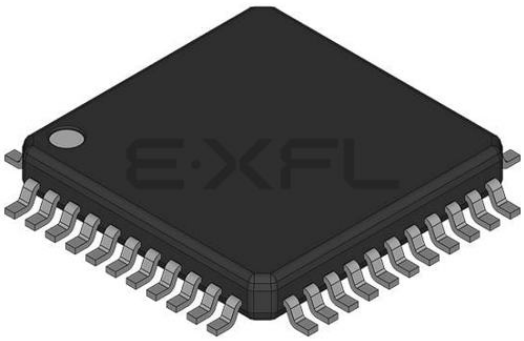


Welcome to E-XFL.COM

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	I²C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	34
Program Memory Size	48KB (48K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LQFP
Supplier Device Package	44-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s08aw48cfger

Chapter 5
Resets, Interrupts, and System Configuration

  - The output of the digitally-controlled oscillator (DCO) in the frequency-locked loop sub-module
  - Control bits inside the ICG determine which source is connected.
- FFE is a control signal generated inside the ICG. If the frequency of ICGOUT > 4 × the frequency of ICGERCLK, this signal is a logic 1 and the fixed-frequency clock will be ICGERCLK/2. Otherwise the fixed-frequency clock will be BUSCLK.
- ICGLCLK — Development tools can select this internal self-clocked source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow.
- ICGERCLK — External reference clock can be selected as the real-time interrupt clock source. Can also be used as the ALTCLK input to the ADC module.

