Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	SCI, SPI
Peripherals	LVD, POR, PWM
Number of I/O	36
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Through Hole
Package / Case	56-SDIP (0.600", 15.24mm)
Supplier Device Package	56-PSDIP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908mr16vbe

- Available packages:
  - 64-pin plastic quad flat pack (QFP)
  - 56-pin shrink dual in-line package (SDIP)
- Low-power design, fully static with wait mode
- Master reset pin ($\overline{\text{RST}}$) and power-on reset (POR)
- Stop mode as an option
- Break module (BRK) supports setting the in-circuit simulator (ICS) single break point

Features of the CPU08 include:

- Enhanced M68HC05 programming model
- Extensive loop control functions
- 16 addressing modes (eight more than the M68HC05)
- 16-bit index register and stack pointer
- Memory-to-memory data transfers
- Fast 8 × 8 multiply instruction
- Fast 16/8 divide instruction
- Binary-coded decimal (BCD) instructions
- Optimization for controller applications
- C language support

1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908MR32.

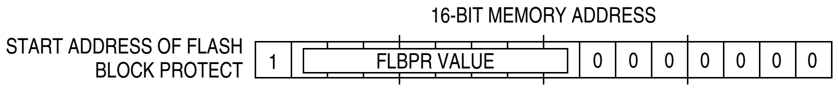


Figure 2-6. FLASH Block Protect Start Address

Refer to Table 2-2 for examples of the protect start address.

Table 2-2. Examples of Protect Start Address

BPR[7:0]	Start of Address of Protect Range
$00	The entire FLASH memory is protected.
$01 (**0000 0001**)	$8080 (1**000 0000 1**000 0000)
$02 (**0000 0010**)	$8100 (1**000 0001 0**000 0000)
and so on...	
$FE (**1111 1110**)	$FF00 (1**111 1111 0**000 0000)
$FF	The entire FLASH memory is not protected.

Note: The end address of the protected range is always $FFFF.

2.8.7 Wait Mode

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly, but there will not be any memory activity since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. Otherwise, the operation will discontinue, and the FLASH will be on standby mode.

2.8.8 Stop Mode

Putting the MCU into stop mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly, but there will not be any memory activity since the CPU is inactive.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH, otherwise the operation will discontinue, and the FLASH will be on standby mode

NOTE

Standby mode is the power-saving mode of the FLASH module in which all internal control signals to the FLASH are inactive and the current consumption of the FLASH is at a minimum.

3.7.1 ADC Status and Control Register

This section describes the function of the ADC status and control register (ADSCR). Writing ADSCR aborts the current conversion and initiates a new conversion.

Address: $0040

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
Write:	R							
Reset:	0	0	0	1	1	1	1	1

R = Reserved

Figure 3-4. ADC Status and Control Register (ADSCR)

COCO — Conversions Complete Bit

In non-interrupt mode (AIEN = 0), COCO is a read-only bit that is set at the end of each conversion. COCO will stay set until cleared by a read of the ADC data register. Reset clears this bit.

In interrupt mode (AIEN = 1), COCO is a read-only bit that is not set at the end of a conversion. It always reads as a 0.

1 = Conversion completed (AIEN = 0)
0 = Conversion not completed (AIEN = 0) or CPU interrupt enabled (AIEN = 1)

NOTE

The write function of the COCO bit is reserved. When writing to the ADSCR register, always have a 0 in the COCO bit position.

AIEN — ADC Interrupt Enable Bit

When this bit is set, an interrupt is generated at the end of an ADC conversion. The interrupt signal is cleared when the data register is read or the status/control register is written. Reset clears the AIEN bit.

1 = ADC interrupt enabled
0 = ADC interrupt disabled

ADCO — ADC Continuous Conversion Bit

When set, the ADC will convert samples continuously and update the ADR register at the end of each conversion. Only one conversion is allowed when this bit is cleared. Reset clears the ADCO bit.

1 = Continuous ADC conversion
0 = One ADC conversion

ADCH[4:0] — ADC Channel Select Bits

ADCH4, ADCH3, ADCH2, ADCH1, and ADCH0 form a 5-bit field which is used to select one of 10 ADC channels. The ADC channels are detailed in Table 3-1.

NOTE

Take care to prevent switching noise from corrupting the analog signal when simultaneously using a port pin as both an analog and digital input.

The ADC subsystem is turned off when the channel select bits are all set to 1. This feature allows for reduced power consumption for the MCU when the ADC is not used.

NOTE

Recovery from the disabled state requires one conversion cycle to stabilize.

4.5.1 PLL Control Register

The PLL control register (PCTL) contains the interrupt enable and flag bits, the on/off switch, and the base clock selector bit.

Address: $005C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
Write:		R			R	R	R	R
Reset:	0	0	1	0	1	1	1	1

R = Reserved

Figure 4-5. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

This read/write bit enables the PLL to generate an interrupt request when the LOCK bit toggles, setting the PLL flag, PLLF. When the AUTO bit in the PLL bandwidth control register (PBWC) is clear, PLLIE cannot be written and reads as logic 0. Reset clears the PLLIE bit.

