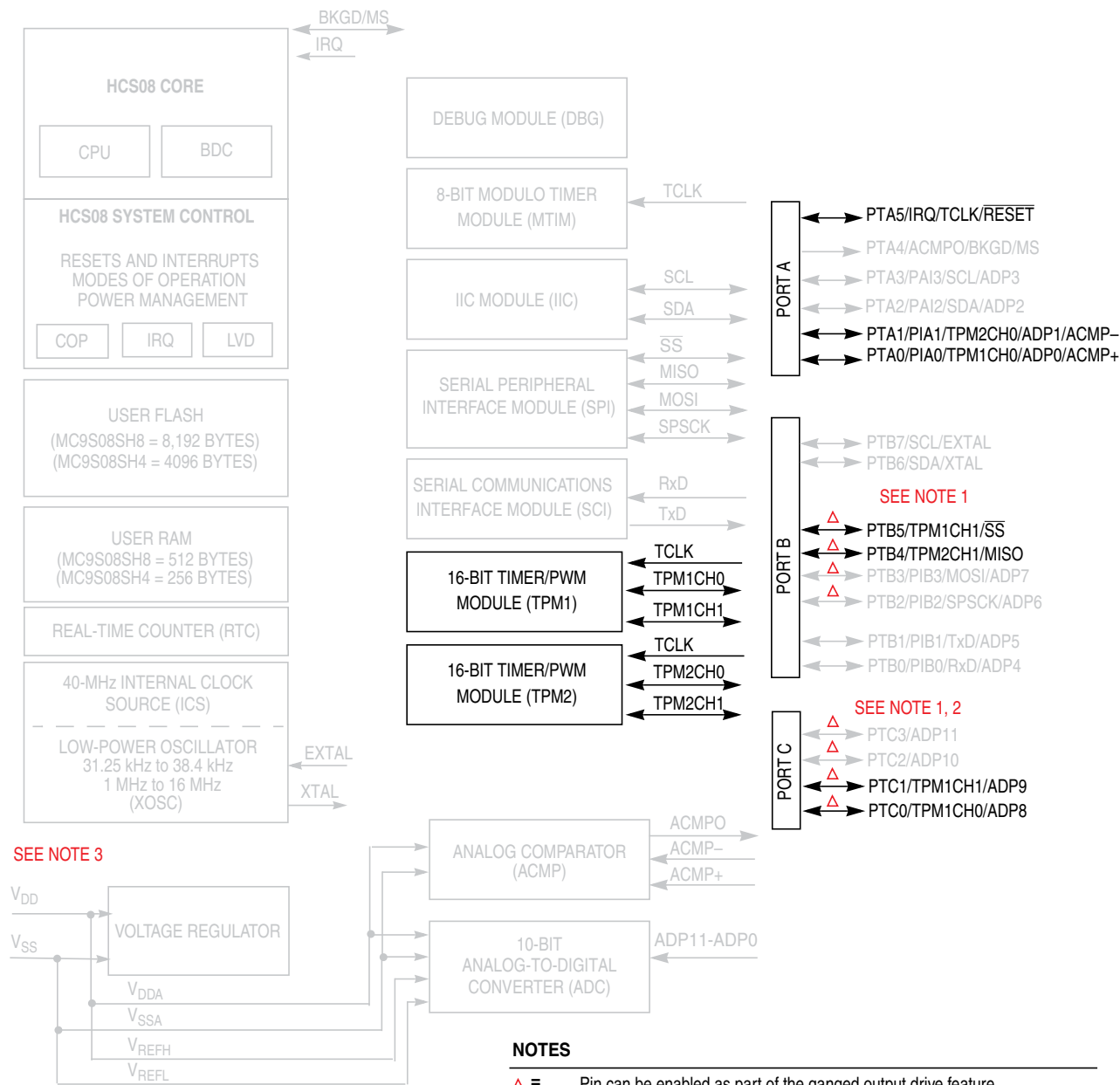


Figure 16-1. MC9S08SH8 Block Diagram Highlighting the TPM Modules

EPWM mode
TPMxMODH:TPMxMODL = 0x0007
TPMxCnVH:TPMxCnVL = 0x0005

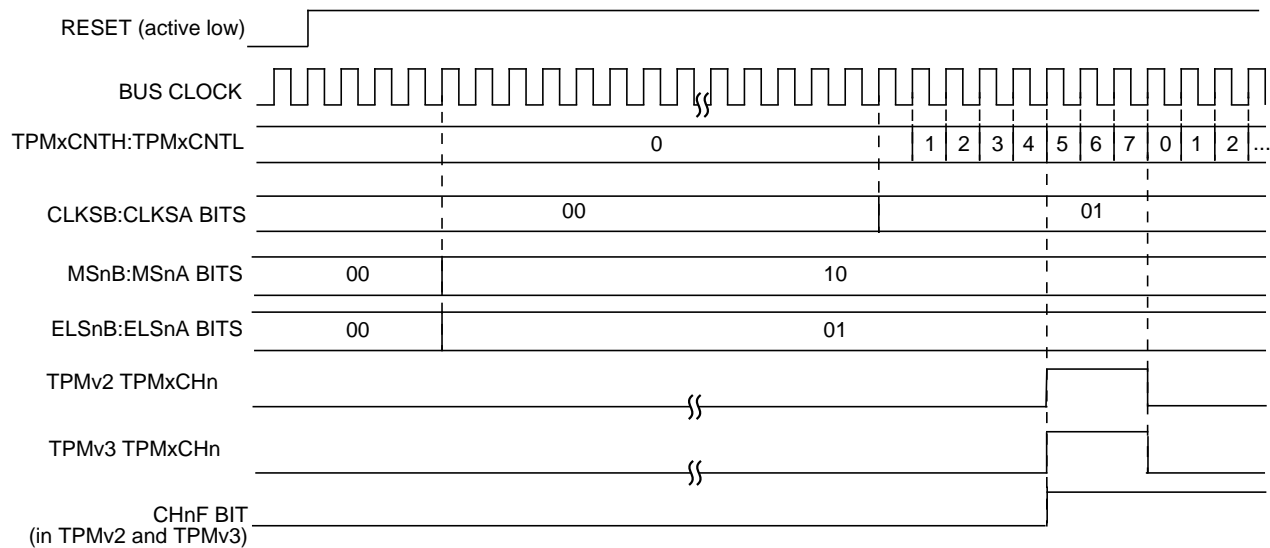


Figure 0-2. Generation of low-true EPWM signal by TPM v2 and v3 after the reset

The following procedure can be used in TPM v3 (when the channel pin is also a port pin) to emulate the high-true EPWM generated by TPM v2 after the reset.

- ...
- configure the channel pin as output port pin and set the output pin;
- configure the channel to generate the EPWM signal but keep ELSnB:ELSnA as 00;
- configure the other registers (TPMxMODH, TPMxMODL, TPMxCnVH, TPMxCnVL, ...);
- configure CLKSb:CLKSA bits (TPM v3 starts to generate the high-true EPWM signal, however TPM does not control the channel pin, so the EPWM signal is not available);
- wait until the TOF is set (or use the TOF interrupt);