2.2 Device Pin Assignment

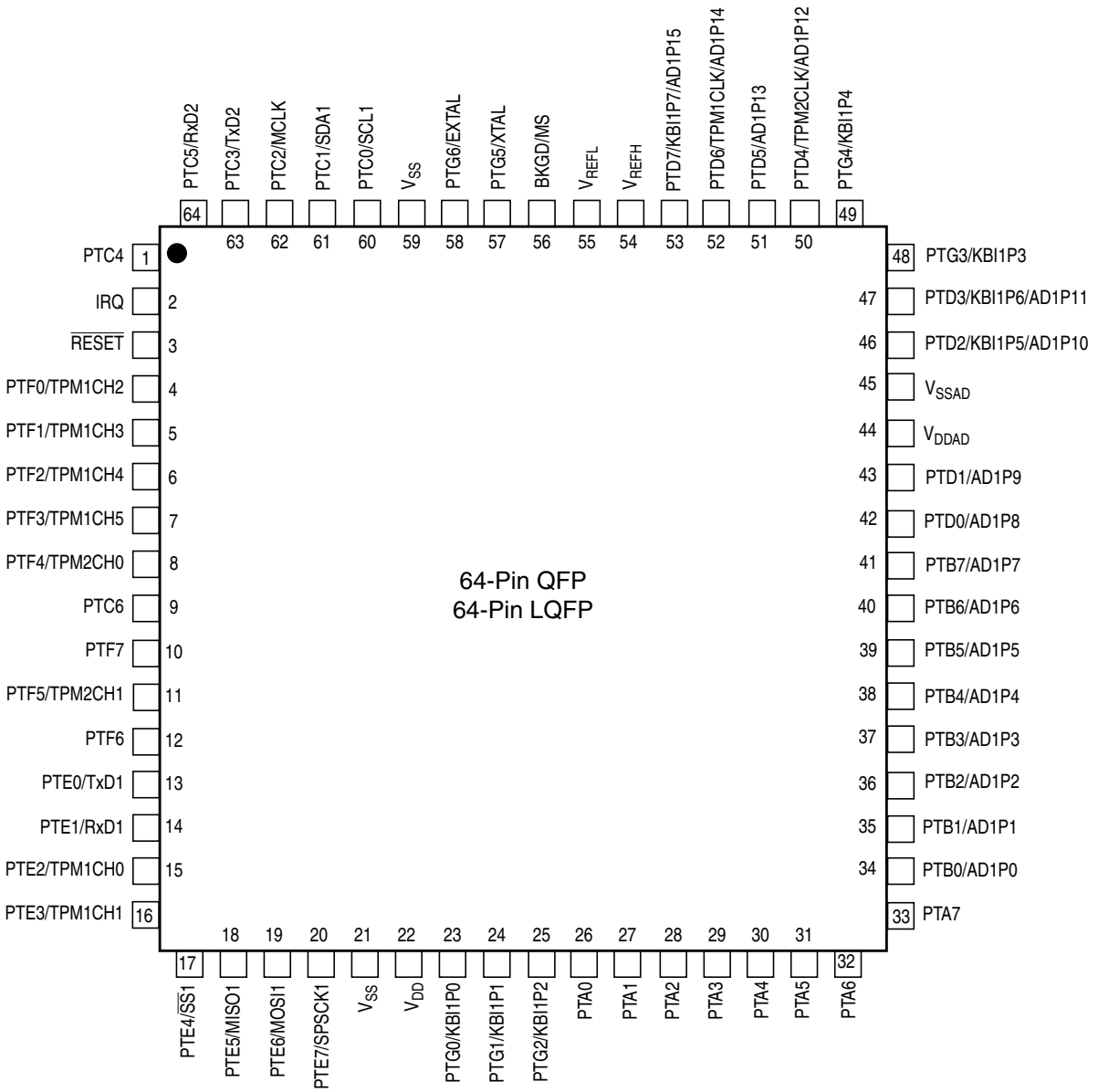


Figure 2-1. MC9S08AW60 Series in 64-Pin QFP/LQFP Package

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's 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's 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 MC9S08AW60 Series 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 Chapter 15, "Development Support."

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 the STOPE bit in the system option register is set. In both stop modes, all internal clocks are halted. If the STOPE bit is not set when

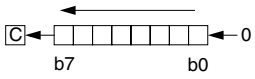
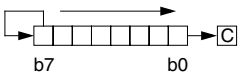
7.4.5 BGND Instruction

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.

IX = 16-bit indexed no offset
IX+ = 16-bit indexed no offset, post increment (CBEQ and MOV only)
IX1 = 16-bit indexed with 8-bit offset from H:X
IX1+ = 16-bit indexed with 8-bit offset, post increment (CBEQ only)
IX2 = 16-bit indexed with 16-bit offset from H:X
REL = 8-bit relative offset
SP1 = Stack pointer with 8-bit offset
SP2 = Stack pointer with 16-bit offset

Table 7-2. HCS08 Instruction Set Summary (Sheet 1 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
ADC #*opr8i* ADC *opr8a* ADC *opr16a* ADC *oprx16*,X ADC *oprx8*,X ADC ,X ADC *oprx16*,SP ADC *oprx8*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 B9 C9 D9 E9 F9 9ED9 9EE9	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
ADD #*opr8i* ADD *opr8a* ADD *opr16a* ADD *oprx16*,X ADD *oprx8*,X ADD ,X ADD *oprx16*,SP ADD *oprx8*,SP	Add without Carry	A ← (A) + (M)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	AB BB CB DB EB FB 9EDB 9EEB	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
AIS #*opr8i*	Add Immediate Value (Signed) to Stack Pointer	SP ← (SP) + (M) M is sign extended to a 16-bit value	–	–	–	–	–	–	IMM	A7	ii	2
AIX #*opr8i*	Add Immediate Value (Signed) to Index Register (H:X)	H:X ← (H:X) + (M) M is sign extended to a 16-bit value	–	–	–	–	–	–	IMM	AF	ii	2
AND #*opr8i* AND *opr8a* AND *opr16a* AND *oprx16*,X AND *oprx8*,X AND ,X AND *oprx16*,SP AND *oprx8*,SP	Logical AND	A ← (A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 B4 C4 D4 E4 F4 9ED4 9EE4	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
ASL *opr8a* ASLA ASLX ASL *oprx8*,X ASL ,X ASL *oprx8*,SP	Arithmetic Shift Left (Same as LSL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 5 4 6
ASR *opr8a* ASRA ASRX ASR *oprx8*,X ASR ,X ASR *oprx8*,SP	Arithmetic Shift Right		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	5 1 1 5 4 6
BCC *rel*	Branch if Carry Bit Clear	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3

8.4.11 Fixed Frequency Clock

The ICG provides a fixed frequency clock output, XCLK, for use by on-chip peripherals. This output is equal to the internal bus clock, BUSCLK, in all modes except FEE. In FEE mode, XCLK is equal to ICGERCLK ÷ 2 when the following conditions are met:

- $(P \times N) \div R \geq 4$ where P is determined by RANGE (see Table 8-11), N and R are determined by MFD and RFD respectively (see Table 8-12).
- LOCK = 1.

If the above conditions are not true, then XCLK is equal to BUSCLK.

When the ICG is in either FEI or SCM mode, XCLK is turned off. Any peripherals which can use XCLK as a clock source must not do so when the ICG is in FEI or SCM mode.

8.4.12 High Gain Oscillator

The oscillator has the option of running in a high gain oscillator (HGO) mode, which improves the oscillator's resistance to EMC noise when running in FBE or FEE modes. This option is selected by writing a 1 to the HGO bit in the ICGC1 register. HGO is used with both the high and low range oscillators but is only valid when REFS = 1 in the ICGC1 register. When HGO = 0, the standard low-power oscillator is selected. This bit is writable only once after any reset.

