



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	Obsolete
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	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	20-DIP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08sh4cpj

Table 4-7. FLASH Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV (Decimal)	f_{CLK}	Program/Erase Timing Pulse (5 μs Min, 6.7 μs Max)
20 MHz	1	12	192.3 kHz	5.2 μs
10 MHz	0	49	200 kHz	5 μs
8 MHz	0	39	200 kHz	5 μs
4 MHz	0	19	200 kHz	5 μs
2 MHz	0	9	200 kHz	5 μs
1 MHz	0	4	200 kHz	5 μs
200 kHz	0	0	200 kHz	5 μs
150 kHz	0	0	150 kHz	6.7 μs

4.7.2 FLASH Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from FLASH into FOPT. To change the value in this register, erase and reprogram the NVOPT location in FLASH memory as usual and then issue a new MCU reset.

	7	6	5	4	3	2	1	0
R	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
W								

Reset This register is loaded from nonvolatile location NVOPT during reset.


 = Unimplemented or Reserved

Figure 4-6. FLASH Options Register (FOPT)

Table 4-8. FOPT Register Field Descriptions

Field	Description
7 KEYEN	Backdoor Key Mechanism Enable — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from user (secured) firmware. BDM commands cannot be used to write key comparison values that would unlock the backdoor key. For more detailed information about the backdoor key mechanism, refer to Section 4.6, “Security.” 0 No backdoor key access allowed. 1 If user firmware writes an 8-byte value that matches the nonvolatile backdoor key (NVBACKKEY through NVBACKKEY+7 in that order), security is temporarily disengaged until the next MCU reset.
6 FNORED	Vector Redirection Disable — When this bit is 1, then vector redirection is disabled. 0 Vector redirection enabled. 1 Vector redirection disabled.
1:0 SEC0[1:0]	Security State Code — This 2-bit field determines the security state of the MCU as shown in Table 4-9 . When the MCU is secure, the contents of RAM and FLASH memory cannot be accessed by instructions from any unsecured source including the background debug interface. SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of FLASH. For more detailed information about security, refer to Section 4.6, “Security.”

6.6.1.8 Port A Interrupt Edge Select Register (PTAES)

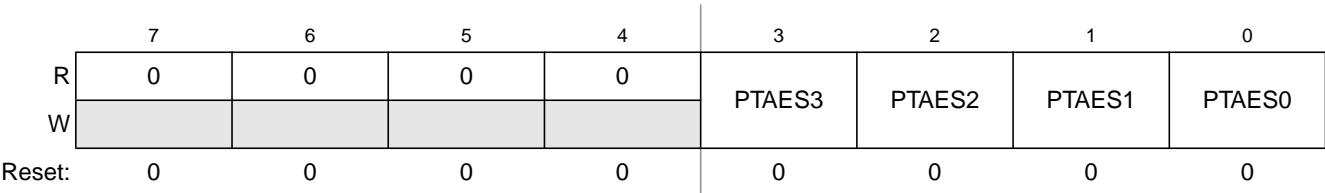


Figure 6-10. Port A Edge Select Register (PTAES)

Table 6-9. PTAES Register Field Descriptions

Field	Description
3:0 PTAES[3:0]	<p>Port A Edge Selects — Each of the PTAESn bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled.</p> <p>0 A pull-up device is connected to the associated pin and detects falling edge/low level for interrupt generation.</p> <p>1 A pull-down device is connected to the associated pin and detects rising edge/high level for interrupt generation.</p>

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

7.4.5 BGND Instruction

The BGND instruction is new to the HCS08 compared to the M68HC08. BGND would not be used in normal user programs because it forces the CPU to stop processing user instructions and enter the active background mode. The only way to resume execution of the user program is through reset or by a host debug system issuing a GO, TRACE1, or TAGGO serial command through the background debug interface.

Software-based breakpoints can be set by replacing an opcode at the desired breakpoint address with the BGND opcode. When the program reaches this breakpoint address, the CPU is forced to active background mode rather than continuing the user program.