- enable the channel output by configuring ELSnB:ELSnA bits (now EPWM signal is available);
- ...

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 DBGTC 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

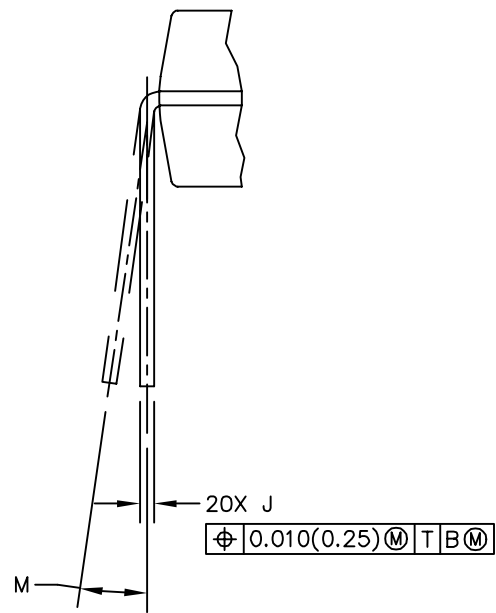
The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGTC register selects one of nine trigger modes. When TRGSEL = 1 in the DBGTC register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGTC 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 DBGTC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGSR. 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 DBGGEN in DBGTC.

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.



VIEW D

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.		MECHANICAL OUTLINE		PRINT VERSION NOT TO SCALE	
TITLE: 20LD .300 PDIP			DOCUMENT NO: 98ASB42899B		REV: B
			CASE NUMBER: 738C-01		24 MAY 2005
			STANDARD: NON-JEDEC		