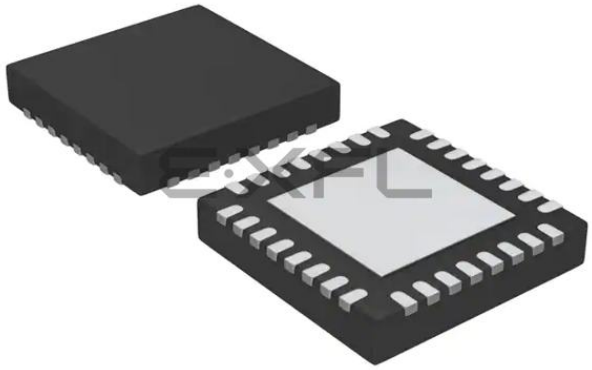


Welcome to E-XFL.COM

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	24
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 4x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-HVQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08gt16acfcer

Part Number	Package Description	Original (gold wire) package document number	Current (copper wire) package document number
MC68HC908JW32	48 QFN	98ARH99048A	98ASA00466D
MC9S08AC16			
MC9S908AC60			
MC9S08AC128			
MC9S08AW60			
MC9S08GB60A			
MC9S08GT16A			
MC9S08JM16			
MC9S08JM60			
MC9S08LL16			
MC9S08QE128			
MC9S08QE32			
MC9S08RG60			
MCF51CN128			
MC9RS08LA8	48 QFN	98ARL10606D	98ASA00466D
MC9S08GT16A	32 QFN	98ARH99035A	98ASA00473D
MC9S908QE32	32 QFN	98ARE10566D	98ASA00473D
MC9S908QE8	32 QFN	98ASA00071D	98ASA00736D
MC9S08JS16	24 QFN	98ARL10608D	98ASA00734D
MC9S08QB8			
MC9S08QG8	24 QFN	98ARL10605D	98ASA00474D
MC9S08SH8	24 QFN	98ARE10714D	98ASA00474D
MC9RS08KB12	24 QFN	98ASA00087D	98ASA00602D
MC9S08QG8	16 QFN	98ARE10614D	98ASA00671D
MC9RS08KB12	8 DFN	98ARL10557D	98ASA00672D
MC9S08QG8			
MC9RS08KA2	6 DFN	98ARL10602D	98ASA00735D

2.3.5 IRQ — External Interrupt Request Pin

IRQ is a dedicated pin with both pullup and pulldown devices built in. This pin has no output capabilities. After a system reset, the IRQ pin is disabled and must be enabled before use. See Section 5.4.2, "IRQ — External Interrupt Request Pin" for more details.

For EMC-sensitive applications, an external RC filter is recommended on the IRQ pin. See Figure 2-5 for an example.

2.3.6 General-Purpose I/O and Peripheral Ports

The remaining 36 pins are shared among general-purpose I/O and on-chip peripheral functions such as timers and serial I/O systems. (Three of these pins are not bonded out on the 44-pin package, five are not bonded out on the 42-pin package, and 15 are not bonded out on the 32-pin package.) Immediately after reset, all 36 of these pins are configured as high-impedance general-purpose inputs with internal pullup devices disabled.

NOTE

To avoid extra current drain from floating input pins, the reset initialization routine in the application program should either enable on-chip pullup devices or change the direction of unused pins to outputs so the pins do not float.

For information about controlling these pins as general-purpose I/O pins, see Chapter 6, "Parallel Input/Output." For information about how and when on-chip peripheral systems use these pins, refer to the appropriate section from Table 2-1.

Table 2-1. Pin Sharing References

Port Pins	Alternate Function	Reference[1]
PTA7–PTA0	KBIP7–KBIP0	Chapter 7, "Keyboard Interrupt (S08KBIV1)"
PTB7–PTB0	ADP7–ADP0	Chapter 14, "Analog-to-Digital Converter (S08ATDV3)"
PTC7–PTC4		
PTC3–PTC2	SCL–SDA	Chapter 13, "Inter-Integrated Circuit (S08IICV1)"
PTC1–PTC0	RxD2–TxD2	Chapter 11, "Serial Communications Interface (S08SCIV1)"
PTD4–PTD3	TPM2CH1–TPM2CH0, TPM2CLK	Chapter 10, "Timer/PWM (S08TPMV2)"
PTD2–PTD0	TPM1CH2–TPM1CH0, TPM1CLK	Chapter 10, "Timer/PWM (S08TPMV2)"
PTE5 PTE4 PTE3 PTE2	SPSCK MISO MOSI $\overline{SS}$	Chapter 12, "Serial Peripheral Interface (S08SPIV3)"
PTE1–PTE0	RxD1–TxD1	Chapter 11, "Serial Communications Interface (S08SCIV1)"
PTG3		
PTG2–PTG1	EXTAL–XTAL	Chapter 9, "Internal Clock Generator (S08ICGV4)"
PTG0	BKGD/MS	Chapter 15, "Development Support"