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL Interrupt Flag

This read-only bit is set whenever the LOCK bit toggles. PLLF generates an interrupt request if the PLLIE bit also is set. PLLF always reads as logic 0 when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. Clear the PLLF bit by reading the PLL control register. Reset clears the PLLF bit.

1 = Change in lock condition
0 = No change in lock condition

NOTE

Do not inadvertently clear the PLLF bit. Any read or read-modify-write operation on the PLL control register clears the PLLF bit.

PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). See 4.3.3 Base Clock Selector Circuit. Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. See 4.3.3 Base Clock Selector Circuit. Reset clears the BCS bit.

1 = CGMVCLK divided by two drives CGMOUT
0 = CGMXCLK divided by two drives CGMOUT

NOTE

PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock *if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. See 4.3.3 Base Clock Selector Circuit.*

PCTL[3:0] — Unimplemented Bits

These bits provide no function and always read as logic 1s.

4.5.2 PLL Bandwidth Control Register

The PLL bandwidth control register (PBWC):

- Selects automatic or manual (software-controlled) bandwidth control mode
- Indicates when the PLL is locked
- In automatic bandwidth control mode, indicates when the PLL is in acquisition or tracking mode
- In manual operation, forces the PLL into acquisition or tracking mode

Address: $005D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	XLD	0	0	0	0
Write:		R			R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 4-6. PLL Bandwidth Control Register (PBWC)

AUTO — Automatic Bandwidth Control Bit

This read/write bit selects automatic or manual bandwidth control. When initializing the PLL for manual operation (AUTO = 0), clear the $\overline{\text{ACQ}}$ bit before turning on the PLL. Reset clears the AUTO bit.

1 = Automatic bandwidth control
0 = Manual bandwidth control

LOCK — Lock Indicator Bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as logic 0 and has no meaning. Reset clears the LOCK bit.

1 = VCO frequency correct or locked
0 = VCO frequency incorrect or unlocked

$\overline{\text{ACQ}}$ — Acquisition Mode Bit

When the AUTO bit is set, $\overline{\text{ACQ}}$ is a read-only bit that indicates whether the PLL is in acquisition mode or tracking mode. When the AUTO bit is clear, $\overline{\text{ACQ}}$ is a read/write bit that controls whether the PLL is in acquisition or tracking mode.

In automatic bandwidth control mode (AUTO = 1), the last-written value from manual operation is stored in a temporary location and is recovered when manual operation resumes. Reset clears this bit, enabling acquisition mode.

1 = Tracking mode
0 = Acquisition mode

6.3.6 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register (CONFIG). See Chapter 5 Configuration Register (CONFIG).

6.4 COP Control Register

The COP control register is located at address $FFFF and overlaps the reset vector. Writing any value to $FFFF clears the COP counter and starts a new timeout period. Reading location $FFFF returns the low byte of the reset vector.

Address: $FFFF

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Low byte of reset vector							
Write:	Clear COP counter							
Reset:	Unaffected by reset							

Figure 6-3. COP Control Register (COPCTL)

6.5 Interrupts

The COP does not generate CPU interrupt requests.

6.6 Monitor Mode

The COP is disabled in monitor mode when V_{HI} is present on the $\overline{IRQ}$ pin or on the $\overline{RST}$ pin.

6.7 Wait Mode

The WAIT instruction puts the MCU in low power-consumption standby mode.

The COP continues to operate during wait mode.

6.8 Stop Mode

Stop mode turns off the CGMXCLK input to the COP and clears the COP prescaler. Service the COP immediately before entering or after exiting stop mode to ensure a full COP timeout period after entering or exiting stop mode.

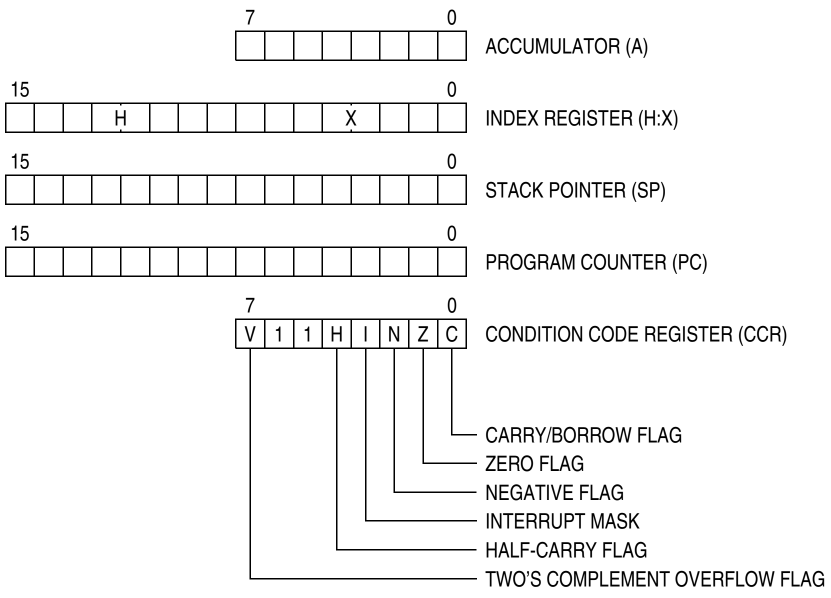


Figure 7-1. CPU Registers

7.3.1 Accumulator

The accumulator is a general-purpose 8-bit register. The CPU uses the accumulator to hold operands and the results of arithmetic/logic operations.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:								
Write:								
Reset:	Unaffected by reset							

