

Welcome to [E-XFL.COM](http://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	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	-
Peripherals	LVD, POR, PWM
Number of I/O	13
Program Memory Size	8KB (8K 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 4x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	16-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	16-TSSOP
Purchase URL	<a href="https://www.e-xfl.com/pro/item?MUrl=&amp;PartUrl=mc908qy8cdte">https://www.e-xfl.com/pro/item?MUrl=&amp;PartUrl=mc908qy8cdte</a>

# **MC68HC908QB8**

# **MC68HC908QB4**

# **MC68HC908QY8**

## **Data Sheet**

---

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

<http://freescale.com/>

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.  
This product incorporates SuperFlash® technology licensed from SST.

© Freescale Semiconductor, Inc., 2005–2010. All rights reserved.

## Table of Contents

15.3.3.3	Transmission Format When CPHA = 1	160
15.3.3.4	Transmission Initiation Latency	161
15.3.4	Queuing Transmission Data	163
15.3.5	Resetting the SPI	164
15.3.6	Error Conditions	164
15.3.6.1	Overflow Error	164
15.3.6.2	Mode Fault Error	166
15.4	Interrupts	167
15.5	Low-Power Modes	168
15.5.1	Wait Mode	168
15.5.2	Stop Mode	168
15.6	SPI During Break Interrupts	168
15.7	I/O Signals	169
15.7.1	MISO (Master In/Slave Out)	169
15.7.2	MOSI (Master Out/Slave In)	169
15.7.3	SPSCK (Serial Clock)	169
15.7.4	$\overline{SS}$ (Slave Select)	169
15.8	Registers	170
15.8.1	SPI Control Register	171
15.8.2	SPI Status and Control Register	172
15.8.3	SPI Data Register	174

## Chapter 16 Timer Interface Module (TIM)

16.1	Introduction	175
16.2	Features	175
16.3	Functional Description	175
16.3.1	TIM Counter Prescaler	175
16.3.2	Input Capture	176
16.3.3	Output Compare	177
16.3.3.1	Unbuffered Output Compare	178
16.3.3.2	Buffered Output Compare	178
16.3.4	Pulse Width Modulation (PWM)	179
16.3.4.1	Unbuffered PWM Signal Generation	179
16.3.4.2	Buffered PWM Signal Generation	180
16.3.4.3	PWM Initialization	181
16.4	Interrupts	182
16.5	Low-Power Modes	182
16.5.1	Wait Mode	182
16.5.2	Stop Mode	182
16.6	TIM During Break Interrupts	182
16.7	I/O Signals	183
16.7.1	TIM Channel I/O Pins (TCH3:TCH0)	183
16.7.2	TIM Clock Pin (TCLK)	183
16.8	Registers	183
16.8.1	TIM Status and Control Register	183

### 2.6.3 FLASH Mass Erase Operation

Use the following procedure to erase the entire FLASH memory to read as a 1:

1. Set both the ERASE bit and the MASS bit in the FLASH control register.
2. Read the FLASH block protect register.
3. Write any data to any FLASH address<sup>(1)</sup> within the FLASH memory address range.
4. Wait for a time,  $t_{NVS}$ .
5. Set the HVEN bit.
6. Wait for a time,  $t_{MErase}$ .
7. Clear the ERASE and MASS bits.

**NOTE**

*Mass erase is disabled whenever any block is protected (FLBPR does not equal \$FF).*

8. Wait for a time,  $t_{NVHL}$ .
9. Clear the HVEN bit.
10. After time,  $t_{RCV}$ , the memory can be accessed in read mode again.

**NOTE**

*Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order as shown, other unrelated operations may occur between the steps.*

**CAUTION**

*A mass erase will erase the internal oscillator trim value at \$FFC0.*

### 2.6.4 FLASH Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 32 consecutive bytes starting from addresses \$XX00, \$XX20, \$XX40, \$XX60, \$XX80, \$XXA0, \$XXC0, or \$XXE0. Use the following step-by-step procedure to program a row of FLASH memory

Figure 2-4 shows a flowchart of the programming algorithm.