[1] See this section for information about modules that share these pins.

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**	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	**PTAPE**	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x0002	**PTASE**	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x0003	**PTADD**	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x0004	**PTBD**	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0005	**PTBPE**	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x0006	**PTBSE**	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x0007	**PTBDD**	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0008	**PTCD**	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
0x0009	**PTCPE**	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x000A	**PTCSE**	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x000B	**PTCDD**	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x000C	**PTDD**	0	0	0	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
0x000D	**PTDPE**	0	0	0	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x000E	**PTDSE**	0	0	0	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x000F	**PTDDD**	0	0	0	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
0x0010	**PTED**	0	0	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
0x0011	**PTEPE**	0	0	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x0012	**PTESE**	0	0	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
0x0013	**PTEDD**	0	0	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
0x0014	**IRQSC**	0	0	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x0015	Reserved	—	—	—	—	—	—	—	—
0x0016	**KBISC**	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBF	KBACK	KBIE	KBIMOD
0x0017	**KBIPE**	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x0018	**SCI1BDH**	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0019	**SCI1BDL**	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x001A	**SCI1C1**	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x001B	**SCI1C2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x001C	**SCI1S1**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x001D	**SCI1S2**	0	0	0	0	0	0	0	RAF
0x001E	**SCI1C3**	R8	T8	TXDIR	0	ORIE	NEIE	FEIE	PEIE
0x001F	**SCI1D**	Bit 7	6	5	4	3	2	1	Bit 0
0x0020	**SCI2BDH**	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0021	**SCI2BDL**	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0022	**SCI2C1**	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0023	**SCI2C2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0024	**SCI2S1**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x0025	**SCI2S2**	0	0	0	0	0	0	0	RAF
0x0026	**SCI2C3**	R8	T8	TXDIR	0	ORIE	NEIE	FEIE	PEIE
0x0027	**SCI2D**	Bit 7	6	5	4	3	2	1	Bit 0

Table 4-7. FLASH Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV5:DIV0 (Decimal)	f_{FCLK}	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

4.6.2 FLASH Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from FLASH into FOPT. Bits 5 through 2 are not used and always read 0. This register may be read at any time, but writes have no meaning or effect. 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 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 4.5, “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.

If an event occurs in an enabled interrupt source, an associated read-only status flag will become set. The CPU will not respond until and unless the local interrupt enable is set to 1 to enable the interrupt. The I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset, which masks (prevents) all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

When the CPU receives a qualified interrupt request, it completes the current instruction before responding to the interrupt. The interrupt sequence follows the same cycle-by-cycle sequence as the SWI instruction and consists of:

- Saving the CPU registers on the stack
- Setting the I bit in the CCR to mask further interrupts
- Fetching the interrupt vector for the highest-priority interrupt that is currently pending
- Filling the instruction queue with the first three bytes of program information starting from the address fetched from the interrupt vector locations

While the CPU is responding to the interrupt, the I bit is automatically set to avoid the possibility of another interrupt interrupting the ISR itself (this is called nesting of interrupts). Normally, the I bit is restored to 0 when the CCR is restored from the value stacked on entry to the ISR. In rare cases, the I bit may be cleared inside an ISR (after clearing the status flag that generated the interrupt) so that other interrupts can be serviced without waiting for the first service routine to finish. This practice is not recommended for anyone other than the most experienced programmers because it can lead to subtle program errors that are difficult to debug.

The interrupt service routine ends with a return-from-interrupt (RTI) instruction which restores the CCR, A, X, and PC registers to their pre-interrupt values by reading the previously saved information off the stack.