8.5 Initialization/Application Information

8.5.1 Introduction

The section is intended to give some basic direction on which configuration a user would want to select when initializing the ICG. For some applications, the serial communication link may dictate the accuracy of the clock reference. For other applications, lowest power consumption may be the chief clock consideration. Still others may have lowest cost as the primary goal. The ICG allows great flexibility in choosing which is best for any application.

Table 8-10. ICG Configuration Consideration

	Clock Reference Source = Internal	Clock Reference Source = External
FLL Engaged	**FEI** 4 MHz < f_{Bus} < 20 MHz. Medium power (will be less than FEE if oscillator range = high) Good clock accuracy (After IRG is trimmed) **Lowest system cost** (no external components required) IRG is on. DCO is on. [1]	**FEE** 4 MHz < f_{Bus} < 20 MHz Medium power (will be less than FEI if oscillator range = low) High clock accuracy Medium/High system cost (crystal, resonator or external clock source required) IRG is off. DCO is on.
FLL Bypassed	**SCM** This mode is mainly provided for quick and reliable system startup. 3 MHz < f_{Bus} < 5 MHz (default). 3 MHz < f_{Bus} < 20 MHz (via filter bits). Medium power Poor accuracy. IRG is off. DCO is on and open loop.	**FBE** f_{Bus} range ≤ 8 MHz when crystal or resonator is used. **Lowest power** Highest clock accuracy Medium/High system cost (Crystal, resonator or external clock source required) IRG is off. DCO is off.

1 The IRG typically consumes 100 µA. The FLL and DCO typically consumes 0.5 to 2.5 mA, depending upon output frequency. For minimum power consumption and minimum jitter, choose N and R to be as small as possible.

The following sections contain initialization examples for various configurations.

NOTE

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

Important configuration information is repeated here for reference.

Table 8-11. ICGOUT Frequency Calculation Options

Clock Scheme	f_{ICGOUT} [1]	P	Note
SCM — self-clocked mode (FLL bypassed internal)	$f_{ICGDCLK}$ / R	NA	Typical f_{ICGOUT} = 8 MHz immediately after reset
FBE — FLL bypassed external	f_{ext} / R	NA	
FEI — FLL engaged internal	(f_{IRG} / 7)* 64 * N / R	64	Typical f_{IRG} = 243 kHz
FEE — FLL engaged external	f_{ext} * P * N / R	Range = 0 ; P = 64 Range = 1; P = 1	

1 Ensure that $f_{ICGDCLK}$, which is equal to f_{ICGOUT} * R, does not exceed $f_{ICGDCLKmax}$.

Table 8-12. MFD and RFD Decode Table

MFD Value	Multiplication Factor (N)
000	4
001	6
010	8
011	10
100	12

RFD	Division Factor (R)
000	÷1
001	÷2
010	÷4
011	÷8
100	÷16

Bits 11:0 FLT No need for user initialization

ICGTRM = $xx

Bits 7:0 TRIM Only need to write when trimming internal oscillator; not used when external crystal is clock source

Figure 8-14 shows flow charts for three conditions requiring ICG initialization.