Table 7-2. Instruction Set Summary (Sheet 3 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
BPL <i>rel</i>	Branch if Plus (if N = 0)	REL	2A rr	3	ppp	- 1 1 -	- - - -
BRA <i>rel</i>	Branch Always (if I = 1)	REL	20 rr	3	ppp	- 1 1 -	- - - -
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear (if (Mn) = 0)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 dd rr 03 dd rr 05 dd rr 07 dd rr 09 dd rr 0B dd rr 0D dd rr 0F dd rr	5 5 5 5 5 5 5 5	rpppp rpppp rpppp rpppp rpppp rpppp rpppp rpppp	- 1 1 -	- - - - ↑
BRN <i>rel</i>	Branch Never (if I = 0)	REL	21 rr	3	ppp	- 1 1 -	- - - -
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set (if (Mn) = 1)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 dd rr 02 dd rr 04 dd rr 06 dd rr 08 dd rr 0A dd rr 0C dd rr 0E dd rr	5 5 5 5 5 5 5 5	rpppp rpppp rpppp rpppp rpppp rpppp rpppp rpppp	- 1 1 -	- - - - ↓
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory (Mn ← 1)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 dd 12 dd 14 dd 16 dd 18 dd 1A dd 1C dd 1E dd	5 5 5 5 5 5 5 5	rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp	- 1 1 -	- - - -
BSR <i>rel</i>	Branch to Subroutine PC ← (PC) + \$0002 push (PCL); SP ← (SP) – \$0001 push (PCH); SP ← (SP) – \$0001 PC ← (PC) + <i>rel</i>	REL	AD rr	5	ssppp	- 1 1 -	- - - -
CBEQ <i>opr8a,rel</i> CBEQA # <i>opr8i,rel</i> CBEQX # <i>opr8i,rel</i> CBEQ <i>opr8,X+,rel</i> CBEQ <i>X+,rel</i> CBEQ <i>opr8,SP,rel</i>	Compare and... Branch if (A) = (M) Branch if (A) = (M) Branch if (X) = (M) Branch if (A) = (M) Branch if (A) = (M) Branch if (A) = (M)	DIR IMM IMM IX1+ IX+ SP1	31 dd rr 41 ii rr 51 ii rr 61 ff rr 71 rr 9E 61 ff rr	5 4 4 5 5 6	rpppp pppp pppp rpppp rfppp prpppp	- 1 1 -	- - - -
CLC	Clear Carry Bit (C ← 0)	INH	98	1	p	- 1 1 -	- - - 0
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	- 1 1 -	0 - - -
CLR <i>opr8a</i> CLRA CLR X CLR H CLR <i>opr8,X</i> CLR <i>,X</i> CLR <i>opr8,SP</i>	Clear M ← \$00 A ← \$00 X ← \$00 H ← \$00 M ← \$00 M ← \$00 M ← \$00	DIR INH INH INH IX1 IX SP1	3F dd 4F 5F 8C 6F ff 7F 9E 6F ff	5 1 1 1 5 4 6	rfwpp p p p rfwpp rfwp prfwpp	0 1 1 -	- 0 1 -