NOTE

For compatibility with the M68HC08, the H register is not automatically saved and restored. It is good programming practice to push H onto the stack at the start of the interrupt service routine (ISR) and restore it just before the RTI that is used to return from the ISR.

When two or more interrupts are pending when the I bit is cleared, the highest priority source is serviced first (see Table 5-1).

5.4.1 Interrupt Stack Frame

Figure 5-1 shows the contents and organization of a stack frame. Before the interrupt, the stack pointer (SP) points at the next available byte location on the stack. The current values of CPU registers are stored on the stack starting with the low-order byte of the program counter (PCL) and ending with the CCR. After stacking, the SP points at the next available location on the stack which is the address that is one less than the address where the CCR was saved. The PC value that is stacked is the address of the instruction in the main program that would have executed next if the interrupt had not occurred.

8.3.6.7 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.

8.4 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.

8.4.1 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 0xFFFE and 0xFFFF and to fill the instruction queue in preparation for execution of the first program instruction.

8.4.2 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 8-2. HCS08 Instruction Set Summary (Sheet 7 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
TAP	Transfer Accumulator to CCR	CCR ← (A)	↕	↕	↕	↕	↕	↕	INH	84		1
TAX	Transfer Accumulator to X (Index Register Low)	X ← (A)	–	–	–	–	–	–	INH	97		1
TPA	Transfer CCR to Accumulator	A ← (CCR)	–	–	–	–	–	–	INH	85		1
TST *opr8a* TSTA TSTX TST *oprx8*,X TST ,X TST *oprx8*,SP	Test for Negative or Zero	(M) – 0x00 (A) – 0x00 (X) – 0x00 (M) – 0x00 (M) – 0x00 (M) – 0x00	0	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	4 1 1 4 3 5
TSX	Transfer SP to Index Reg.	H:X ← (SP) + 0x0001	–	–	–	–	–	–	INH	95		2
TXA	Transfer X (Index Reg. Low) to Accumulator	A ← (X)	–	–	–	–	–	–	INH	9F		1
TXS	Transfer Index Reg. to SP	SP ← (H:X) – 0x0001	–	–	–	–	–	–	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	I bit ← 0; Halt CPU	–	–	0	–	–	–	INH	8F		2+

[1] Bus clock frequency is one-half of the CPU clock frequency.

Table 9-1. ICGC1 Register Field Descriptions (continued)

Field	Description
2 OSCSTEN	**Enable Oscillator in Off Mode** — The OSCSTEN bit controls whether or not the oscillator circuit remains enabled when the ICG enters off mode. This bit has no effect if HGO = 1 and RANGE = 1. 0 Oscillator disabled when ICG is in off mode unless ENABLE is high, CLKS = 10, and REFST = 1. 1 Oscillator enabled when ICG is in off mode, CLKS = 1X and REFST = 1.
1 LOCD	**Loss of Clock Disable** 0 Loss of clock detection enabled. 1 Loss of clock detection disabled.