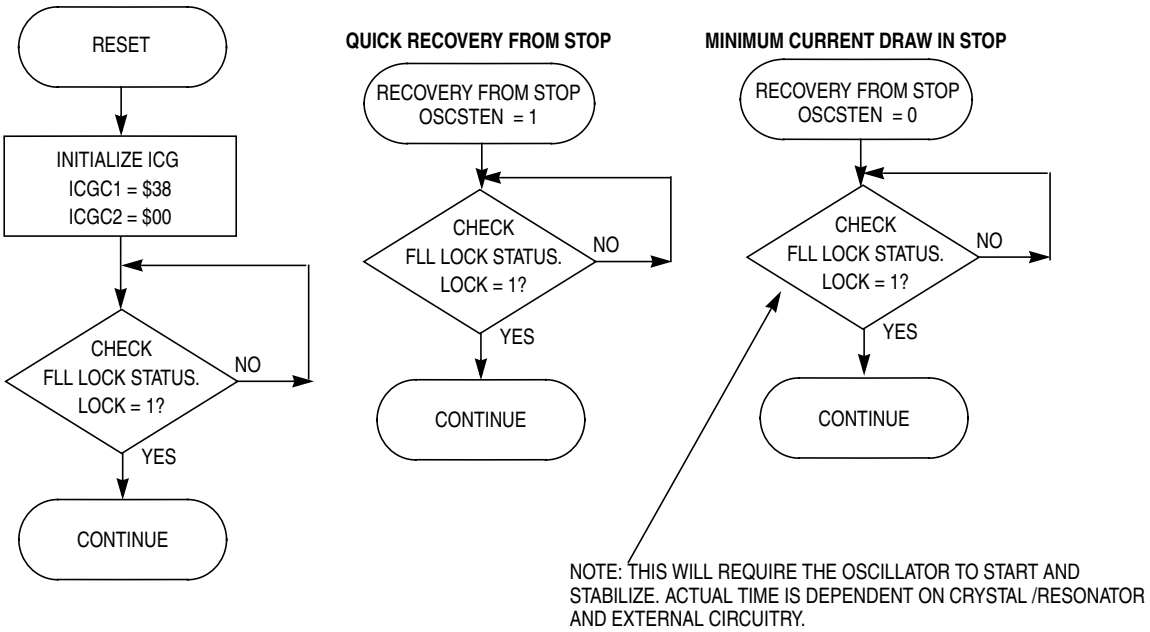


Figure 8-14. ICG Initialization for FEE in Example #1

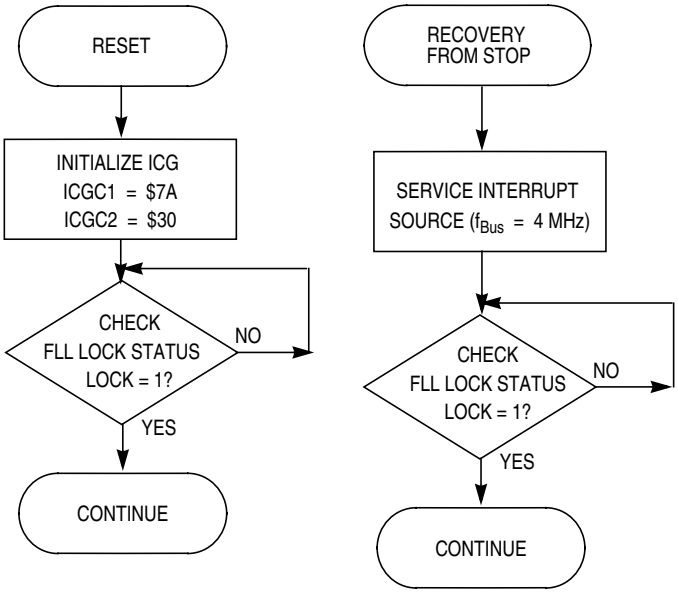


Figure 8-15. ICG Initialization and Stop Recovery for Example #2

ICGTRM = $xx

Bit 7:0 TRIM — Only need to write when trimming internal oscillator; done in separate operation (see example #4)

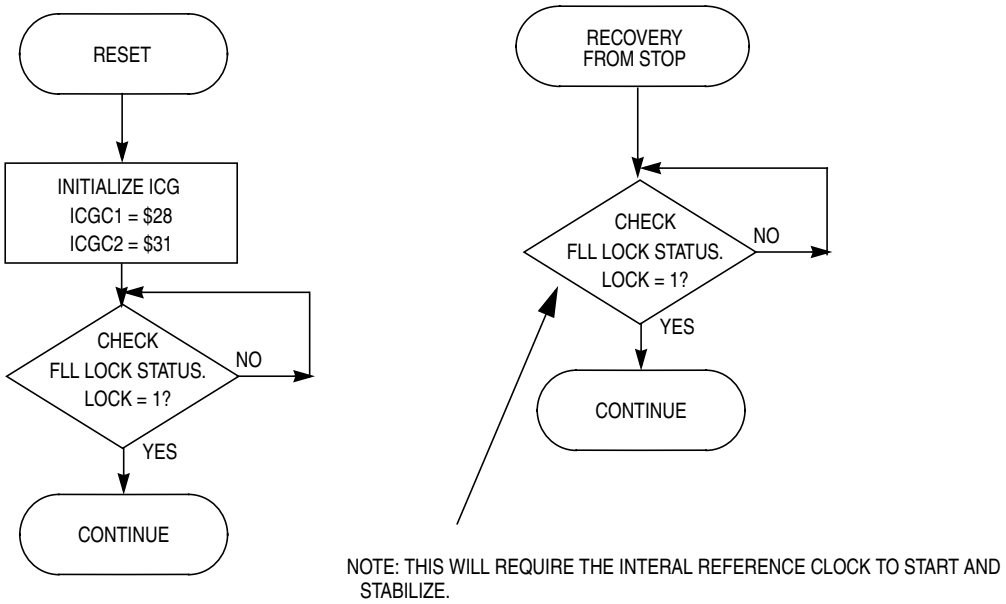


Figure 8-16. ICG Initialization and Stop Recovery for Example #3

10.5.3 Center-Aligned PWM Mode

This type of PWM output uses the up-/down-counting mode of the timer counter (CPWMS = 1). The output compare value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

pulse width = 2 x (TPMxCnVH:TPMxCnVL) ***Eqn. 10-1***

period = 2 x (TPMxMODH:TPMxMODL);
for TPMxMODH:TPMxMODL = 0x0001–0x7FFF ***Eqn. 10-2***

If the channel value register TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the (nonzero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if generation of 100% duty cycle is not necessary). This is not a significant limitation because the resulting period is much longer than required for normal applications.

