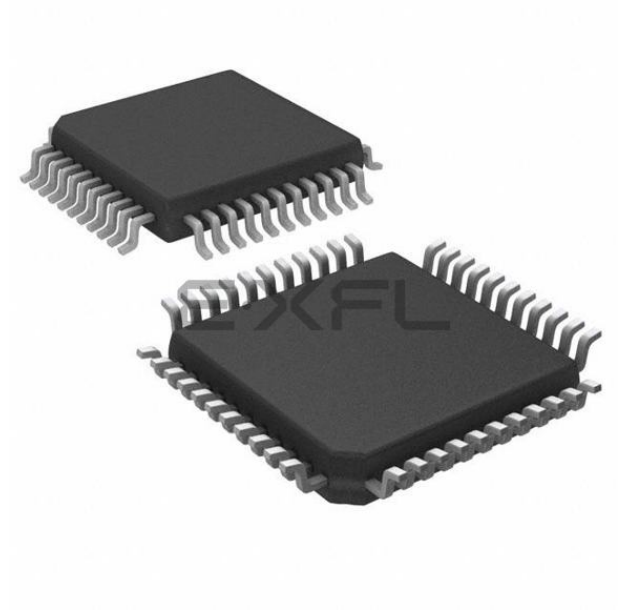


Welcome to E-XFL.COM

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	I²C, IRSCI, SCI, SPI
Peripherals	LED, LVD, POR, PWM
Number of I/O	32
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-QFP
Supplier Device Package	44-QFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc908ap8cfb