Figure 7-2. Accumulator (A)

7.3.2 Index Register

The 16-bit index register allows indexed addressing of a 64-Kbyte memory space. H is the upper byte of the index register, and X is the lower byte. H:X is the concatenated 16-bit index register.

In the indexed addressing modes, the CPU uses the contents of the index register to determine the conditional address of the operand.

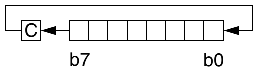
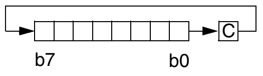
The index register can serve also as a temporary data storage location.

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read:																
Write:																
Reset:	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X

X = Indeterminate

Figure 7-3. Index Register (H:X)

Table 7-1. Instruction Set Summary (Sheet 5 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	INH	88		2
ROL *opr*	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR	39	dd	4
ROLA									INH	49		1
ROLX									INH	59		1
ROL *opr*,X									IX1	69	ff	4
ROL ,X									IX	79		3
ROL *opr*,SP									SP1	9E69	ff	5
ROR *opr*	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR	36	dd	4
RORA									INH	46		1
RORX									INH	56		1
ROR *opr*,X									IX1	66	ff	4
ROR ,X									IX	76		3
ROR *opr*,SP									SP1	9E66	ff	5
RSP	Reset Stack Pointer	SP ← $FF	–	–	–	–	–	–	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕	↕	↕	↕	↕	↕	INH	80		7
RTS	Return from Subroutine	SP ← SP + 1; Pull (PCH) SP ← SP + 1; Pull (PCL)	–	–	–	–	–	–	INH	81		4
SBC #*opr*	Subtract with Carry	A ← (A) – (M) – (C)	↕	–	–	↕	↕	↕	IMM	A2	ii	2
SBC *opr*									DIR	B2	dd	3
SBC *opr*									EXT	C2	hh ll	4
SBC *opr*,X									IX2	D2	ee ff	4
SBC *opr*,X									IX1	E2	ff	3
SBC ,X									IX	F2		2
SBC *opr*,SP									SP1	9EE2	ff	4
SBC *opr*,SP									SP2	9ED2	ee ff	5
SEC	Set Carry Bit	C ← 1	–	–	–	–	–	1	INH	99		1
SEI	Set Interrupt Mask	I ← 1	–	–	1	–	–	–	INH	9B		2
STA *opr*	Store A in M	M ← (A)	0	–	–	↕	↕	–	DIR	B7	dd	3
STA *opr*									EXT	C7	hh ll	4
STA *opr*,X									IX2	D7	ee ff	4
STA *opr*,X									IX1	E7	ff	3
STA ,X									IX	F7		2
STA *opr*,SP									SP1	9EE7	ff	4
STA *opr*,SP									SP2	9ED7	ee ff	5
STHX *opr*	Store H:X in M	(M:M + 1) ← (H:X)	0	–	–	↕	↕	–	DIR	35	dd	4
STOP	Enable Interrupts, Stop Processing, Refer to MCU Documentation	I ← 0; Stop Processing	–	–	0	–	–	–	INH	8E		1
STX *opr*	Store X in M	M ← (X)	0	–	–	↕	↕	–	DIR	BF	dd	3
STX *opr*									EXT	CF	hh ll	4
STX *opr*,X									IX2	DF	ee ff	4
STX *opr*,X									IX1	EF	ff	3
STX ,X									IX	FF		2
STX *opr*,SP									SP1	9EEF	ff	4
STX *opr*,SP									SP2	9EDF	ee ff	5
SUB #*opr*	Subtract	A ← (A) – (M)	↕	–	–	↕	↕	↕	IMM	A0	ii	2
SUB *opr*									DIR	B0	dd	3
SUB *opr*									EXT	C0	hh ll	4
SUB *opr*,X									IX2	D0	ee ff	4
SUB *opr*,X									IX1	E0	ff	3
SUB ,X									IX	F0		2
SUB *opr*,SP									SP1	9EE0	ff	4
SUB *opr*,SP									SP2	9ED0	ee ff	5

When complementary operation is used, two additional features are provided:

- Dead-time insertion
- Separate top/bottom pulse width correction to correct for distortions caused by the motor drive characteristics

If independent operation is chosen, each PWM has its own PWM value register.