**NOTE**

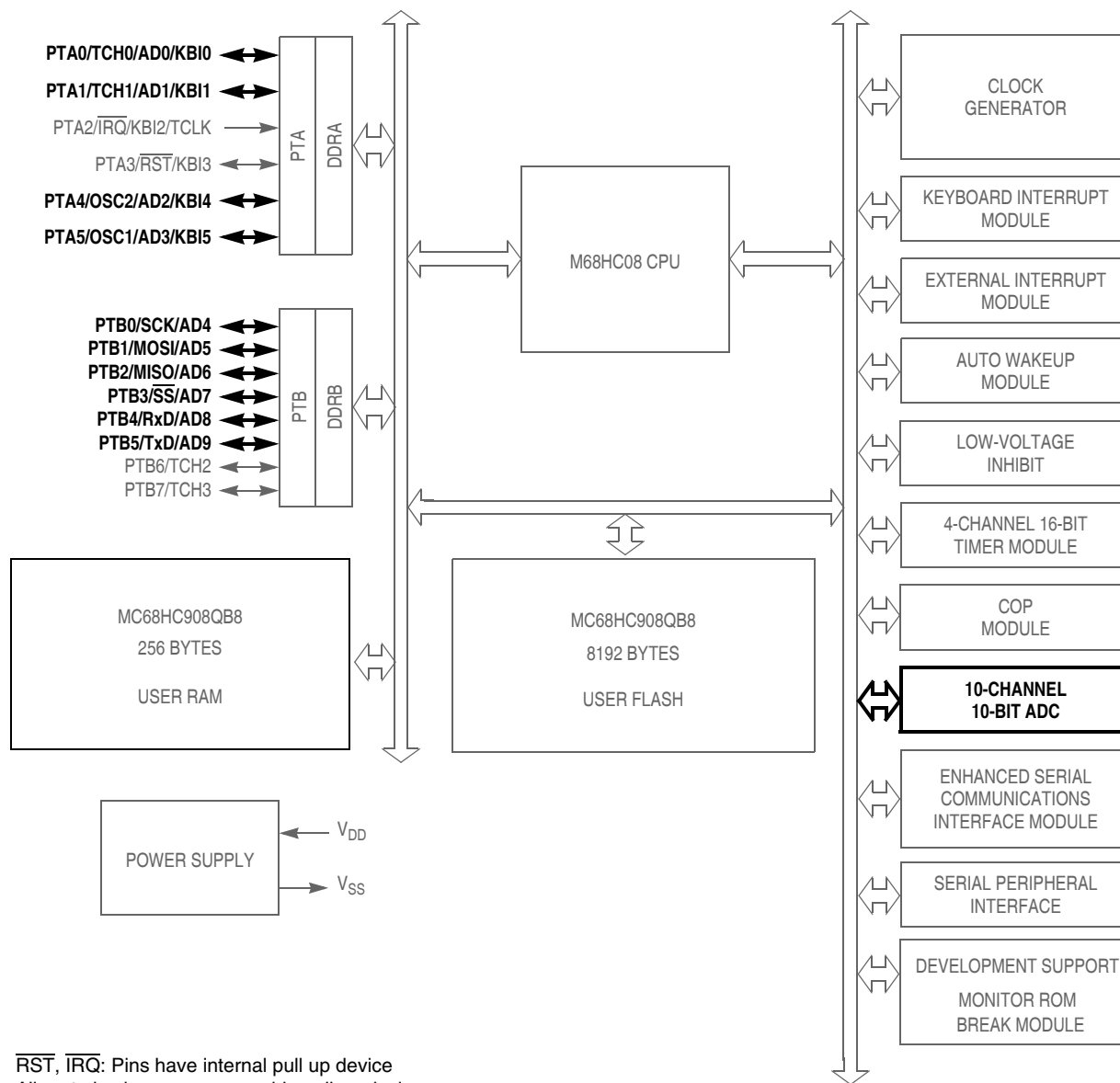
*Do not program any byte in the FLASH more than once after a successful erase operation. Reprogramming bits to a byte which is already programmed is not allowed without first erasing the page in which the byte resides or mass erasing the entire FLASH memory. Programming without first erasing may disturb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

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Read the FLASH block protect register.
3. Write any data to any FLASH location within the address range desired.
4. Wait for a time,  $t_{NVS}$ .
5. Set the HVEN bit.

---

1. When in monitor mode, with security sequence failed (see [17.3.2 Security](#)), write to the FLASH block protect register instead of any FLASH address.

### Analog-to-Digital Converter (ADC10) Module



$\overline{RST}$ ,  $\overline{IRQ}$ : Pins have internal pull up device  
 All port pins have programmable pull up device  
 PTA[0:5]: Higher current sink and source capability

**Figure 3-1. Block Diagram Highlighting ADC10 Block and Pins**

## 4.5 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### 4.5.1 Wait Mode

The AWU module remains inactive in wait mode.

### 4.5.2 Stop Mode

When the AWU module is enabled (AWUIE = 1 in the keyboard interrupt enable register) it is activated automatically upon entering stop mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of stop mode. The AWU counters start from 0 each time stop mode is entered.