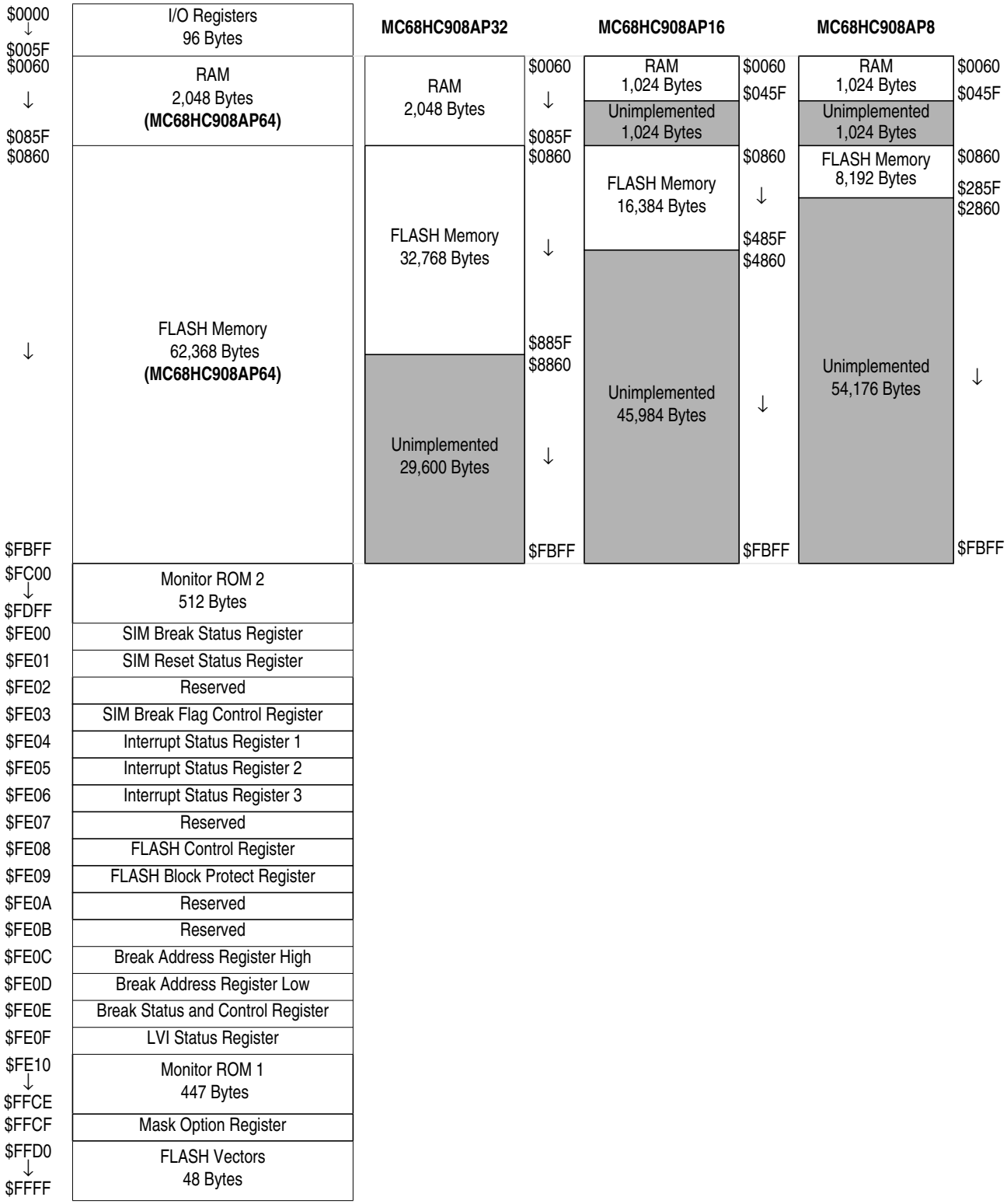


Figure 2-1. Memory Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001A	Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Keyboard Interrupt Enable Register (KBIER)	Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001C	IRQ2 Status and Control Register (INTSCR2)	Read:	0	PUC0ENB	0	0	IRQ2F	0	IMASK2	MODE2
		Write:						ACK2		
		Reset:	0	0	0	0	0	0	0	0
$001D	Configuration Register 2 (CONFIG2)†	Read:	STOP_ ICLKDIS	STOP_ RCLKEN	STOP_ XCLKEN	OSCCLK1	OSCCLK0	0	0	SCIBDSRC
		Write:								
		Reset:	0	0	0	0	0	0	0	0

† One-time writable register after each reset.

$001E	IRQ1 Status and Control Register (INTSCR1)	Read:	0	0	0	0	IRQ1F	0	IMASK1	MODE1
		Write:						ACK1		
		Reset:	0	0	0	0	0	0	0	0
$001F	Configuration Register 1 (CONFIG1)†	Read:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVIREGD	SSREC	STOP	COPD
		Write:								
		Reset:	0	0	0	0	0	0	0	0

† One-time writable register after each reset.

$0020	Timer 1 Status and Control Register (T1SC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0021	Timer 1 Counter Register High (T1CNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0022	Timer 1 Counter Register Low (T1CNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0023	Timer 1 Counter Modulo Register High (T1MODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0024	Timer 1 Counter Modulo Register Low (T1MODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer 1 Channel 0 Status and Control Register (T1SC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0040	IRSCI Control Register 1 (IRSCC1)	Read:	LOOPS	ENSCI	0	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0041	IRSCI Control Register 2 (IRSCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0042	IRSCI Control Register 3 (IRSCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	U	0	0	0	0	0	0
$0043	IRSCI Status Register 1 (IRSCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0044	IRSCI Status Register 2 (IRSCS2)	Read:							BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0045	IRSCI Data Register (IRSCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0046	IRSCI Baud Rate Register (IRSCBR)	Read:	CKS	0	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0047	IRSCI Infrared Control Register (IRSCIRCR)	Read:	R	0	0	0	R	TNP1	TNP0	IREN
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0048	MMIIC Address Register (MMADR)	Read:	MMAD7	MMAD6	MMAD5	MMAD4	MMAD3	MMAD2	MMAD1	MMEXTAD
		Write:								
		Reset:	1	0	1	0	0	0	0	0
$0049	MMIIC Control Register 1 (MMCR1)	Read:	MMEN	MMIEN	0	0	MMTXAK	REPSEN	MMCRCBY TE	0
		Write:			MMCLRBB					
		Reset:	0	0	0	0	0	0	0	0
$004A	MMIIC Control Register 2 (MMCR2)	Read:	MMALIF	MMNAKIF	MMBB	MMAST	MMRW	0	0	MMCRCEF
		Write:	0	0						
		Reset:	0	0	0	0	0	0	0	Unaffected
$004B	MMIIC Status Register (MMSR)	Read:	MMRXIF	MMTXIF	MMATCH	MMSRW	MMRXAK	MMCRCBF	MMTXBE	MMRXBF
		Write:	0	0						
		Reset:	0	0	0	0	1	0	1	0
$004C	MMIIC Data Transmit Register (MMDTR)	Read:	MMTD7	MMTD6	MMTD5	MMTD4	MMTD3	MMTD2	MMTD1	MMTD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 9)

Table 2-1. Vector Addresses (Continued)