12.5.2 Dead-Time Insertion

As shown in Figure 12-13, in complementary mode, each PWM pair can be used to drive top-side/bottom-side transistors.

When controlling dc-to-ac inverters such as this, the top and bottom PWMs in one pair should ***never*** be active at the same time. In Figure 12-13, if PWM1 and PWM2 were on at the same time, large currents would flow through the two transistors as they discharge the bus capacitor. The IGBTs could be weakened or destroyed.

Simply forcing the two PWMs to be inversions of each other is not always sufficient. Since a time delay is associated with turning off the transistors in the motor drive, there must be a dead-time between the deactivation of one PWM and the activation of the other.

A dead-time can be specified in the dead-time write-once register. This 8-bit value specifies the number of CPU clock cycles to use for the dead-time. The dead-time is not affected by changes in the PWM period caused by the prescaler.

Dead-time insertion is achieved by feeding the top PWM outputs of the PWM generator into dead-time generators, as shown in Figure 12-14. Current sensing determines which PWM value of a PWM generator pair to use for the top PWM in the next PWM cycle. See 12.5.3 Top/Bottom Correction with Motor Phase Current Polarity Sensing. When output control is enabled, the odd OUT bits, rather than the PWM generator outputs, are fed into the dead-time generators. See 12.5.5 PWM Output Port Control.

Whenever an input to a dead-time generator transitions, a dead-time is inserted (for example, both PWMs in the pair are forced to their inactive state). The bottom PWM signal is generated from the top PWM and the dead-time. In the case of output control enabled, the odd OUTx bits control the top PWMs, the even OUTx bits control the bottom PWMs *with respect to the odd OUTx bits* (see Table 12-6). Figure 12-15 shows the effects of the dead-time insertion.

As seen in Figure 12-15, some pulse width distortion occurs when the dead-time is inserted. The active pulse widths are reduced. For example, in Figure 12-15, when the PWM value register is equal to two, the ideal waveform (with no dead-time) has pulse widths equal to four. However, the actual pulse widths shrink to two after a dead-time of two was inserted. In this example, with the prescaler set to divide by one and center-aligned operation selected, this distortion can be compensated for by adding or subtracting half the dead-time value to or from the PWM register value. This correction is further described in 12.5.3 Top/Bottom Correction with Motor Phase Current Polarity Sensing.

Further examples of dead-time insertion are shown in Figure 12-16 and Figure 12-17. Figure 12-16 shows the effects of dead-time insertion at the duty cycle boundaries (near 0 percent and 100 percent duty cycles). Figure 12-17 shows the effects of dead-time insertion on pulse widths smaller than the dead-time.

13.3.2.4 Idle Characters

An idle character contains all 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCC1. The preamble is a synchronizing idle character that begins every transmission.

If the TE bit is cleared during a transmission, the PTF5/TxD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the character currently being transmitted.

NOTE

When a break sequence is followed immediately by an idle character, this SCI design exhibits a condition in which the break character length is reduced by one half bit time. In this instance, the break sequence will consist of a valid start bit, eight or nine data bits (as defined by the M bit in SCC1) of logic 0 and one half data bit length of logic 0 in the stop bit position followed immediately by the idle character. To ensure a break character of the proper length is transmitted, always queue up a byte of data to be transmitted while the final break sequence is in progress.

When queueing an idle character, return the TE bit to 1 before the stop bit of the current character shifts out to the PTF5/TxD pin. Setting TE after the stop bit appears on PTF5/TxD causes data previously written to the SCDR to be lost.

A good time to toggle the TE bit is when the SCTE bit becomes set and just before writing the next byte to the SCDR.

13.3.2.5 Inversion of Transmitted Output

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at 1. See 13.7.1 SCI Control Register 1.

13.3.2.6 Transmitter Interrupts

These conditions can generate CPU interrupt requests from the SCI transmitter:

- SCI transmitter empty (SCTE) — The SCTE bit in SCS1 indicates that the SCDR has transferred a character to the transmit shift register. SCTE can generate a transmitter CPU interrupt request. Setting the SCI transmit interrupt enable bit, SCTIE, in SCC2 enables the SCTE bit to generate transmitter CPU interrupt requests.
- Transmission complete (TC) — The TC bit in SCS1 indicates that the transmit shift register and the SCDR are empty and that no break or idle character has been generated. The transmission complete interrupt enable bit, TCIE, in SCC2 enables the TC bit to generate transmitter CPU interrupt requests.

13.3.3 Receiver

Figure 13-6 shows the structure of the SCI receiver.