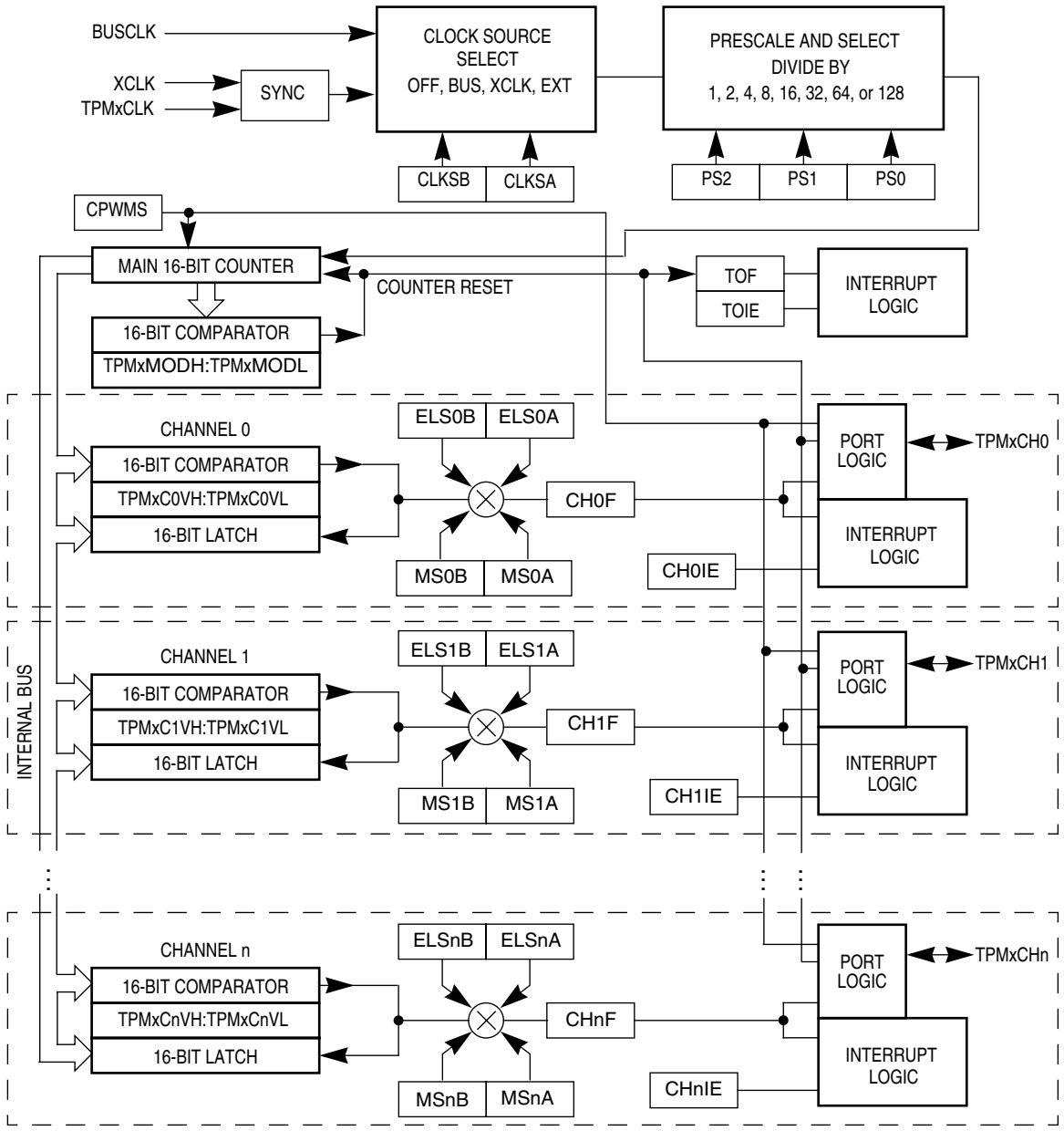


Figure 10-2. TPM Block Diagram

The central component of the TPM is the 16-bit counter that can operate as a free-running counter, a modulo counter, or an up-/down-counter when the TPM is configured for center-aligned PWM. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter. (The values 0x0000 or 0xFFFF effectively make the counter free running.) Software can read the counter value at any time without affecting the counting sequence. Any write to either byte of the TPMxCNT counter resets the counter regardless of the data value written.

When center-aligned PWM operation is specified, the counter counts upward from 0x0000 through its terminal count and then counts downward to 0x0000 where it returns to up-counting. Both 0x0000 and the terminal count value (value in TPMxMODH:TPMxMODL) are normal length counts (one timer clock period long).

An interrupt flag and enable are associated with the main 16-bit counter. The timer overflow flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE = 0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE = 1) where a static hardware interrupt is automatically generated whenever the TOF flag is 1.

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the main 16-bit counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the main 16-bit counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The 0x0000 count value corresponds to the center of a period.)

Because the HCS08 MCU is an 8-bit architecture, a coherency mechanism is built into the timer counter for read operations. Whenever either byte of the counter is read (TPMxCNTH or TPMxCNTL), both bytes are captured into a buffer so when the other byte is read, the value will represent the other byte of the count at the time the first byte was read. The counter continues to count normally, but no new value can be read from either byte until both bytes of the old count have been read.

The main timer counter can be reset manually at any time by writing any value to either byte of the timer count TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only one byte of the counter was read before resetting the count.

10.4.2 Channel Mode Selection

Provided CPWMS = 0 (center-aligned PWM operation is not specified), the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and buffered edge-aligned PWM.