Table 7-2. Instruction Set Summary (Sheet 8 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
SUB #opr8i SUB opr8a SUB opr16a SUB oprx16,X SUB oprx8,X SUB ,X SUB oprx16,SP SUB oprx8,SP	Subtract $A \leftarrow (A) - (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 ii B0 dd C0 hh ll D0 ee ff E0 ff F0 9E D0 ee ff 9E E0 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	\uparrow 1 1 -	- \uparrow \uparrow \uparrow
SWI	Software Interrupt $PC \leftarrow (PC) + \$0001$ Push (PCL); $SP \leftarrow (SP) - \$0001$ Push (PCH); $SP \leftarrow (SP) - \$0001$ Push (X); $SP \leftarrow (SP) - \$0001$ Push (A); $SP \leftarrow (SP) - \$0001$ Push (CCR); $SP \leftarrow (SP) - \$0001$ $I \leftarrow 1$; PCH \leftarrow Interrupt Vector High Byte PCL \leftarrow Interrupt Vector Low Byte	INH	83	11	sssssvvfppp	- 1 1 -	1 - - -
TAP	Transfer Accumulator to CCR $CCR \leftarrow (A)$	INH	84	1	p	\uparrow 1 1 \uparrow	\uparrow \uparrow \uparrow \uparrow
TAX	Transfer Accumulator to X (Index Register Low) $X \leftarrow (A)$	INH	97	1	p	- 1 1 -	- - - -
TPA	Transfer CCR to Accumulator $A \leftarrow (CCR)$	INH	85	1	p	- 1 1 -	- - - -
TST opr8a TSTA TSTX TST oprx8,X TST ,X TST oprx8,SP	Test for Negative or Zero (M) - \$00 (A) - \$00 (X) - \$00 (M) - \$00 (M) - \$00 (M) - \$00	DIR INH INH IX1 IX SP1	3D dd 4D 5D 6D ff 7D 9E 6D ff	4 1 1 4 3 5	rfpp p p rfpp rfp prfpp	0 1 1 -	- \uparrow \uparrow -
TSX	Transfer SP to Index Reg. $H:X \leftarrow (SP) + \$0001$	INH	95	2	fp	- 1 1 -	- - - -
TXA	Transfer X (Index Reg. Low) to Accumulator $A \leftarrow (X)$	INH	9F	1	p	- 1 1 -	- - - -

Table 7-2. Instruction Set Summary (Sheet 9 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR					
						V	1	1	H	I	N Z C
TXS	Transfer Index Reg. to SP $SP \leftarrow (H:X) - \$0001$	INH	94	2	f _p	-	1	1	-	-	- - -
WAIT	Enable Interrupts; Wait for Interrupt I bit \leftarrow 0; Halt CPU	INH	8F	2+	f _p . . .	-	1	1	-	0	- - -

Source Form: Everything in the source forms columns, *except expressions in italic characters*, is literal information which must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic and the characters (#, () and +) are always a literal characters.

- n* Any label or expression that evaluates to a single integer in the range 0-7.
- opr8i* Any label or expression that evaluates to an 8-bit immediate value.
- opr16i* Any label or expression that evaluates to a 16-bit immediate value.
- opr8a* Any label or expression that evaluates to an 8-bit direct-page address (\$00xx).
- opr16a* Any label or expression that evaluates to a 16-bit address.
- opr8* Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing.
- opr16* Any label or expression that evaluates to a 16-bit value, used for indexed addressing.
- rel* Any label or expression that refers to an address that is within -128 to +127 locations from the start of the next instruction.

Operation Symbols:

- A Accumulator
- CCR Condition code register
- H Index register high byte
- M Memory location
- n* Any bit
- opr* Operand (one or two bytes)
- PC Program counter
- PCH Program counter high byte
- PCL Program counter low byte
- rel* Relative program counter offset byte
- SP Stack pointer
- SPL Stack pointer low byte
- X Index register low byte
- & Logical AND
- | Logical OR
- ⊕ Logical EXCLUSIVE OR
- () Contents of
- + Add
- Subtract, Negation (two's complement)
- × Multiply
- ÷ Divide
- # Immediate value
- ← Loaded with
- :

CCR Bits:

- V Overflow bit
- H Half-carry bit
- I Interrupt mask
- N Negative bit
- Z Zero bit
- C Carry/borrow bit

Addressing Modes:

- DIR Direct addressing mode
- EXT Extended addressing mode
- IMM Immediate addressing mode
- INH Inherent addressing mode
- IX Indexed, no offset addressing mode
- IX1 Indexed, 8-bit offset addressing mode
- IX2 Indexed, 16-bit offset addressing mode
- IX+ Indexed, no offset, post increment addressing mode
- IX1+ Indexed, 8-bit offset, post increment addressing mode
- REL Relative addressing mode
- SP1 Stack pointer, 8-bit offset addressing mode
- SP2 Stack pointer 16-bit offset addressing mode