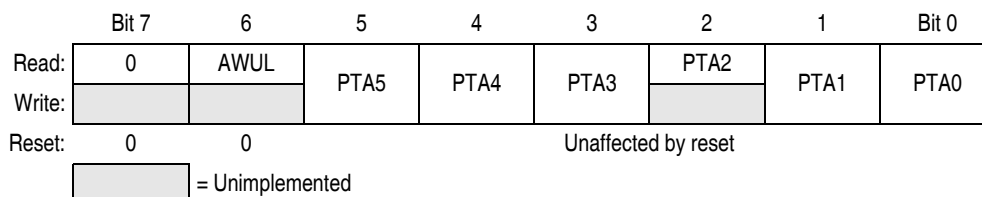
## 4.6 Registers

The AWU shares registers with the keyboard interrupt (KBI) module, the port A I/O module and configuration register 2. The following I/O registers control and monitor operation of the AWU:

- Port A data register (PTA)
- Keyboard interrupt status and control register (KBSCR)
- Keyboard interrupt enable register (KBIER)
- Configuration register 1 (CONFIG1)
- Configuration register 2 (CONFIG2)

### 4.6.1 Port A I/O Register

The port A data register (PTA) contains a data latch for the state of the AWU interrupt request, in addition to the data latches for port A.



**Figure 4-2. Port A Data Register (PTA)**

#### AWUL — Auto Wakeup Latch

This is a read-only bit which has the value of the auto wakeup interrupt request latch. The wakeup request signal is generated internally. There is no PTA6 port or any of the associated bits such as PTA6 data direction or pullup bits.

- 1 = Auto wakeup interrupt request is pending
- 0 = Auto wakeup interrupt request is not pending

**NOTE**


*PTA5–PTA0 bits are not used in conjunction with the auto wakeup feature. To see a description of these bits, see [12.2.1 Port A Data Register](#).*

### 4.6.2 Keyboard Status and Control Register

The keyboard status and control register (KBSCR):

- Flags keyboard/auto wakeup interrupt requests
- Acknowledges keyboard/auto wakeup interrupt requests
- Masks keyboard/auto wakeup interrupt requests

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
Write:						ACKK		
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 4-3. Keyboard Status and Control Register (KBSCR)**

#### Bits 7–4 — Not used

These read-only bits always read as 0s.

#### KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending on port A or auto wakeup. Reset clears the KEYF bit.

- 1 = Keyboard/auto wakeup interrupt pending
- 0 = No keyboard/auto wakeup interrupt pending

#### ACKK — Keyboard Acknowledge Bit

Writing a 1 to this write-only bit clears the keyboard/auto wakeup interrupt request on port A and auto wakeup logic. ACKK always reads as 0. Reset clears ACKK.

#### IMASKK— Keyboard Interrupt Mask Bit

Writing a 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests on port A or auto wakeup. Reset clears the IMASKK bit.

- 1 = Keyboard/auto wakeup interrupt requests masked
- 0 = Keyboard/auto wakeup interrupt requests not masked

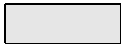
**NOTE**

*MODEK is not used in conjunction with the auto wakeup feature. To see a description of this bit, see [9.8.1 Keyboard Status and Control Register \(KBSCR\)](#).*

### 4.6.3 Keyboard Interrupt Enable Register

The keyboard interrupt enable register (KBIER) enables or disables the auto wakeup to operate as a keyboard/auto wakeup interrupt input.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	AWUIE	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Write:								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 4-4. Keyboard Interrupt Enable Register (KBIER)**

## Auto Wakeup Module (AWU)

**COPRS (In Stop Mode) — Auto Wakeup Period Selection Bit, depends on OSCSTOPEN in CONFIG2 and bus clock source (BUSCLKX2).**

1 = Auto wakeup short cycle =  $512 \times (\text{INTRCOSC or BUSCLKX2})$

0 = Auto wakeup long cycle =  $16,384 \times (\text{INTRCOSC or BUSCLKX2})$

**SSREC — Short Stop Recovery Bit**

SSREC enables the CPU to exit stop mode with a delay of 32 BUSCLKX4 cycles instead of a 4096 BUSCLKX4 cycle delay.