10.4.2.1 Input Capture Mode

With the input capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TPM latches the contents of the TPM counter into the channel value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either byte of the 16-bit capture register is read, both bytes are latched into a buffer to support coherent 16-bit accesses regardless of order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An input capture event sets a flag bit (CHnF) that can optionally generate a CPU interrupt request.

the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCIxD.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD1 pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD1 high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

11.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIxC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD1 pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD1 is an output driving a logic 1. This ensures that the TxD1 line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

11.3.3 Receiver Functional Description

In this section, the data sampling technique used to reconstruct receiver data is described in more detail; two variations of the receiver wakeup function are explained. (The receiver block diagram is shown in Figure 11-3.)

The receiver is enabled by setting the RE bit in SCIxC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 11.3.5.1, “8- and 9-Bit Data Modes.” For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program

Chapter 12
Serial Peripheral Interface (S08SPIV3)

12.1 Introduction

The MC9S08GT16A/GT8A provides one serial peripheral interface (SPI) module. The four pins associated with SPI functionality are shared with port E pins 2–5. See the Appendix A, “Electrical Characteristics,” appendix for SPI electrical parametric information. When the SPI is enabled, the direction of pins is controlled by module configuration. If the SPI is disabled, all four pins can be used as general-purpose I/O.

12.1.2 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

12.1.2.1 SPI System Block Diagram

Figure 12-3 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ($\overline{SS}$ pin). In this system, the master device has configured its $\overline{SS}$ pin as an optional slave select output.

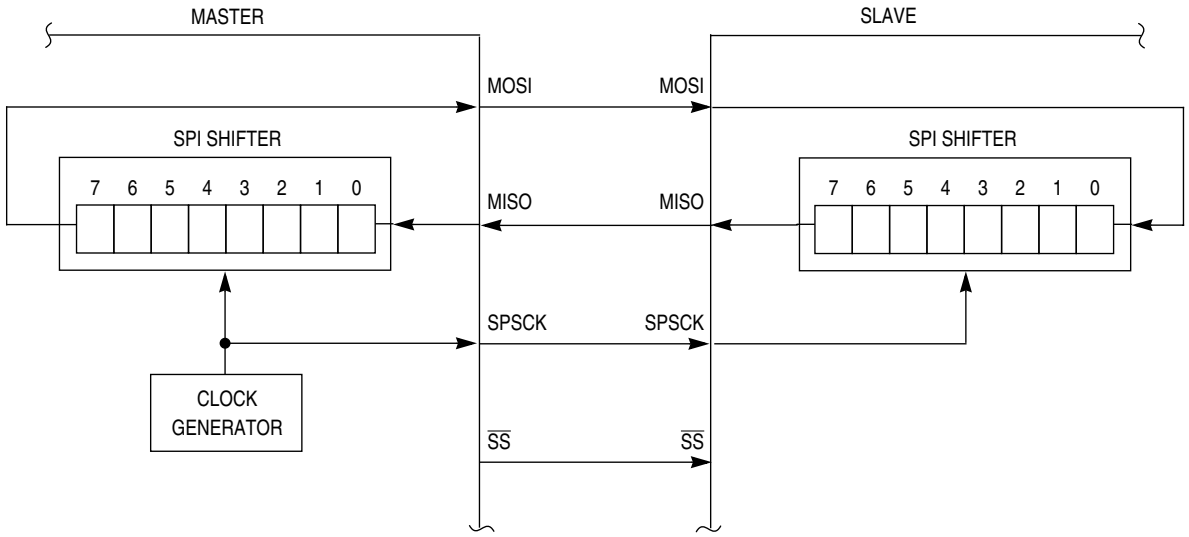


Figure 12-3. SPI System Connections

The most common uses of the SPI system include connecting simple shift registers for adding input or output ports or connecting small peripheral devices such as serial A/D or D/A converters. Although Figure 12-3 shows a system where data is exchanged between two MCUs, many practical systems involve simpler connections where data is unidirectionally transferred from the master MCU to a slave or from a slave to the master MCU.

12.1.2.2 SPI Module Block Diagram

Figure 12-4 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPID) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPID). Pin multiplexing logic controls connections between MCU pins and the SPI module.