Cycle-by-Cycle Codes:

- f Free cycle. This indicates a cycle where the CPU does not require use of the system buses. An f cycle is always one cycle of the system bus clock and is always a read cycle.
- p Program fetch; read from next consecutive location in program memory
- r Read 8-bit operand
- s Push (write) one byte onto stack
- u Pop (read) one byte from stack
- v Read vector from \$FFxx (high byte first)
- w Write 8-bit operand

CCR Effects:

- ↑ Set or cleared
- Not affected
- U Undefined

Table 9-11. APCTL3 Register Field Descriptions (continued)

Field	Description
1 ADPC17	ADC Pin Control 17 — ADPC17 is used to control the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	ADC Pin Control 16 — ADPC16 is used to control the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

9.4 Functional Description

The ADC module is disabled during reset or when the ADCH bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle, the module is in its lowest power state.

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. The selected channel voltage is converted by a successive approximation algorithm into an 11-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

When the conversion is completed, the result is placed in the data registers (ADCRH and ADCRL). In 10-bit mode, the result is rounded to 10 bits and placed in ADCRH and ADCRL. In 8-bit mode, the result is rounded to 8 bits and placed in ADCRL. The conversion complete flag (COCO) is then set and an interrupt is generated if the conversion complete interrupt has been enabled (AIEN = 1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of its compare registers. The compare function is enabled by setting the ACFE bit and operates in conjunction with any of the conversion modes and configurations.

9.4.1 Clock Select and Divide Control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADICLK bits.

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by 2. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK) – This clock is generated from a clock source within the ADC module. When selected as the clock source this clock remains active while the MCU is in wait or stop3 mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC will not perform according to specifications. If the available clocks

9.4.7.2 Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the ADC interrupt is enabled (AIEN = 1).

NOTE

It is possible for the ADC module to wake the system from low power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure that the data transfer blocking mechanism (discussed in [Section 9.4.4.2, "Completing Conversions"](#)) is cleared when entering stop3 and continuing ADC conversions.

9.4.8 MCU Stop1 and Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters either stop1 or stop2 mode. All module registers contain their reset values following exit from stop1 or stop2. Therefore the module must be re-enabled and re-configured following exit from stop1 or stop2.

9.5 Initialization Information

This section gives an example which provides some basic direction on how a user would initialize and configure the ADC module. The user has the flexibility of choosing between configuring the module for 8-bit or 10-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to [Table 9-6](#), [Table 9-7](#), and [Table 9-8](#) for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

9.5.1 ADC Module Initialization Example

9.5.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.



Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full ($RDRF = 1$), it gets the data from the receive data register by reading $SCIxD$. The $RDRF$ flag is cleared by reading $SCIxS1$ while $RDRF = 1$ and then reading $SCIxD$.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, $SCIxS1$ must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading $SCIxS1$ while $IDLE = 1$ and then reading $SCIxD$. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set $RDRF$.

If the associated error was detected in the received character that caused $RDRF$ to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as $RDRF$. These flags are not set in overrun cases.

If $RDRF$ was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the $RXEDGIF$ flag to set. The $RXEDGIF$ flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled ($RE = 1$).