Priority	INT Flag	Address	Vector
↓	IF8	$FFEC	TIM2 Channel 1 Vector (High)
		$FFED	TIM2 Channel 1 Vector (Low)
	IF7	$FFEE	TIM2 Channel 0 Vector (High)
		$FFEF	TIM2 Channel 0 Vector (Low)
	IF6	$FFF0	TIM1 Overflow Vector (High)
		$FFF1	TIM1 Overflow Vector (Low)
	IF5	$FFF2	TIM1 Channel 1 Vector (High)
		$FFF3	TIM1 Channel 1 Vector (Low)
	IF4	$FFF4	TIM1 Channel 0 Vector (High)
		$FFF5	TIM1 Channel 0 Vector (Low)
	IF3	$FFF6	PLL Vector (High)
		$FFF7	PLL Vector (Low)
	IF2	$FFF8	$\overline{\text{IRQ2}}$ Vector (High)
		$FFF9	$\overline{\text{IRQ2}}$ Vector (Low)
	IF1	$FFFA	$\overline{\text{IRQ1}}$ Vector (High)
		$FFFB	$\overline{\text{IRQ1}}$ Vector (Low)
	—	$FFFC	SWI Vector (High)
		$FFFD	SWI Vector (Low)
	—	$FFFE	Reset Vector (High)
Highest		$FFFF	Reset Vector (Low)

2.4 Random-Access Memory (RAM)

The following table shows the RAM size and address range:

Device	RAM Size (Bytes)	Memory Address Range
MC68HC908AP64	2,048	$0060–$085F
MC68HC908AP32		
MC68HC908AP16	1,024	$0060–$045F
MC68HC908AP8		

The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64k-byte memory space.

Chapter 6
Clock Generator Module (CGM)

6.1 Introduction

This section describes the clock generator module (CGM). The CGM generates the base clock signal, CGMOUT, which is based on either the oscillator clock divided by two or the divided phase-locked loop (PLL) clock, CGMPCLK, divided by two. CGMOUT is the clock from which the SIM derives the system clocks, including the bus clock, which is at a frequency of CGMOUT 2.

The PLL is a frequency generator designed for use with a low frequency crystal (typically 32.768 kHz) to generate a base frequency and dividing to a maximum bus frequency of 8 MHz.

6.2 Features

Features of the CGM include:

- Phase-locked loop with output frequency in integer multiples of an integer dividend of the crystal reference
- Low-frequency crystal operation with low-power operation and high-output frequency resolution
- Programmable prescaler for power-of-two increases in frequency
- Programmable hardware voltage-controlled oscillator (VCO) for low-jitter operation
- Automatic bandwidth control mode for low-jitter operation
- Automatic frequency lock detector
- CPU interrupt on entry or exit from locked condition
- Configuration register bit to allow oscillator operation during stop mode

6.3 Functional Description

The CGM consists of three major sub-modules:

- Oscillator module — The oscillator module generates the constant reference frequency clock, CGMRCLK (buffered CGMXCLK).
- Phase-locked loop (PLL) — The PLL generates the programmable VCO frequency clock, CGMVCLK, and the divided VCO clock, CGMPCLK.
- Base clock selector circuit — This software-controlled circuit selects either CGMXCLK divided by two or the divided VCO clock, CGMPCLK, divided by two as the base clock, CGMOUT. The SIM derives the system clocks from either CGMOUT or CGMXCLK.

Figure 6-1 shows the structure of the CGM.

Figure 6-2 is a summary of the CGM registers.

VCO clock frequency is corrupt, and appropriate precautions should be taken. If the application is not frequency sensitive, interrupts should be disabled to prevent PLL interrupt service routines from impeding software performance or from exceeding stack limitations.

NOTE

Software can select the CGMPCLK divided by two as the CGMOUT source even if the PLL is not locked (LOCK = 0). Therefore, software should make sure the PLL is locked before setting the BCS bit.

6.7 Special Modes

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

6.7.1 Wait Mode

The WAIT instruction does not affect the CGM. Before entering wait mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL) to save power. Less power-sensitive applications can disengage the PLL without turning it off, so that the PLL clock is immediately available at WAIT exit. This would be the case also when the PLL is to wake the MCU from wait mode, such as when the PLL is first enabled and waiting for LOCK or LOCK is lost.

6.7.2 Stop Mode

The STOP instruction disables the PLL analog circuits and no clock will be driven out of the VCO.

When entering stop mode with the VCO clock (CGMPCLK) selected, before executing the STOP instruction:

1. Set the oscillator stop mode enable bit (STOP_XCLKEN in CONFIG2) if continuos clock is required in stop mode.
2. Clear the BCS bit to select CGMXCLK as CGMOUT.