1 = Stop mode recovery after 32 BUSCLKX4 cycles

0 = Stop mode recovery after 4096 BUSCLKX4 cycles

**NOTE**

*LVISTOP, LVIRST, LVIPWRD, LVITRIP and COPD bits are not used in conjunction with the auto wakeup feature. To see a description of these bits, see [Chapter 5 Configuration Register \(CONFIG\)](#)*



### 7.3.5 Condition Code Register

The 8-bit condition code register contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to 1. The following paragraphs describe the functions of the condition code register.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	V	1	1	H	I	N	Z	C
Write:								
Reset:	X	1	1	X	1	X	X	X

X = Indeterminate

**Figure 7-6. Condition Code Register (CCR)**

#### V — Overflow Flag

The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag.

- 1 = Overflow
- 0 = No overflow

#### H — Half-Carry Flag

The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add-without-carry (ADD) or add-with-carry (ADC) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C flags to determine the appropriate correction factor.

- 1 = Carry between bits 3 and 4
- 0 = No carry between bits 3 and 4

#### I — Interrupt Mask

When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the interrupt vector is fetched.

- 1 = Interrupts disabled
- 0 = Interrupts enabled

**NOTE**

*To maintain M6805 Family compatibility, the upper byte of the index register (H) is not stacked automatically. If the interrupt service routine modifies H, then the user must stack and unstack H using the PSHH and PULH instructions.*

After the I bit is cleared, the highest-priority interrupt request is serviced first.

A return-from-interrupt (RTI) instruction pulls the CPU registers from the stack and restores the interrupt mask from the stack. After any reset, the interrupt mask is set and can be cleared only by the clear interrupt mask software instruction (CLI).

#### N — Negative Flag

The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result.

- 1 = Negative result
- 0 = Non-negative result

# Chapter 10

## Low-Voltage Inhibit (LVI)

### 10.1 Introduction

The low-voltage inhibit (LVI) module is provided as a system protection mechanism to prevent the MCU from operating below a certain operating supply voltage level. The module has several configuration options to allow functionality to be tailored to different system level demands.

The configuration registers (see [Chapter 5 Configuration Register \(CONFIG\)](#)) contain control bits for this module.

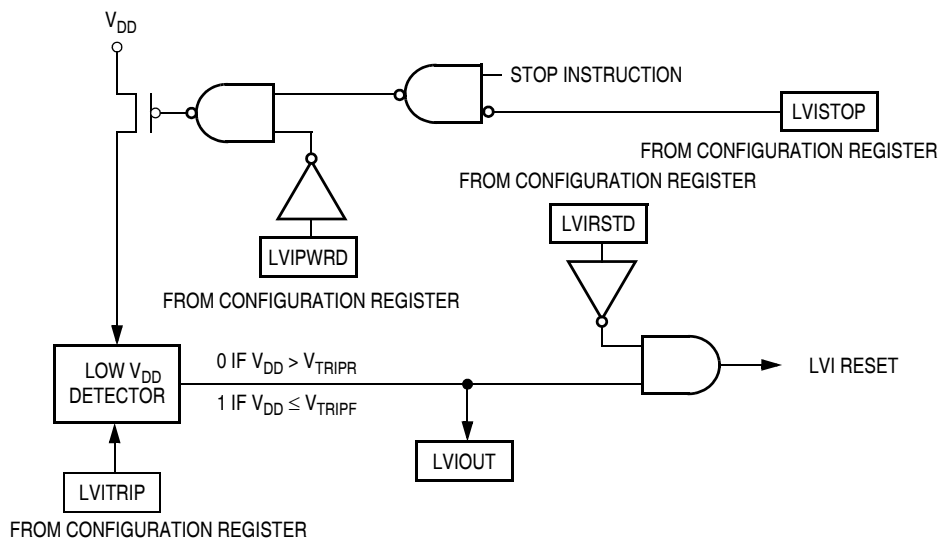
### 10.2 Features

Features of the LVI module include:

- Programmable LVI reset
- Selectable LVI trip voltage
- Programmable stop mode operation

### 10.3 Functional Description

[Figure 10-1](#) shows the structure of the LVI module. LVISTOP, LVIPWRD, LVITRIP, and LVIRSTD are user selectable options found in the configuration register.



**Figure 10-1. LVI Module Block Diagram**

# Chapter 11

## Oscillator Module (OSC)

### 11.1 Introduction

The oscillator (OSC) module is used to provide a stable clock source for the MCU system and bus.

The OSC shares its pins with general-purpose input/output (I/O) port pins. See [Figure 11-1](#) for port location of these shared pins. The OSC2EN bit is located in the port A pull enable register (PTAPUEN) on this MCU. See [Chapter 12 Input/Output Ports \(PORTS\)](#) for information on PTAPUEN register.