14.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

14.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in $SCIxC1$. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in $SCIxC3$. For the receiver, the ninth bit is held in R8 in $SCIxC3$.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to $SCIxD$.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from $SCIxD$ to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

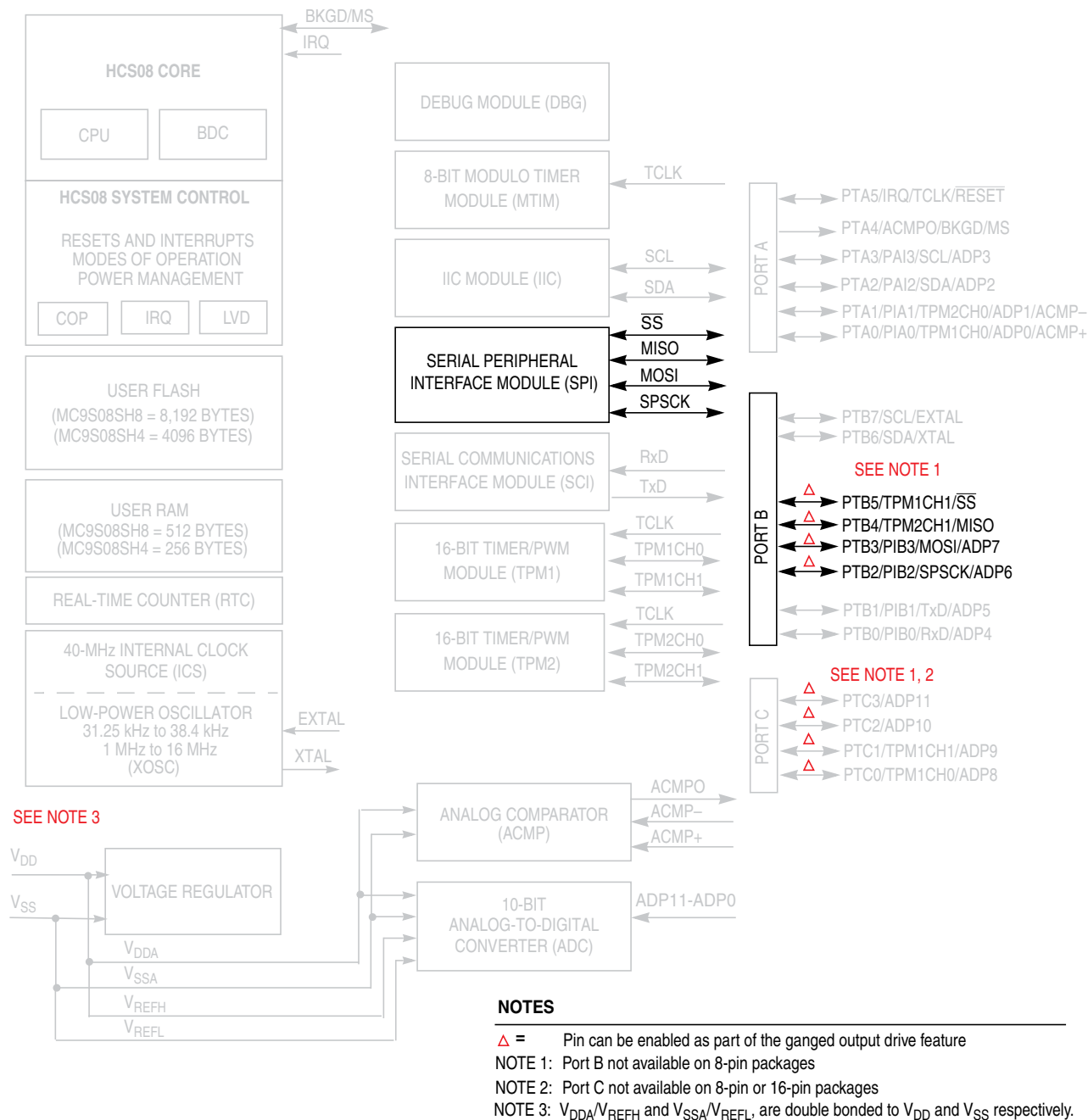


Figure 15-1. MC9S08SH8 Block Diagram Highlighting the SPI Module

in LSBFE. Both variations of SPSCCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The \overline{SS} OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master \overline{SS} output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCCK cycle after the end of the eighth bit time of the transfer. The \overline{SS} IN waveform applies to the slave select input of a slave.