On exit from stop mode:

1. Set the PLLON bit if cleared before entering stop mode.
2. Wait for PLL to lock by checking the LOCK bit.
3. Set BCS bit to select CGMPCLK as CGMOUT.

6.7.3 CGM 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 SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See 7.7.3 SIM Break Flag Control Register.)

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

To protect the PLLF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write the PLL control register during the break state without affecting the PLLF bit.

If the stop enable bit, STOP, in the mask option register is logic 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

7.3.2.4 Illegal Address Reset

An opcode fetch from an unmapped address generates an illegal address reset. The SIM verifies that the CPU is fetching an opcode prior to asserting the ILAD bit in the SIM reset status register (SRSR) and resetting the MCU. A data fetch from an unmapped address does not generate a reset. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

7.3.2.5 Low-Voltage Inhibit (LVI) Reset

The low-voltage inhibit module (LVI) asserts its output to the SIM when the V_{DD} voltage falls to the LVI_{TRIPF} voltage. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{RST}$) is held low while the SIM counter counts out 4096 + 32 ICLK cycles. Thirty-two ICLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

7.3.2.6 Monitor Mode Entry Module Reset

The monitor mode entry module reset asserts its output to the SIM when monitor mode is entered in the condition where the reset vectors are blank ($FF). (See Chapter 8 Monitor ROM (MON).) When MODRST gets asserted, an internal reset occurs. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

7.4 SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus (IBUS) clocks. The SIM counter also serves as a prescaler for the computer operating properly module (COP). The SIM counter overflow supplies the clock for the COP module. The SIM counter is 13 bits long and is clocked by the falling edge of CGMXCLK.

7.4.1 SIM Counter During Power-On Reset

The power-on reset module (POR) detects power applied to the MCU. At power-on, the POR circuit asserts the signal PORRST. Once the SIM is initialized, it enables the clock generation module (CGM) to drive the bus clock state machine.

7.4.2 SIM Counter During Stop Mode Recovery

The SIM counter also is used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt, break, or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the mask option register. If the SSREC bit is a logic 1, then the stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32 CGMXCLK cycles. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared.