### 11.2 Features

The bus clock frequency is one fourth of any of these clock source options:

1. Internal oscillator: An internally generated, fixed frequency clock, trimmable to  $\pm 0.4\%$ . There are three choices for the internal oscillator, 12.8 MHz, 8 MHz, or 4 MHz. The 4-MHz internal oscillator is the default option out of reset.
2. External oscillator: An external clock that can be driven directly into OSC1.
3. External RC: A built-in oscillator module (RC oscillator) that requires an external R connection only. The capacitor is internal to the chip.
4. External crystal: A built-in XTAL oscillator that requires an external crystal or ceramic-resonator. There are three crystal frequency ranges supported, 8–32 MHz, 1–8 MHz, and 32–100 kHz.

### 11.3 Functional Description

The oscillator contains these major subsystems:

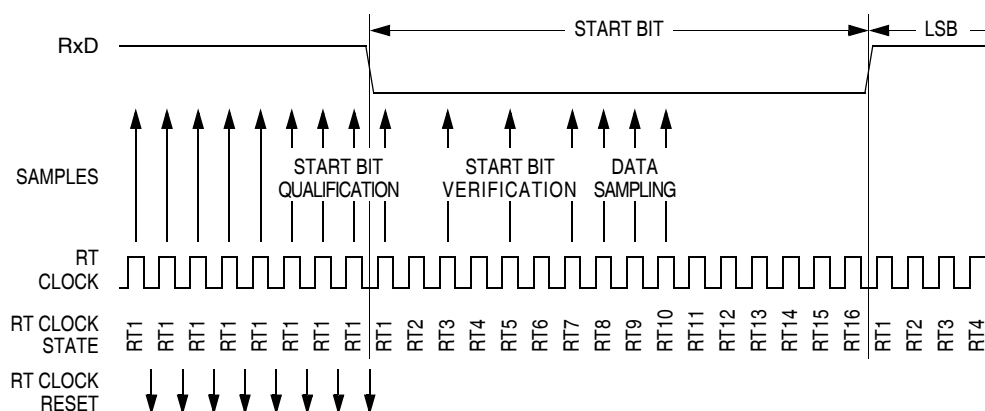
- Internal oscillator circuit
- Internal or external clock switch control
- External clock circuit
- External crystal circuit
- External RC clock circuit

### 13.3.3.3 Data Sampling

The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at these times (see Figure 13-6):

- After every start bit
- After the receiver detects a data bit change from 1 to 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid 0)

To locate the start bit, data recovery logic does an asynchronous search for a 0 preceded by three 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.



**Figure 13-6. Receiver Data Sampling**

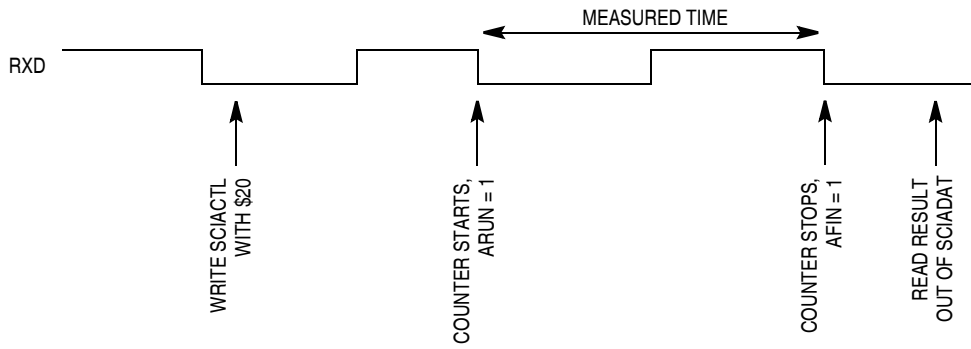
To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 13-1 summarizes the results of the start bit verification samples.

**Table 13-1. Start Bit Verification**

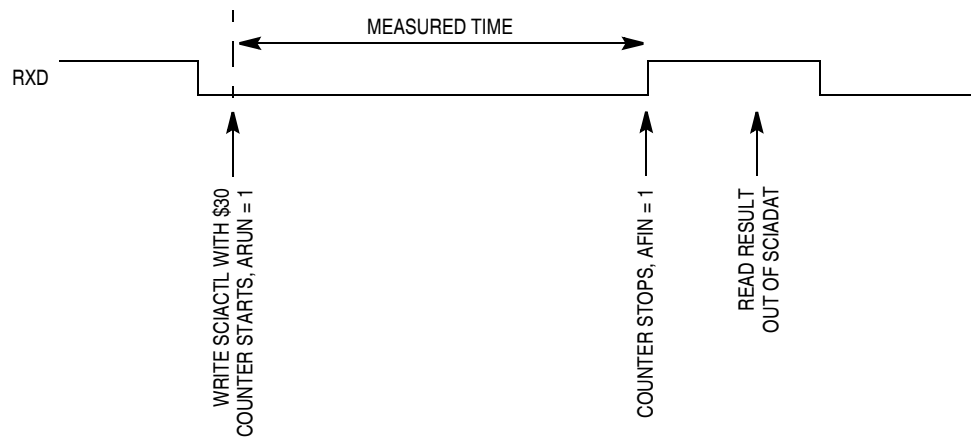
RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1
101	No	0
110	No	0
111	No	0