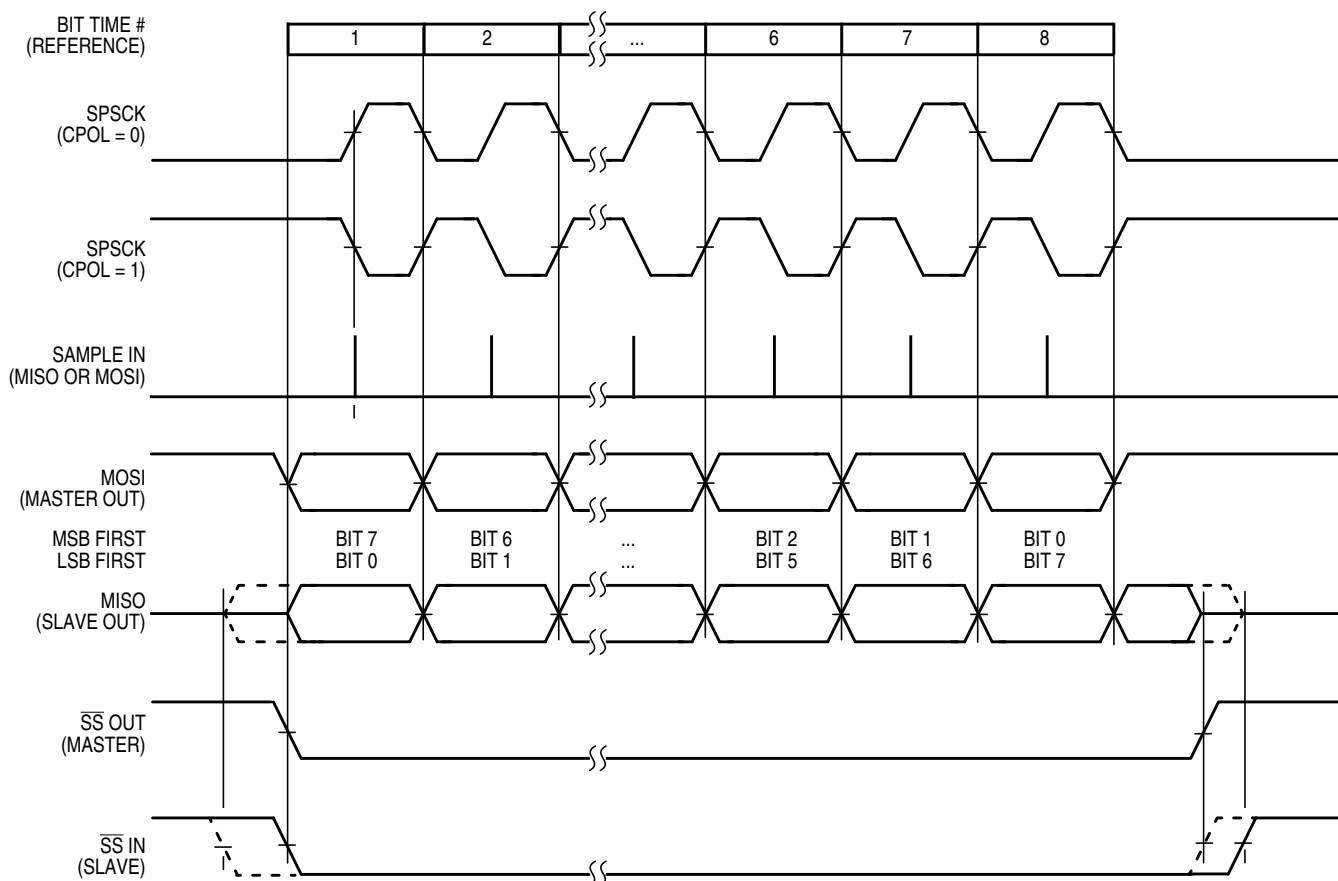


Figure 15-11. SPI Clock Formats (CPHA = 0)

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when \overline{SS} goes to active low. The first SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's \overline{SS} input must go to its inactive high level between transfers.