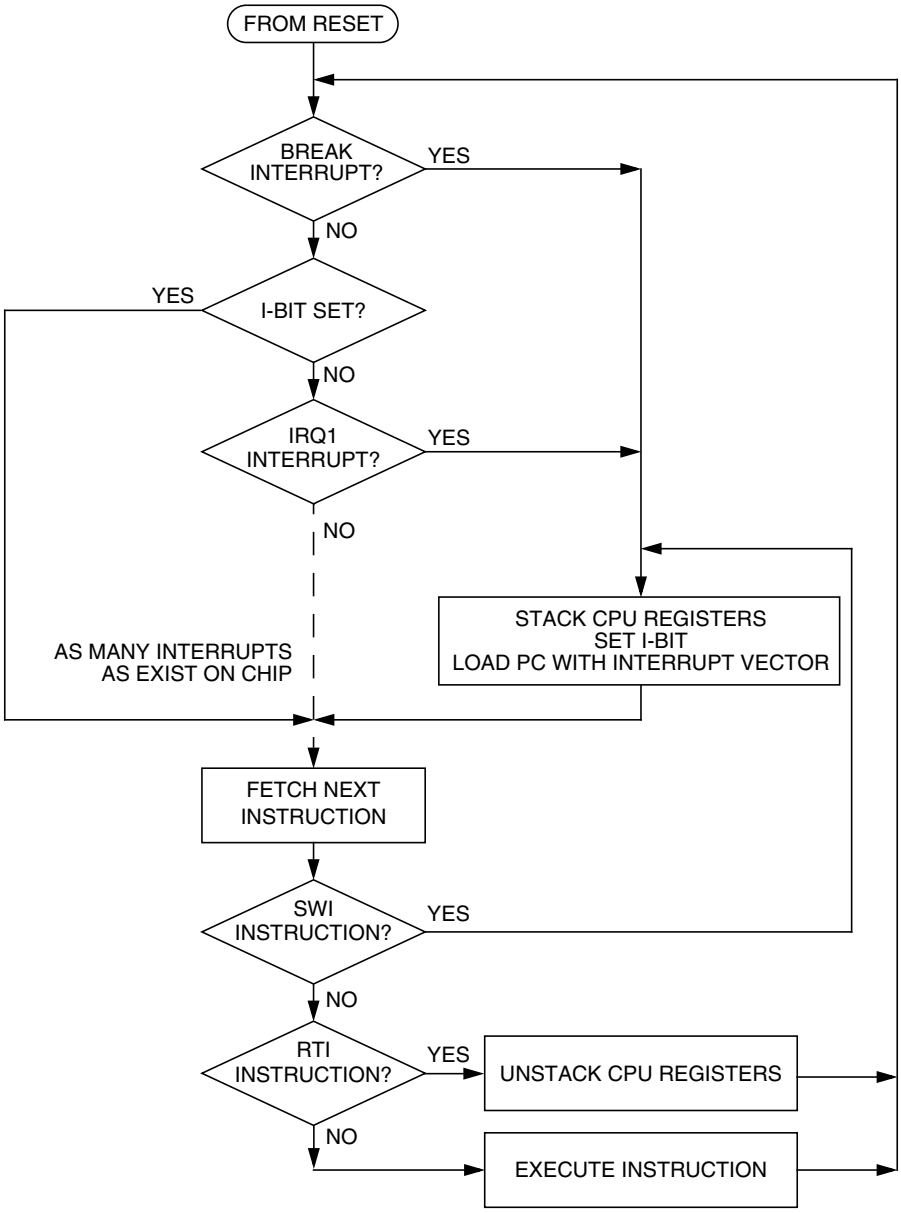


Figure 7-10. Interrupt Processing

7.5.1.1 Hardware Interrupts

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register) and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. Figure 7-11 demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

7.5.2.1 Interrupt Status Register 1

Address: $FE04

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7-12. Interrupt Status Register 1 (INT1)

IF6–IF1 — Interrupt Flags 6–1

These flags indicate the presence of interrupt requests from the sources shown in Table 7-3.

1 = Interrupt request present
0 = No interrupt request present

Bit 0 and Bit 1 — Always read 0

7.5.2.2 Interrupt Status Register 2

Address: $FE05

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7-13. Interrupt Status Register 2 (INT2)

IF14–IF7 — Interrupt Flags 14–7

These flags indicate the presence of interrupt requests from the sources shown in Table 7-3.

1 = Interrupt request present
0 = No interrupt request present

7.5.2.3 Interrupt Status Register 3

Address: $FE06

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	IF21	IF20	IF19	IF18	IF17	IF16	IF15
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7-14. Interrupt Status Register 3 (INT3)

IF21–IF15 — Interrupt Flags 21–15

These flags indicate the presence of an interrupt request from the source shown in Table 7-3.

1 = Interrupt request present
0 = No interrupt request present

Table 8-8. READSP (Read Stack Pointer) Command

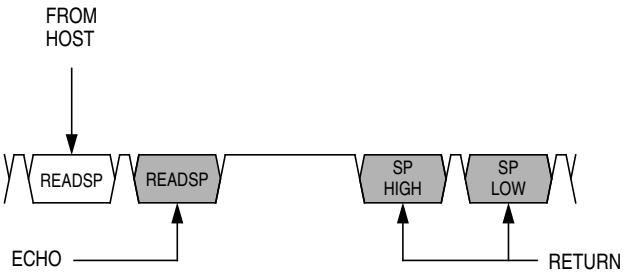
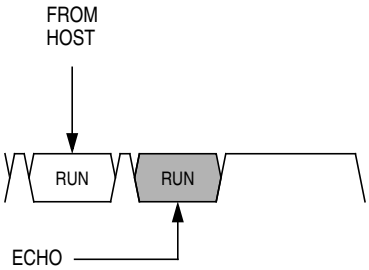
Description	Reads stack pointer
Operand	None
Data Returned	Returns incremented stack pointer value (SP + 1) in high-byte:low-byte order
Opcode	$0C
Command Sequence	

Table 8-9. RUN (Run User Program) Command

Description	Executes PULH and RTI instructions
Operand	None
Data Returned	None
Opcode	$28
Command Sequence	

The MCU executes the SWI and PSHH instructions when it enters monitor mode. The RUN command tells the MCU to execute the PULH and RTI instructions. Before sending the RUN command, the host can modify the stacked CPU registers to prepare to run the host program. The READSP command returns the incremented stack pointer value, SP + 1. The high and low bytes of the program counter are at addresses SP + 5 and SP + 6.

Upon power-on reset, if the received bytes of the security code do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

NOTE

The MCU does not transmit a break character until after the host sends the eight security bits.