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

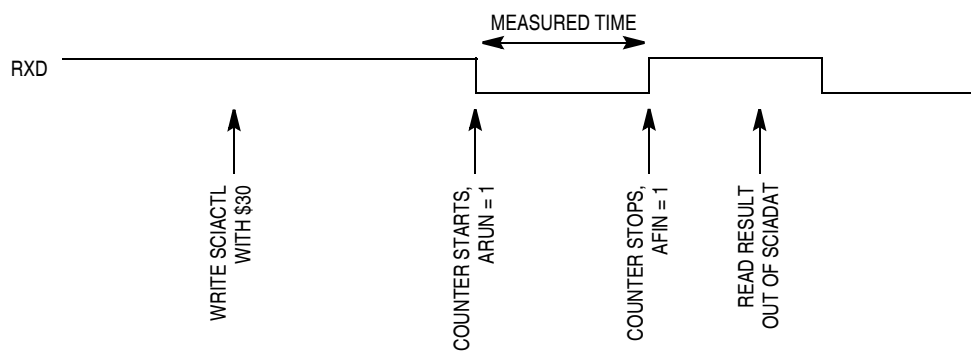
To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 13-2 summarizes the results of the data bit samples.



**Figure 13-20. Bit Time Measurement with ACLK = 0**



**Figure 13-21. Bit Time Measurement with ACLK = 1, Scenario A**



**Figure 13-22. Bit Time Measurement with ACLK = 1, Scenario B**

### 13.9.4 Arbitration Mode

If AM[1:0] is set to 10, the arbiter module operates in arbitration mode. On every rising edge of SCI\_TxD (output of the transmit shift register, see [Figure 13-2](#)), the counter is started. When the counter reaches \$38 (ACLK = 0) or \$08 (ACLK = 1), RxD is statically sensed. If in this case, RxD is sensed low (for example, another bus is driving the bus dominant) ALOST is set. As long as ALOST is set, the TxD pin is forced to 1, resulting in a seized transmission.

If SCI\_TxD senses 0 without having sensed a 0 before on RxD, the counter will be reset, arbitration operation will be restarted after the next rising edge of SCI\_TxD.

## Serial Peripheral Interface (SPI) Module

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error interrupt request.

The mode fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error interrupt requests.

The following sources in the SPI status and control register can generate interrupt requests:

- SPI receiver full bit (SPRF) — SPRF becomes set every time a byte transfers from the shift register to the receive data register. If the SPI receiver interrupt enable bit, SPRIE, is also set, SPRF generates an SPI receiver/error interrupt request.
- SPI transmitter empty bit (SPTE) — SPTE becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE generates an SPTE interrupt request.

## 15.5 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### 15.5.1 Wait Mode

The SPI module remains active after the execution of a WAIT instruction. In wait mode the SPI module registers are not accessible by the CPU. Any enabled interrupt request from the SPI module can bring the MCU out of wait mode.

If SPI module functions are not required during wait mode, reduce power consumption by disabling the SPI module before executing the WAIT instruction.

To exit wait mode when an overflow condition occurs, enable the OVRF bit to generate interrupt requests by setting the error interrupt enable bit (ERRIE). See [15.4 Interrupts](#).

### 15.5.2 Stop Mode

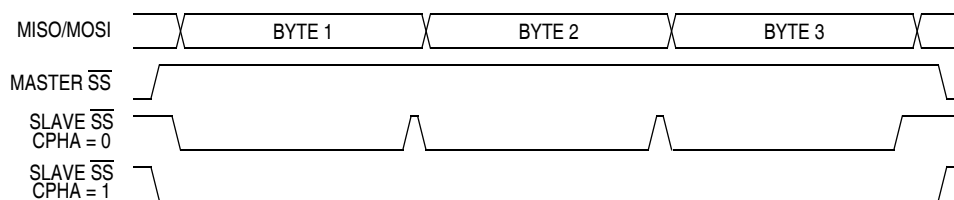
The SPI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions. SPI operation resumes after an external interrupt. If stop mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

## 15.6 SPI During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See BFCR in the SIM section of this data sheet.

To allow software to clear status bits during a break interrupt, write a 1 to BCFE. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a 0 to BCFE. With BCFE cleared (its default state), software can read and write registers during the break state without affecting status bits. Some status bits have a two-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is cleared. After the break, doing the second step clears the status bit.