Table 12-3. SPIC2 Register Field Descriptions

Field	Description
4 MODFEN	**Master Mode-Fault Function Enable** — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to Table 12-2 for more details). 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	**Bidirectional Mode Output Enable** — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
1 SPISWAI	**SPI Stop in Wait Mode** 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	**SPI Pin Control 0** — The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin. 0 SPI uses separate pins for data input and data output 1 SPI configured for single-wire bidirectional operation

12.4.3 SPI Baud Rate Register (SPIBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

	7	6	5	4	3	2	1	0
R	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 12-8. SPI Baud Rate Register (SPIBR)

Table 12-4. SPIBR Register Field Descriptions

Field	Description
6:4 SPPR[2:0]	**SPI Baud Rate Prescale Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 12-5. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 12-5).
2:0 SPR[2:0]	**SPI Baud Rate Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Table 12-6. The input to this divider comes from the SPI baud rate prescaler (see Figure 12-5). The output of this divider is the SPI bit rate clock for master mode.

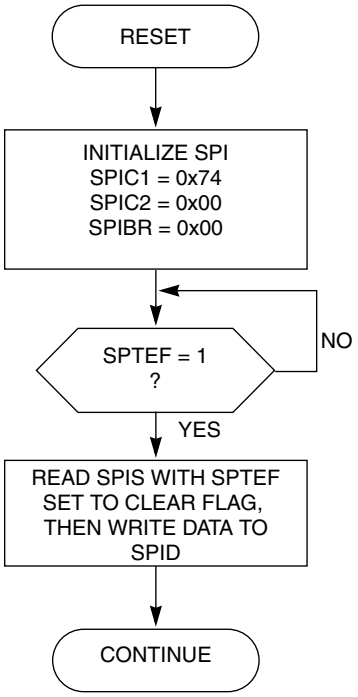


Figure 12-13. Initialization Flowchart Example for SPI Master Device

13.1.1 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus busy detection

13.1.2 Modes of Operation

The IIC functions the same in normal and monitor modes. A brief description of the IIC in the various MCU modes is given here.

- Run mode — This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait mode — The module will continue to operate while the MCU is in wait mode and can provide a wake-up interrupt.
- Stop mode — The IIC is inactive in stop3 mode for reduced power consumption. The STOP instruction does not affect IIC register states. Stop1 and stop2 will reset the register contents.

13.3.3 IIC Control Register (IICC)

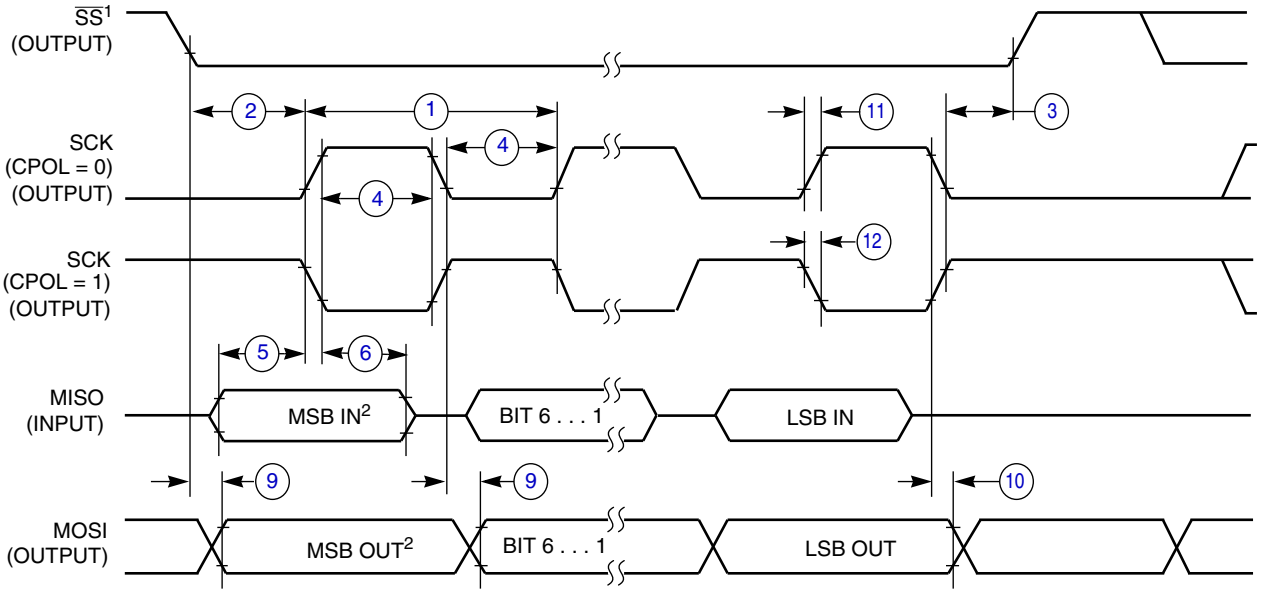
	7	6	5	4	3	2	1	0
R	IICEN	IICIE	MST	TX	TXAK	0	0	0
W						RSTA		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 13-5. IIC Control Register (IICC)