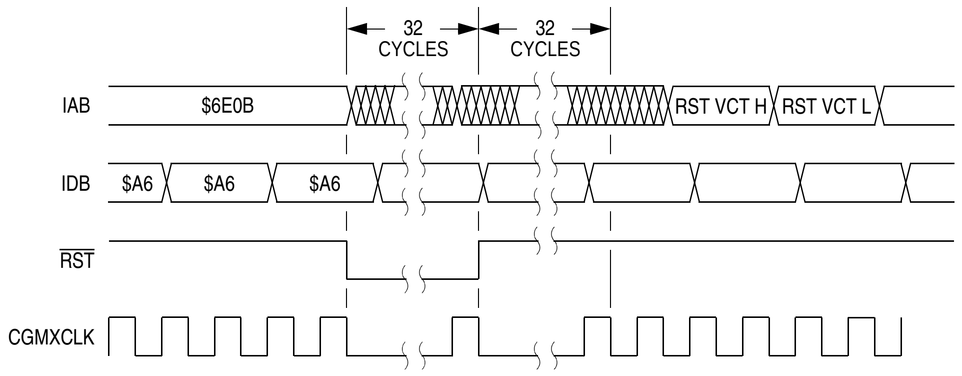


Figure 14-13. Wait Recovery from Internal Reset

14.6.2 Stop Mode

In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

The SIM disables the clock generator module outputs (CGMOUT and CGMXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is hard wired at the normal delay of 4096 CGMXCLK cycles.

It is important to note that when using the PWM generator, its outputs will stop toggling when stop mode is entered. The PWM module must be disabled before entering stop mode to prevent external inverter failure.

14.7 SIM Registers

This subsection describes the SIM registers.

14.7.1 SIM Break Status Register

The SIM break status register (SBSR) contains a flag to indicate that a break caused an exit from wait mode.

Address: $FE00

	BIt 7	6	5	4	3	2	1	Bit 0
Read:	R	R	R	R	R	R	SBSW	R
Write:							Note(1)	
Reset:							0	

R = Reserved

Note 1. Writing a logic 0 clears SBSW.

Figure 14-14. SIM Break Status Register (SBSR)

SBSW — SIM Break Stop/Wait

This status bit is useful in applications requiring a return to wait mode after exiting from a break interrupt. Clear SBSW by writing a logic 0 to it. Reset clears SBSW.

1 = Wait mode was exited by break interrupt.

0 = Wait mode was not exited by break interrupt.

OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes to logic 0. A mode fault in a master SPI causes these events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error CPU interrupt request.
- The SPE bit is cleared.
- The SPTE bit is set.
- The SPI state counter is cleared.
- The data direction register of the shared I/O port regains control of port drivers.

NOTE

To prevent bus contention with another master SPI after a mode fault error, clear all SPI bits of the data direction register of the shared I/O port before enabling the SPI.

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK goes back to its idle level following the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its idle level following the shift of the last data bit. See 15.5 Transmission Formats.

NOTE

Setting the MODF flag does not clear the SPMSTR bit. Reading SPMSTR when MODF = 1 will indicate a mode fault error occurred in either master mode or slave mode.

When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is at logic 0) and later unselected ($\overline{SS}$ is at logic 1) even if no SPSCK is sent to that slave. This happens because $\overline{SS}$ at logic 0 indicates the start of the transmission (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.

In a slave SPI (MSTR = 0), the MODF bit generates an SPI receiver/error CPU interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by clearing the SPE bit of the slave.

NOTE

A logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.

To clear the MODF flag, read the SPSCR with the MODF bit set and then write to the SPCR register. This entire clearing procedure must occur with no MODF condition existing or else the flag is not cleared.

15.8 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low:

- The SPTE flag is set.
- Any transmission currently in progress is aborted.
- The shift register is cleared.
- The SPI state counter is cleared, making it ready for a new complete transmission.
- All the SPI port logic is defaulted back to being general-purpose I/O.

These items are reset only by a system reset:

- All control bits in the SPCR
- All control bits in the SPSCR (MODFEN, ERRIE, SPR1, and SPR0)
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transmissions without having to set all control bits again when SPE is set back high for the next transmission.