16.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

16.2.1.2 TPMxCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKSB:CLKSA = 0:0) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMxCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMxCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMxCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

In this mode and if (CLKSB:CLKSA not = 0:0), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer at the next change of the TPM counter (end of the prescaler counting) after the second byte is written. Instead, the TPM v2 always updates these registers when their second byte is written.

The following procedure can be used in the TPM v3 to verify if the TPMxCnVH:L registers were updated with the new value that was written to these registers (value in their write buffer).

...

write the new value to TPMxCnVH:L;

read TPMxCnVH and TPMxCnVL registers;

while (the read value of TPMxCnVH:L is different from the new value written to TPMxCnVH:L)

begin

 read again TPMxCnVH and TPMxCnVL;

end

...

In this point, the TPMxCnVH:L registers were updated, so the program can continue and, for example, write to TPMxC0SC without cancelling the previous write to TPMxCnVH:L registers.

— Edge-Aligned PWM (Section 16.4.2.3, “Edge-Aligned PWM Mode)

In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from \$FFFE to \$FFFF. Instead, the TPM v2 makes this update after that the both bytes were written and when the TPM counter changes from TPMxMODH:L to \$0000.

— Center-Aligned PWM (Section 16.4.2.4, “Center-Aligned PWM Mode)

In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from \$FFFE to \$FFFF. Instead, the TPM v2 makes this update after that the both bytes were written and when the TPM counter changes from TPMxMODH:L to (TPMxMODH:L - 1).

5. Center-Aligned PWM (Section 16.4.2.4, “Center-Aligned PWM Mode)

— TPMxCnVH:L = TPMxMODH:L [SE110-TPM case 1]

In this case, the TPM v3 produces 100% duty cycle. Instead, the TPM v2 produces 0% duty cycle.

— TPMxCnVH:L = (TPMxMODH:L - 1) [SE110-TPM case 2]

In this case, the TPM v3 produces almost 100% duty cycle. Instead, the TPM v2 produces 0% duty cycle.

— TPMxCnVH:L is changed from 0x0000 to a non-zero value [SE110-TPM case 3 and 5]

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct communications speed to use for BDC communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

The target, upon detecting the SYNC request from the host (which is a much longer low time than would ever occur during normal BDC communications):

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

17.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

The on-chip debug module (DBG) includes circuitry for two additional hardware breakpoints that are more flexible than the simple breakpoint in the BDC module.

The average chip-junction temperature (T_J) in $^{\circ}\text{C}$ can be obtained from:

$$T_J = T_A + (P_D \times \theta_{JA}) \quad \text{Eqn. A-1}$$

where:

T_A = Ambient temperature, $^{\circ}\text{C}$

θ_{JA} = Package thermal resistance, junction-to-ambient, $^{\circ}\text{C}/\text{W}$

$P_D = P_{\text{int}} + P_{\text{I/O}}$

$P_{\text{int}} = I_{\text{DD}} \times V_{\text{DD}}$, Watts — chip internal power

$P_{\text{I/O}}$ = Power dissipation on input and output pins — user determined

For most applications, $P_{\text{I/O}} \ll P_{\text{int}}$ and can be neglected. An approximate relationship between P_D and T_J (if $P_{\text{I/O}}$ is neglected) is:

$$P_D = K \div (T_J + 273^{\circ}\text{C}) \quad \text{Eqn. A-2}$$

Solving Equation A-1 and Equation A-2 for K gives:

$$K = P_D \times (T_A + 273^{\circ}\text{C}) + \theta_{JA} \times (P_D)^2 \quad \text{Eqn. A-3}$$

where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A . Using this value of K, the values of P_D and T_J can be obtained by solving Equation A-1 and Equation A-2 iteratively for any value of T_A .

A.7 Supply Current Characteristics

This section includes information about power supply current in various operating modes.