TPMxMODH:TPMxMODL = 0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS = 0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS = 1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

Figure 10-12 shows the output compare value in the TPM channel registers (multiplied by 2), which determines the pulse width (duty cycle) of the CPWM signal. If ELSnA = 0, the compare match while counting up forces the CPWM output signal low and a compare match while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.

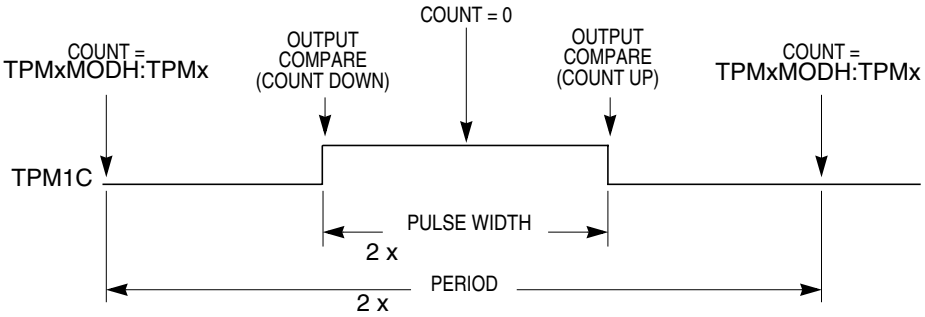
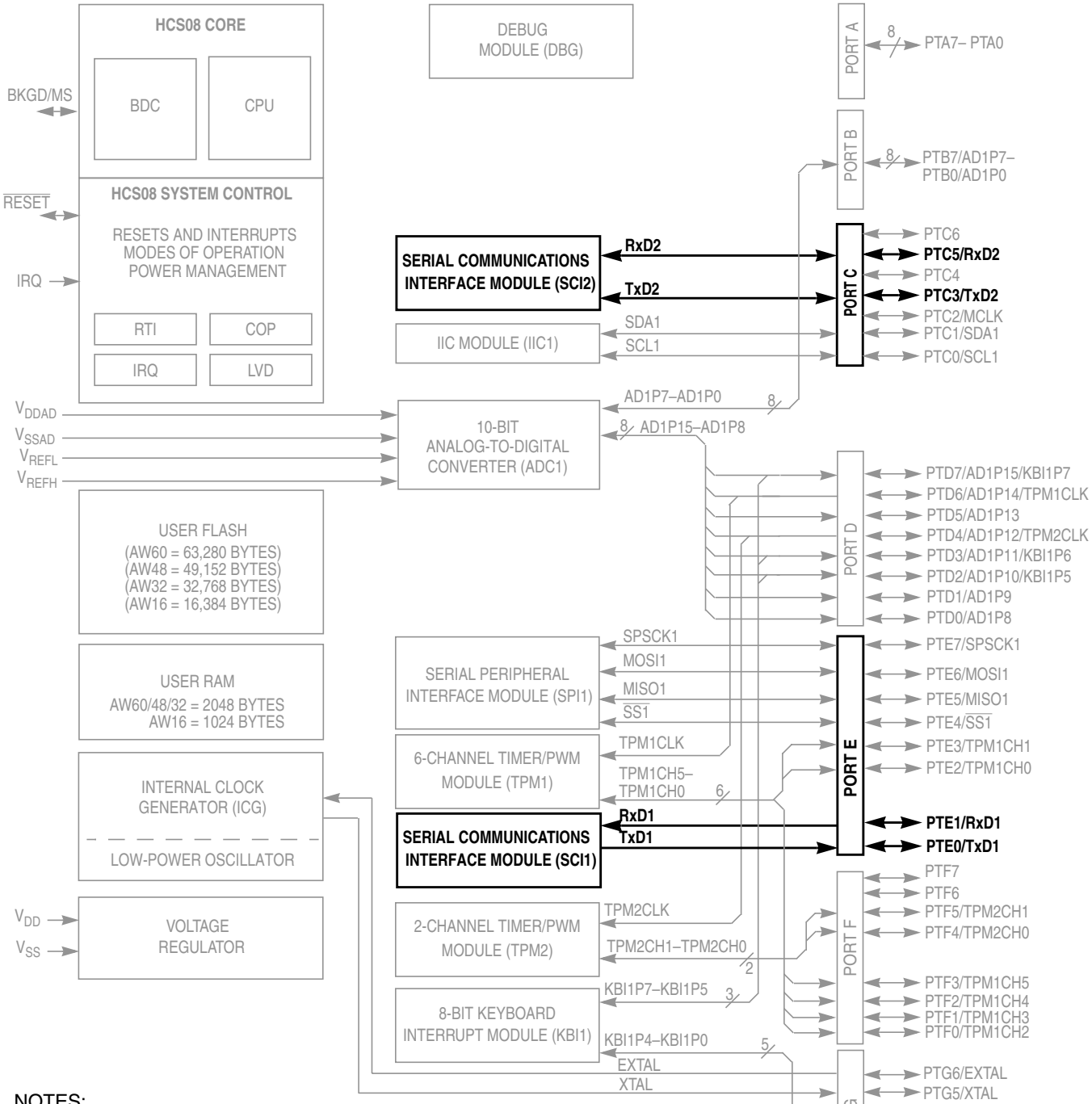