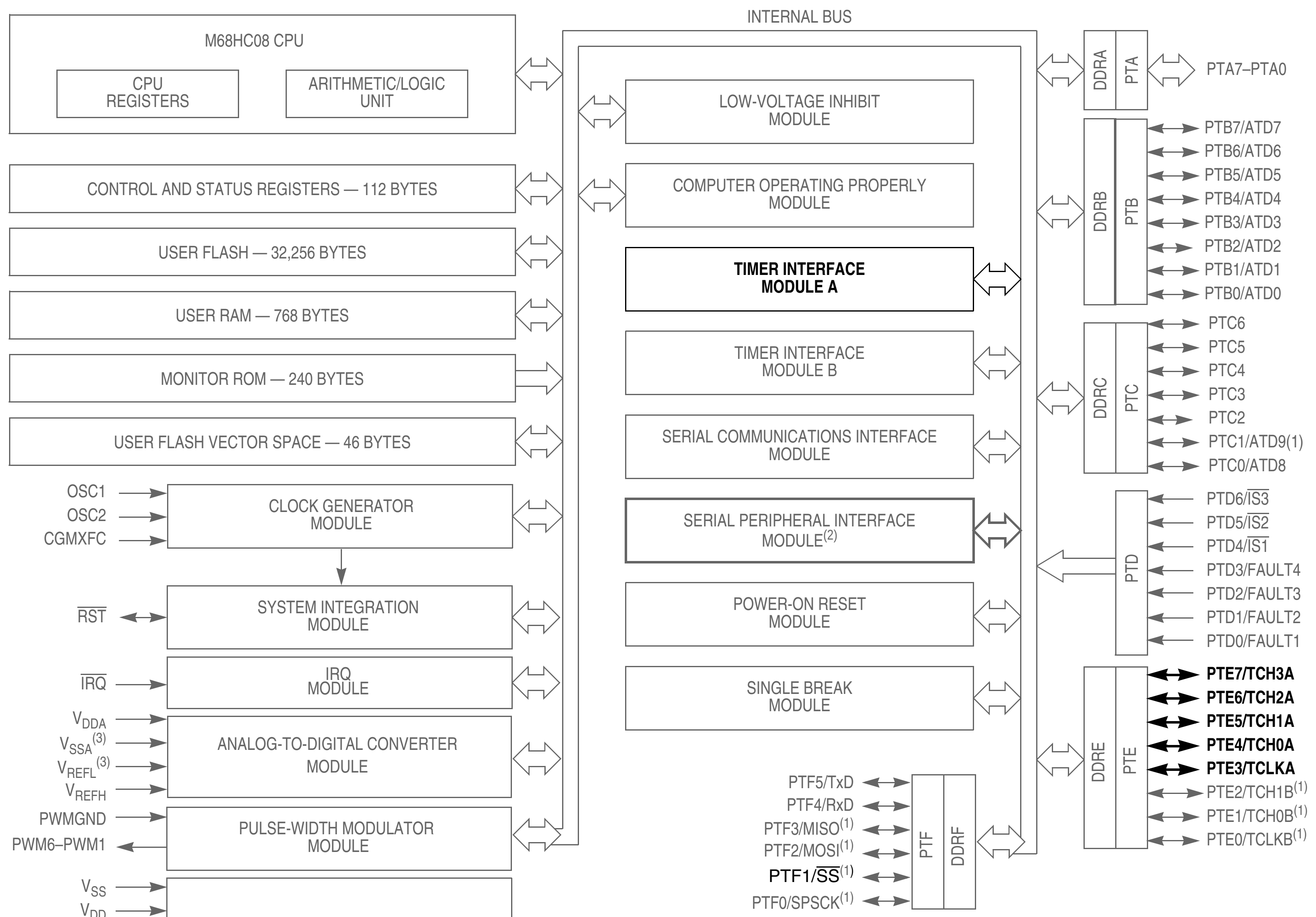
By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI has been disabled. The user can disable the SPI by writing 0 to the SPE bit. The SPI can also be disabled by a mode fault occurring in an SPI that was configured as a master with the MODFEN bit set.

15.9 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the transmit data register only when the SPTE bit is high. Figure 15-12 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA:CPOL = 1:0).

For a slave, the transmit data buffer allows back-to-back transmissions without the slave precisely timing its writes between transmissions as in a system with a single data buffer. Also, if no new data is written to the data buffer, the last value contained in the shift register is the next data word to be transmitted.

For an idle master or idle slave that has no data loaded into its transmit buffer, the SPTE is set again no more than two bus cycles after the transmit buffer empties into the shift register. This allows the user to queue up a 16-bit value to send. For an already active slave, the load of the shift register cannot occur until the transmission is completed. This implies that a back-to-back write to the transmit data register is not possible. The SPTE indicates when the next write can occur.



Notes:

1. These pins are not available in the 56-pin SDIP package.
2. This module is not available in the 56-pin SDIP package.
3. In the 56-pin SDIP package, these pins are bonded together.

Figure 16-1. Block Diagram Highlighting TIMA Block and Pins

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

If TIMA functions are not required during wait mode, reduce power consumption by stopping the TIMA before executing the WAIT instruction.

16.6 I/O Signals

Port E shares five of its pins with the TIMA:

- PTE3/TCLKA is an external clock input to the TIMA prescaler.
- The four TIMA channel I/O pins are PTE4/TCH0A, PTE5/TCH1A, PTE6/TCH2A, and PTE7/TCH3A.

16.6.1 TIMA Clock Pin (PTE3/TCLKA)

PTE3/TCLKA is an external clock input that can be the clock source for the TIMA counter instead of the prescaled internal bus clock. Select the PTE3/TCLKA input by writing logic 1s to the three prescaler select bits, PS[2:0]. See 16.7.1 TIMA Status and Control Register.

The maximum TCLK frequency is the least: 4 MHz or bus frequency ÷ 2.

PTE3/TCLKA is available as a general-purpose I/O pin when not used as the TIMA clock input. When the PTE3/TCLKA pin is the TIMA clock input, it is an input regardless of the state of the DDRE3 bit in data direction register E.