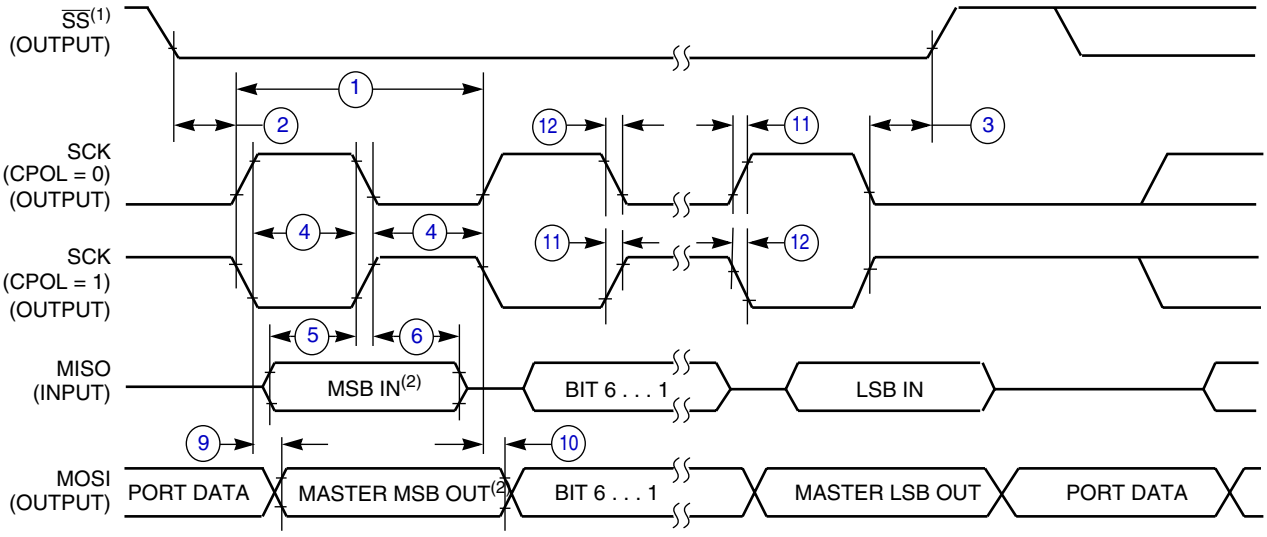
Table 13-4. IICC Register Field Descriptions

Field	Description
7 IICEN	**IIC Enable** — The IICEN bit determines whether the IIC module is enabled. 0 IIC is not enabled. 1 IIC is enabled.
6 IICIE	**IIC Interrupt Enable** — The IICIE bit determines whether an IIC interrupt is requested. 0 IIC interrupt request not enabled. 1 IIC interrupt request enabled.
5 MST	**Master Mode Select** — The MST bit is changed from a 0 to a 1 when a START signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a STOP signal is generated and the mode of operation changes from master to slave. 0 Slave Mode. 1 Master Mode.
4 TX	**Transmit Mode Select** — The TX bit selects the direction of master and slave transfers. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. 0 Receive. 1 Transmit.
3 TXAK	**Transmit Acknowledge Enable** — This bit specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. 0 An acknowledge signal will be sent out to the bus after receiving one data byte. 1 No acknowledge signal response is sent.
2 RSTA	**Repeat START** — Writing a one to this bit will generate a repeated START condition provided it is the current master. This bit will always be read as a low. Attempting a repeat at the wrong time will result in loss of arbitration.