Table A-7. Supply Current Characteristics

Num	C	Parameter	Symbol	V _{DD} (V)	Typ ¹	Max ²	Unit
1	C	Run supply current ³ measured at (CPU clock = 4 MHz, f _{BUS} = 2 MHz)	R _I DD	5	1.1	1.5	mA
	C			3	1	1.5	
2	P	Run supply current ³ measured at (CPU clock = 16 MHz, f _{BUS} = 8 MHz)	R _I DD	5	3.9	5	mA
	C			3	3.9	5	
3	C	Run supply current ⁴ measured at (CPU clock = 32 MHz, f _{BUS} = 16 MHz)	R _I DD	5	7.25	7.7	mA
	C			3	7.15	7.6	
4		Stop3 mode supply current	S3I _{DD}				
	C	−40 °C (C & M suffix)		5	1.1	—	μA
	P	25 °C (All parts)			1.5	—	
	P ⁵	85 °C (C suffix only)			9.0	26	
	P ⁵	125 °C (M suffix only)			45.2	130	
	C	C and M suffix −40 °C		3	1.0	—	μA
	C	All parts 25 °C			1.4	—	
	C	C suffix only 85 °C			7.8	19	
	C	M suffix only 125 °C			40.1	95	
	5			Stop2 mode supply current	S2I _{DD}		
C		C and M suffix −40 °C	5	1.1		—	μA
P		All parts 25 °C		1.4		—	
P ⁵		C suffix only 85 °C		6.8		22	
P ⁵		M suffix only 125 °C		32.7		99	
C		C and M suffix −40 °C	3	1.0		—	μA
C		All parts 25 °C		1.3		—	
C		C suffix only 85 °C		5.8		16	
C		M suffix only 125 °C		28.3		76	
6		C	RTC adder to stop2 or stop3 ⁶	S23I _{DDRTI}		5	300
	3				300	500	nA
7	C	LVD adder to stop3 (LVDE = LVDSE = 1)	S3I _{DDLVD}	5	110	180	μA
				3	90	160	μA
8	C	Adder to stop3 for oscillator enabled ⁷ (EREFSTEN = 1)	S3I _{DDOSC}	5, 3	5	8	μA

A.12.2 TPM/MTIM Module Timing

Synchronizer circuits determine the shortest input pulses that can be recognized or the fastest clock that can be used as the optional external source to the timer counter. These synchronizers operate from the current bus rate clock.

Table A-14. TPM Input Timing

Num	C	Rating	Symbol	Min	Max	Unit
1	—	External clock frequency ($1/t_{\text{TCLK}}$)	f_{TCLK}	dc	$f_{\text{Bus}}/4$	MHz
2	—	External clock period	t_{TCLK}	4	—	t_{cyc}
3	—	External clock high time	t_{clkh}	1.5	—	t_{cyc}
4	—	External clock low time	t_{clkl}	1.5	—	t_{cyc}
5	—	Input capture pulse width	t_{ICPW}	1.5	—	t_{cyc}

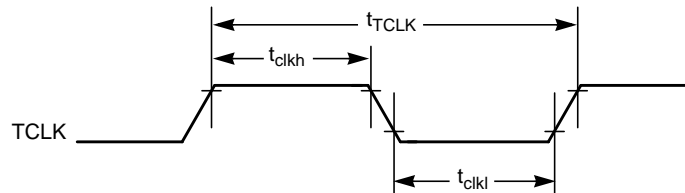


Figure A-12. Timer External Clock

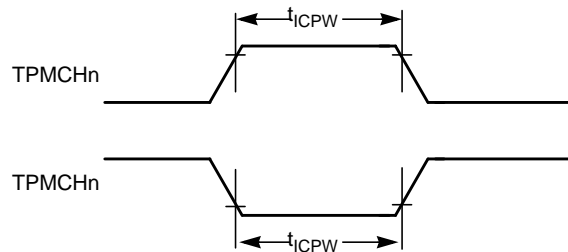


Figure A-13. Timer Input Capture Pulse

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. 2006-2008. All rights reserved.