**Figure 15-12. CPHA/SS Timing**

When an SPI is configured as a slave, the SS pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of SS from creating a MODF error. See [15.8.2 SPI Status and Control Register](#).

**NOTE**

*A high on the SS pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCCK clocks, even if it was already in the middle of a transmission.*

When an SPI is configured as a master, the SS input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCCK. (See [15.3.6.2 Mode Fault Error](#).) For the state of the SS pin to set the MODF flag, the MODFEN bit in the SPSCCK register must be set. If the MODFEN bit is 0 for an SPI master, the SS pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. When MODFEN is 1, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

User software can read the state of the SS pin by configuring the appropriate pin as an input and reading the port data register. See [Table 15-2](#).

**Table 15-2. SPI Configuration**

SPE	SPMSTR	MODFEN	SPI Configuration	Function of SS Pin
0	X <sup>(1)</sup>	X	Not enabled	General-purpose I/O; SS ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; SS ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

1. X = Don't care

## 15.8 Registers

The following registers allow the user to control and monitor SPI operation:

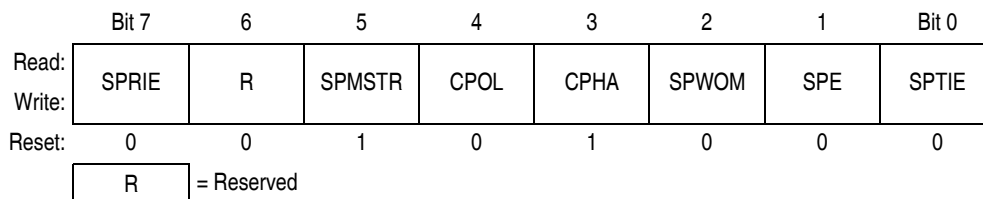
- SPI control register (SPCR)
- SPI status and control register (SPSCR)
- SPI data register (SPDR)



## 15.8.1 SPI Control Register

The SPI control register:

- Enables SPI module interrupt requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase
- Configures the SPSCCK, MOSI, and MISO pins as open-drain outputs
- Enables the SPI module



**Figure 15-13. SPI Control Register (SPCR)**

### SPRIE — SPI Receiver Interrupt Enable Bit

This read/write bit enables interrupt requests generated by the SPRF bit. The SPRF bit is set when a byte transfers from the shift register to the receive data register.

- 1 = SPRF interrupt requests enabled
- 0 = SPRF interrupt requests disabled

### SPMSTR — SPI Master Bit

This read/write bit selects master mode operation or slave mode operation.

- 1 = Master mode
- 0 = Slave mode

### CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCCK pin between transmissions. (See [Figure 15-4](#) and [Figure 15-6](#).) To transmit data between SPI modules, the SPI modules must have identical CPOL values.

### CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See [Figure 15-4](#) and [Figure 15-6](#).) To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the  $\overline{SS}$  pin of the slave SPI module must be high between bytes. (See [Figure 15-12](#).)

### SPWOM — SPI Wired-OR Mode Bit

This read/write bit configures pins SPSCCK, MOSI, and MISO so that these pins become open-drain outputs.

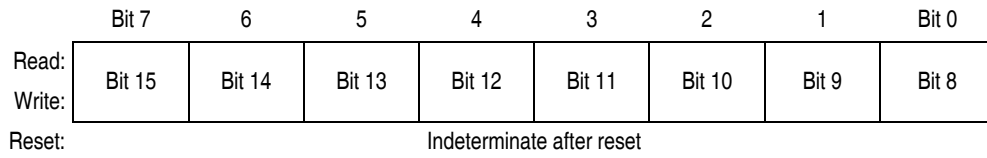
- 1 = Wired-OR SPSCCK, MOSI, and MISO pins
- 0 = Normal push-pull SPSCCK, MOSI, and MISO pins

### SPE — SPI Enable

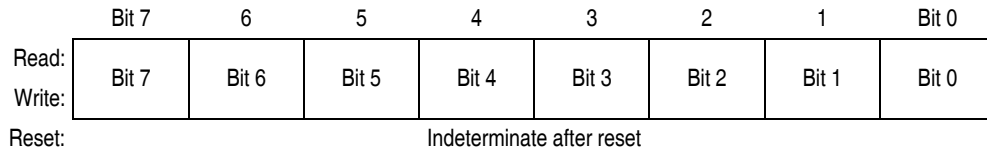
This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See [15.3.5 Resetting the SPI](#).)