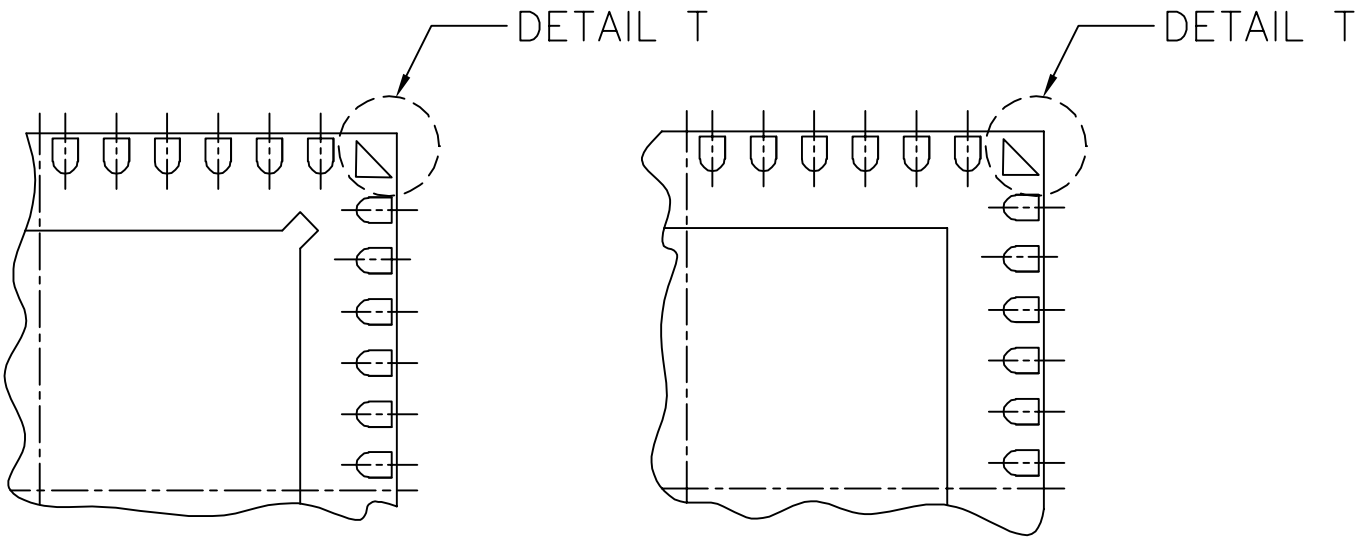
NOTES:
1. $\overline{SS}$ output mode (DDS7 = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

Figure A-17. SPI Master Timing (CPHA = 0)



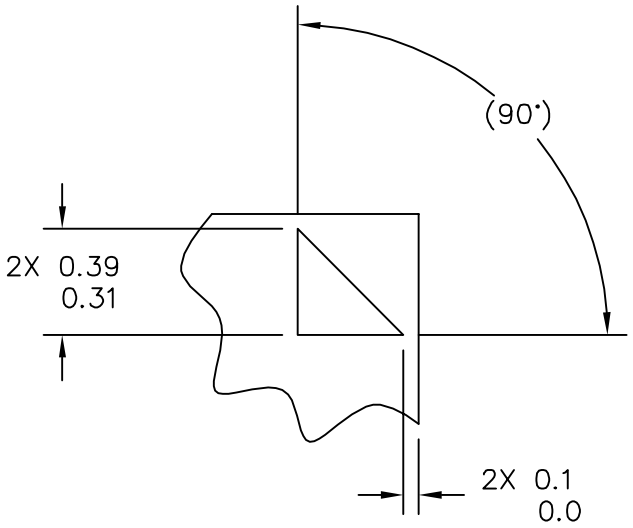
NOTES:
1. $\overline{SS}$ output mode (DDS7 = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

Figure A-18. SPI Master Timing (CPHA = 1)



DETAIL M
PIN 1 BACKSIDE IDENTIFIER OPTION

DETAIL M
PIN 1 BACKSIDE IDENTIFIER OPTION



DETAIL T

MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE: THERMALLY ENHANCED QUAD FLAT NON-LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)	DOCUMENT NO: 98ARH99048A	REV: F
	CASE NUMBER: 1314-05	05 DEC 2005
	STANDARD: JEDEC-MO-220 VKKD-2	