Figure 10-12. CPWM Period and Pulse Width (ELSnA = 0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers, TPMxMODH, TPMxMODL, TPMxCnVH, and TPMxCnVL, actually write to buffer registers. Values are



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1). Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ should not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. Pins PTD7, PTD3, PTD2, and PTG4 contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

Figure 11-1. Block Diagram Highlighting the SCI Modules

Table 13-2. IIC1A Register Field Descriptions

Field	Description
7:6 MULT	**IIC Multiplier Factor** — The MULT bits define the multiplier factor mul. This factor is used along with the SCL divider to generate the IIC baud rate. The multiplier factor mul as defined by the MULT bits is provided below. 00 mul = 01 01 mul = 02 10 mul = 04 11 Reserved
5:0 ICR	**IIC Clock Rate** — The ICR bits are used to prescale the bus clock for bit rate selection. These bits are used to define the SCL divider and the SDA hold value. The SCL divider multiplied by the value provided by the MULT register (multiplier factor mul) is used to generate IIC baud rate. IIC baud rate = bus speed (Hz)/(mul * SCL divider) SDA hold time is the delay from the falling edge of the SCL (IIC clock) to the changing of SDA (IIC data). The ICR is used to determine the SDA hold value. SDA hold time = bus period (s) * SDA hold value Table 13-3 provides the SCL divider and SDA hold values for corresponding values of the ICR. These values can be used to set IIC baud rate and SDA hold time. For example: Bus speed = 8 MHz MULT is set to 01 (mul = 2) Desired IIC baud rate = 100 kbps IIC baud rate = bus speed (Hz)/(mul * SCL divider) 100000 = 8000000/(2*SCL divider) SCL divider = 40 Table 13-3 shows that ICR must be set to 0B to provide an SCL divider of 40 and that this will result in an SDA hold value of 9. SDA hold time = bus period (s) * SDA hold value SDA hold time = 1/8000000 * 9 = 1.125 µs If the generated SDA hold value is not acceptable, the MULT bits can be used to change the ICR. This will result in a different SDA hold value.

13.4.1.1 START Signal

When the bus is free; i.e., no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 13-8, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

13.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 13-8).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address that is equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC will revert to slave mode and operate correctly even if it is being addressed by another master.

13.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 13-8. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the 9th bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new calling by generating a repeated START signal.

14.3.1 Analog Power (V_{DDAD})

The ADC analog portion uses V_{DDAD} as its power connection. In some packages, V_{DDAD} is connected internally to V_{DD}. If externally available, connect the V_{DDAD} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDAD} for good results.

14.3.2 Analog Ground (V_{SSAD})

The ADC analog portion uses V_{SSAD} as its ground connection. In some packages, V_{SSAD} is connected internally to V_{SS}. If externally available, connect the V_{SSAD} pin to the same voltage potential as V_{SS}.

14.3.3 Voltage Reference High (V_{REFH})

V_{REFH} is the high reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDAD}. If externally available, V_{REFH} may be connected to the same potential as V_{DDAD}, or may be driven by an external source that is between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}).

14.3.4 Voltage Reference Low (V_{REFL})

V_{REFL} is the low reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSAD}. If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSAD}.

14.3.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

14.4 Register Definition

These memory mapped registers control and monitor operation of the ADC:

- Status and control register, ADC1SC1
- Status and control register, ADC1SC2
- Data result registers, ADC1RH and ADC1RL
- Compare value registers, ADC1CVH and ADC1CVL
- Configuration register, ADC1CFG
- Pin enable registers, APCTL1, APCTL2, APCTL3

14.4.1 Status and Control Register 1 (ADC1SC1)

This section describes the function of the ADC status and control register (ADC1SC1). Writing ADC1SC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

Table 14-9. APCTL1 Register Field Descriptions (continued)

Field	Description
1 ADPC1	**ADC Pin Control 1** — ADPC1 is used to control the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	**ADC Pin Control 0** — ADPC0 is used to control the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

14.4.9 Pin Control 2 Register (APCTL2)

APCTL2 is used to control channels 8–15 of the ADC module.