- 1 = SPI module enabled
- 0 = SPI module disabled

In output compare mode ( $MSxB:MSxA \neq 0:0$ ), writing to the high byte of the TIM channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.



**Figure 16-14. TIM Channel x Register High (TCHxH)**



**Figure 16-15. TIM Channel x Register Low (TCHxL)**

## 18.16 Timer Interface Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Timer input capture pulse width <sup>(1)</sup>	$t_{TH}, t_{TL}$	2	—	$t_{cyc}$
Timer input capture period	$t_{TLTL}$	Note <sup>(2)</sup>	—	$t_{cyc}$
Timer input clock pulse width <sup>(1)</sup>	$t_{TCL}, t_{TCH}$	$t_{cyc} + 5$	—	ns

1. Values are based on characterization results, not tested in production.
2. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1  $t_{cyc}$ .

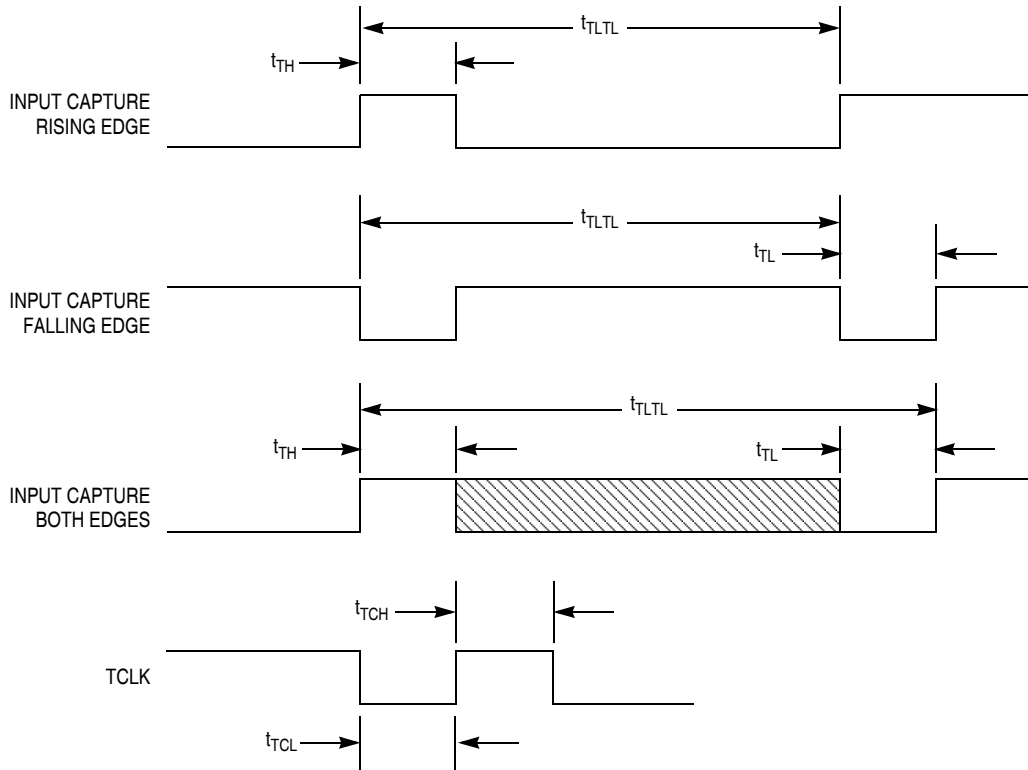


Figure 18-13. Timer Input Timing



## How to Reach Us:

### Home Page:

[www.freescale.com](http://www.freescale.com)

### Web Support:

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

### USA/Europe or Locations Not Listed:

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

### Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[www.freescale.com/support](http://www.freescale.com/support)

### Japan:

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064  
Japan  
0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

### Asia/Pacific:

Freescale Semiconductor China Ltd.  
Exchange Building 23F  
No. 118 Jianguo Road  
Chaoyang District  
Beijing 100022  
China  
+86 10 5879 8000  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

### For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center  
1-800-441-2447 or 303-675-2140  
Fax: 303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto: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. 2005–2010. All rights reserved.