16.6.2 TIMA Channel I/O Pins (PTE4/TCH0A–PTE7/TCH3A)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE2/TCH0 and PTE4/TCH2 can be configured as buffered output compare or buffered PWM pins.

16.7 I/O Registers

These input/output (I/O) registers control and monitor TIMA operation:

- TIMA status and control register (TASC)
- TIMA control registers (TACNTH–TACNTL)
- TIMA counter modulo registers (TAMODH–TAMODL)
- TIMA channel status and control registers (TASC0, TASC1, TASC2, and TASC3)
- TIMA channel registers (TACH0H–TACH0L, TACH1H–TACH1L, TACH2H–TACH2L, and TACH3H–TACH3L)

16.7.1 TIMA Status and Control Register

The TIMA status and control register:

- Enables TIMA overflow interrupts
- Flags TIMA overflows
- Stops the TIMA counter
- Resets the TIMA counter
- Prescales the TIMA counter clock

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0056	TIMB Channel 0 Status/Control Register (TBSC0) See page 247.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0057	TIMB Channel 0 Register High (TBCH0H) See page 250.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0058	TIMB Channel 0 Register Low (TBCH0L) See page 250.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0059	TIMB Channel 1 Status/Control Register (TBSC1) See page 247.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$005A	TIMB Channel 1 Register High (TBCH1H) See page 250.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$005B	TIMB Channel 1 Register Low (TBCH1L) See page 250.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

R = Reserved

Figure 17-3. TIMB I/O Register Summary (Continued)

17.3.1 TIMB Counter Prescaler

The TIMB clock source can be one of the seven prescaler outputs or the TIMB clock pin, PTE0/TCLKB. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMB status and control register select the TIMB clock source.

17.3.2 Input Capture

An input capture function has three basic parts:

1. Edge select logic
2. Input capture latch
3. 16-bit counter

Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TBSC0–TBSC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMB latches the contents of the TIMB counter into the TIMB channel registers, TCHxH–TCHxL. Input captures can generate TIMB CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The free-running counter contents are transferred to the TIMB channel status and control register (TBCHxH–TBCHxL, see 17.7.5 TIMB Channel Registers) on each proper signal transition regardless of

Features include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- 4800 baud–28.8 Kbaud communication with host computer
- Execution of code in random-access memory (RAM) or ROM
- FLASH programming

18.3.1 Functional Description

The monitor ROM receives and executes commands from a host computer. Figure 18-8 shows a sample circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute host-computer code in RAM while all MCU pins retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

18.3.1.1 Entering Monitor Mode

There are two methods for entering monitor:

- The first is the traditional M68HC08 method where $V_{DD} + V_{HI}$ is applied to $\overline{IRQ1}$ and the mode pins are configured appropriately.
- A second method, intended for in-circuit programming applications, will force entry into monitor mode without requiring high voltage on the $\overline{IRQ1}$ pin when the reset vector locations of the FLASH are erased ($FF).

NOTE

For both methods, holding the PTC2 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator. The CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50 percent duty cycle at maximum bus frequency.

Table 18-1 is a summary of the differences between user mode and monitor mode.

Table 18-1. Mode Differences

Modes	Functions						
	COP	**Rest Vector High**	**Reset Vector Low**	**Break Vector High**	**Break Vector Low**	**SWI Vector High**	**SWI Vector Low**
User	Enabled	$FFFE	$FFFF	$FFFC	$FFFD	$FFFC	$FFFD
Monitor	Disabled(1)	$FEFE	$FEFF	$FEFC	$FEFD	$FEFC	$FEFD

1. If the high voltage ($V_{DD} + V_{HI}$) is removed from the $\overline{IRQ1}$ pin or the $\overline{RST}$ pin, the SIM asserts its COP enable output. The COP is a mask option enabled or disabled by the COPD bit in the configuration register.