	7	6	5	4	3	2	1	0
R W	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
Reset:	0	0	0	0	0	0	0	0

Figure 14-12. Pin Control 2 Register (APCTL2)

Table 14-10. APCTL2 Register Field Descriptions

Field	Description
7 ADPC15	**ADC Pin Control 15** — ADPC15 is used to control the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	**ADC Pin Control 14** — ADPC14 is used to control the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	**ADC Pin Control 13** — ADPC13 is used to control the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	**ADC Pin Control 12** — ADPC12 is used to control the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	**ADC Pin Control 11** — ADPC11 is used to control the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	**ADC Pin Control 10** — ADPC10 is used to control the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled

15.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

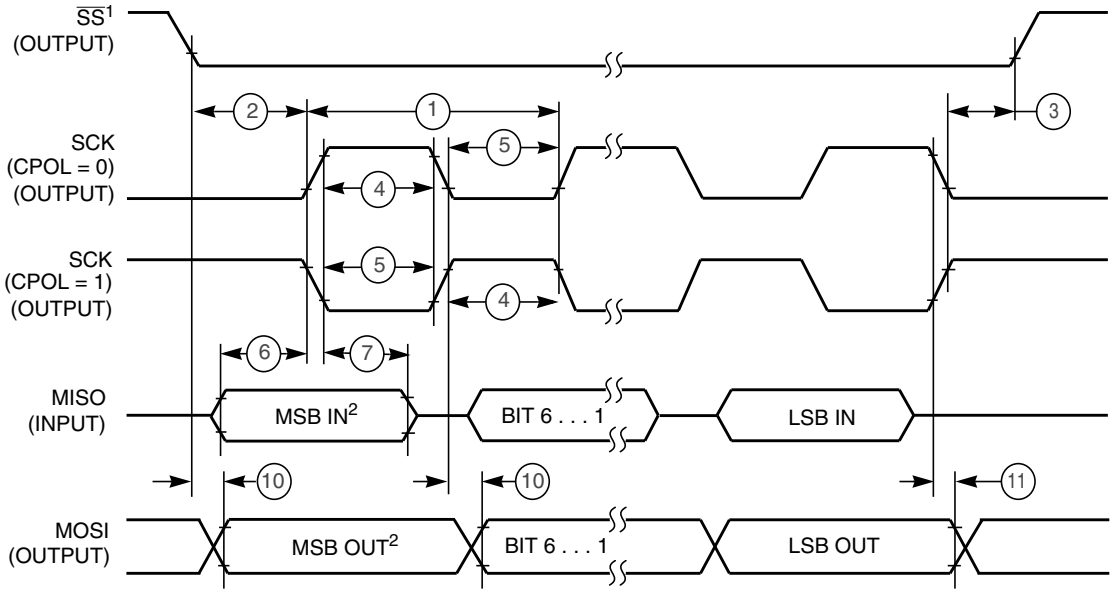
15.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It isn't necessary to read DBGFH in this case.

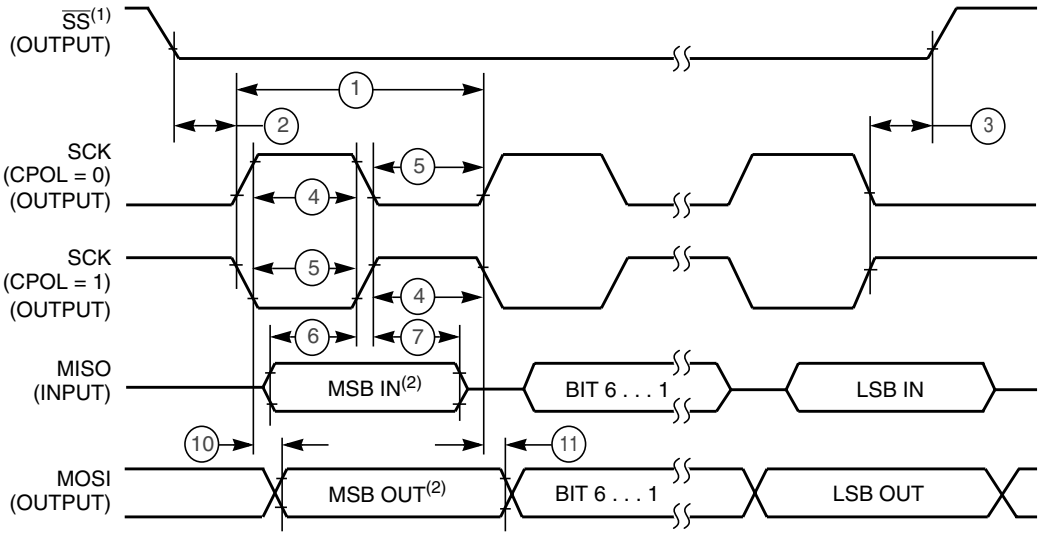
Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.



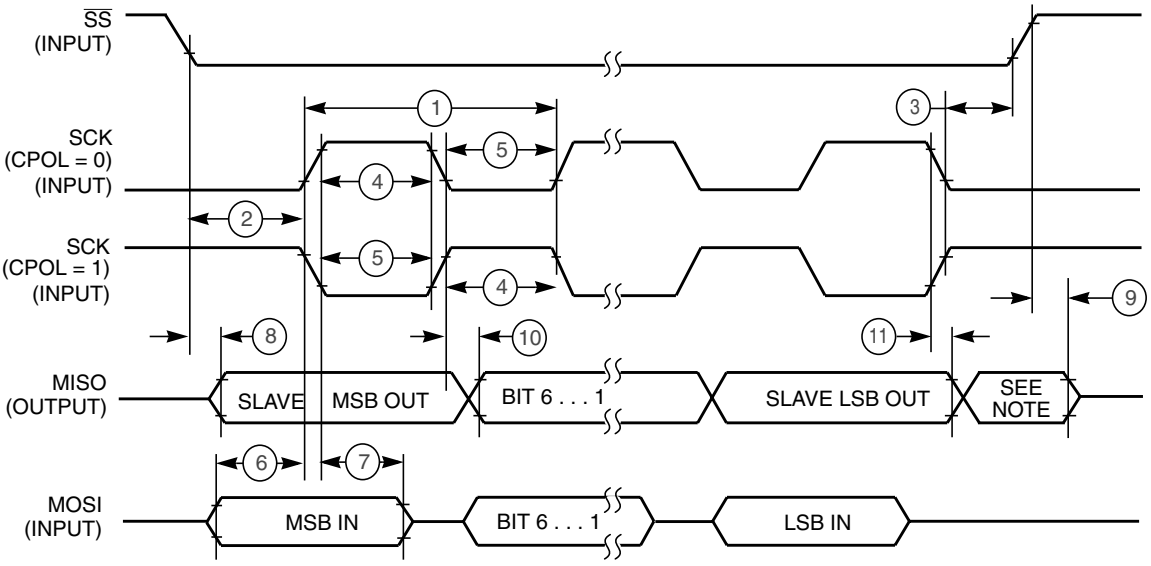
NOTES:
1. $\overline{SS}$ output mode (MODFEN = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

Figure A-16. SPI Master Timing (CPHA = 0)



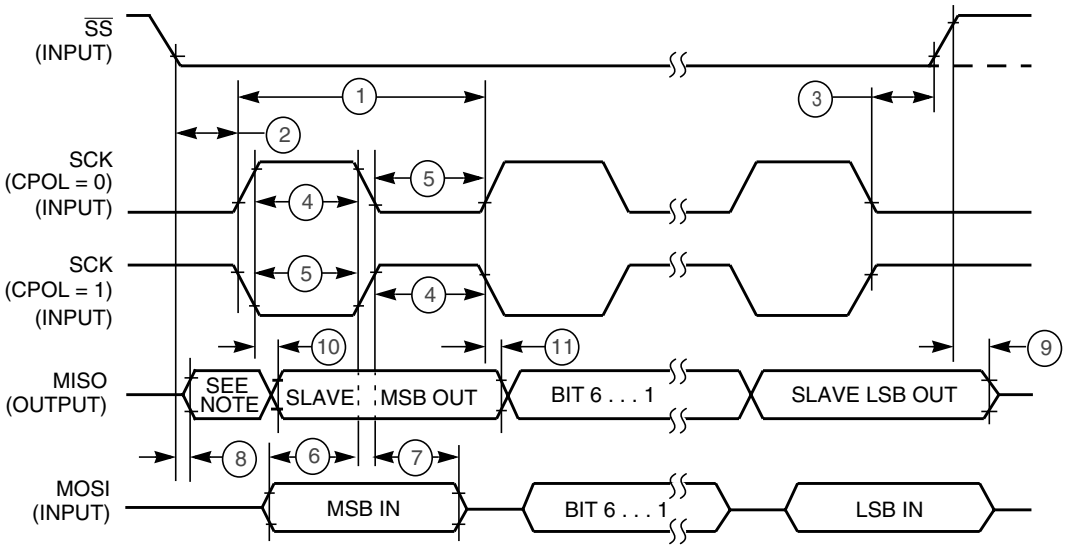
NOTES:
1. $\overline{SS}$ output mode (MODFEN = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

Figure A-17. SPI Master Timing (CPHA = 1)



NOTE:
1. Not defined but normally MSB of character just received

Figure A-18. SPI Slave Timing (CPHA = 0)



NOTE:
1. Not defined but normally LSB of character just received

Figure A-19. SPI Slave Timing (CPHA = 1)