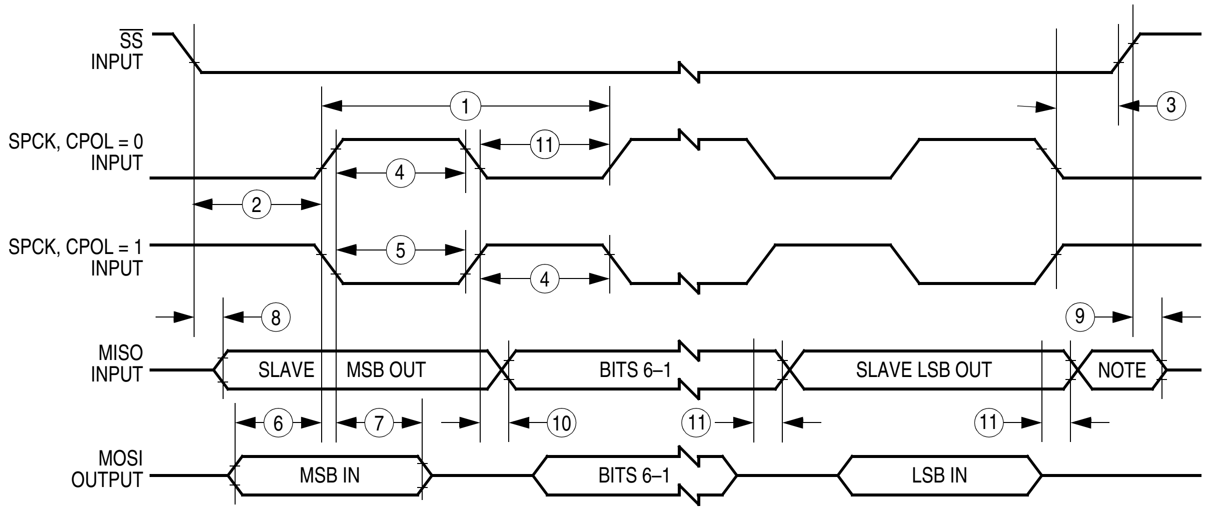
18.3.1.2 Normal Monitor Mode

Table 18-2 shows the pin conditions for entering monitor mode.

19.8 Serial Peripheral Interface Characteristics

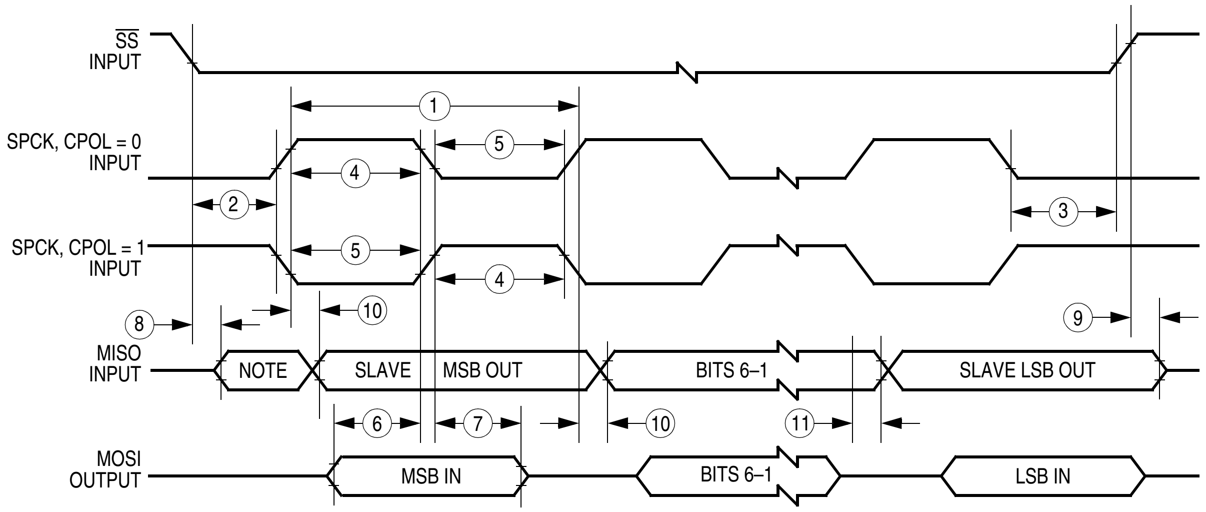
Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	$f_{OP(M)}$ $f_{OP(S)}$	$f_{OP}/128$ dc	$f_{OP}/2$ f_{OP}	MHz
1	Cycle time Master Slave	$t_{CYC(M)}$ $t_{CYC(S)}$	2 1	128 —	t_{CYC}
2	Enable lead time	$t_{Lead(S)}$	15	—	ns
3	Enable lag time	$t_{Lag(S)}$	15	—	ns
4	Clock (SPCK) high time Master Slave	$t_{SCKH(M)}$ $t_{SCKH(S)}$	100 50	— —	ns
5	Clock (SPCK) low time Master Slave	$t_{SCKL(M)}$ $t_{SCKL(S)}$	100 50	— —	ns
6	Data setup time (inputs) Master Slave	$t_{SU(M)}$ $t_{SU(S)}$	45 5	— —	ns
7	Data hold time (inputs) Master Slave	$t_{H(M)}$ $t_{H(S)}$	0 15	— —	ns
8	Access time, slave[3] CPHA = 0 CHPA = 1	$t_{A(CP0)}$ $t_{A(CP1)}$	0 0	40 20	ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	25	ns
10	Data valid time after enable edge Master Slave[5]	$t_{V(M)}$ $t_{V(S)}$	— —	10 40	ns

1. V_{DD} = 5.0 Vdc ± 10%, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted; assumes 100 pF load on all SPI pins
2. Numbers refer to dimensions in Figure 19-1 and Figure 19-2.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins



Note: Not defined, but normally MSB of character just received

a) SPI Slave Timing (CPHA = 0)



Note: Not defined, but normally LSB of character previously transmitted

b) SPI Slave Timing (CPHA = 1)

Figure 19-2. SPI Slave Timing