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $60 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

8.5 ROM-Resident Routines

Seven routines stored in the monitor ROM area (thus ROM-resident) are provided for FLASH memory manipulation. Five of the seven routines are intended to simplify FLASH program, erase, and load operations. The other two routines are intended to simplify the use of the FLASH memory as EEPROM. Table 8-10 shows a summary of the ROM-resident routines.

Table 8-10. Summary of ROM-Resident Routines

Routine Name	Routine Description	Call Address	Stack Used (bytes)
PRGRNGE	Program a range of locations	$FC34	15
ERARNGE	Erase a page or the entire array	$FCE4	9
LDRNGE	Loads data from a range of locations	$FC00	7
MON_PRGRNGE	Program a range of locations in monitor mode	$FF24	17
MON_ERARNGE	Erase a page or the entire array in monitor mode	$FF28	11
EE_WRITE	Emulated EEPROM write. Data size ranges from 7 to 15 bytes at a time.	$FF36	30
EE_READ	Emulated EEPROM read. Data size ranges from 7 to 15 bytes at a time.	$FD5B	18

The routines are designed to be called as stand-alone subroutines in the user program or monitor mode. The parameters that are passed to a routine are in the form of a contiguous data block, stored in RAM. The index register (H:X) is loaded with the address of the first byte of the data block (acting as a pointer), and the subroutine is called (JSR). Using the start address as a pointer, multiple data blocks can be used, any area of RAM be used. A data block has the control and data bytes in a defined order, as shown in Figure 8-9.

9.4.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the TCH0 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The TIM channel 0 registers initially control the pulse width on the TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, TCH1, is available as a general-purpose I/O pin.

NOTE

In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.

9.4.4.3 PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use the following initialization procedure:

1. In the TIM status and control register (TSC):
   a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
   b. Reset the TIM counter and prescaler by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. (See Table 9-3.)
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB:ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See Table 9-3.)

   NOTE

   In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.
5. In the TIM status control register (TSC), clear the TIM stop bit, TSTOP.

10.5 Interrupts

The timebase module can interrupt the CPU on a regular basis with a rate defined by TBR[2:0]. When the timebase counter chain rolls over, the TBIF flag is set. If the TBIE bit is set, enabling the timebase interrupt, the counter chain overflow will generate a CPU interrupt request. The interrupt vector is defined in Table 2-1 . Vector Addresses.

Interrupts must be acknowledged by writing a logic 1 to the TACK bit.

10.6 Low-Power Modes

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

10.6.1 Wait Mode

The timebase module remains active after execution of the WAIT instruction. In wait mode, the timebase register is not accessible by the CPU.

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

10.6.2 Stop Mode

The timebase module may remain active after execution of the STOP instruction if the oscillator has been enabled to operate during stop mode through the stop mode oscillator enable bit (STOP_ICLKDIS, STOP_RCLKEN, or STOP_XCLKEN) for the selected oscillator in the CONFIG2 register. The timebase module can be used in this mode to generate a periodic walk-up from stop mode.

If the oscillator has not been enabled to operate in stop mode, the timebase module will not be active during stop mode. In stop mode the timebase register is not accessible by the CPU.

If the timebase functions are not required during stop mode, reduce the power consumption by stopping the timebase before enabling the STOP instruction.

11.5 Low-Power Modes

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

11.5.1 Wait Mode

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

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

Refer to 7.6 Low-Power Modes for information on exiting wait mode.

11.5.2 Stop Mode

The SCI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect SCI register states. SCI module operation resumes after an external interrupt.

Because the internal clock is inactive during stop mode, entering stop mode during an SCI transmission or reception results in invalid data.

Refer to 7.6 Low-Power Modes for information on exiting stop mode.

11.6 SCI During Break Module 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 SIM break flag control register (SBFCR) enables software to clear status bits during the break state.

To allow software to clear status bits during a break interrupt, write a logic 1 to the BCFE bit. 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 logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-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 at logic 0. After the break, doing the second step clears the status bit.

11.7 I/O Signals

Port B shares two of its pins with the SCI module.
The two SCI I/O pins are:

- PTB2/TxD — Transmit data
- PTB3/RxD — Receive data

11.7.1 TxD (Transmit Data)

When the SCI is enabled (ENSCI=1), the PTB2/TxD pin becomes the serial data output, TxD, from the SCI transmitter regardless of the state of the DDRB2 bit in data direction register B (DDRB). The TxD pin is an open-drain output and requires a pullup resistor to be connected for proper SCI operation.

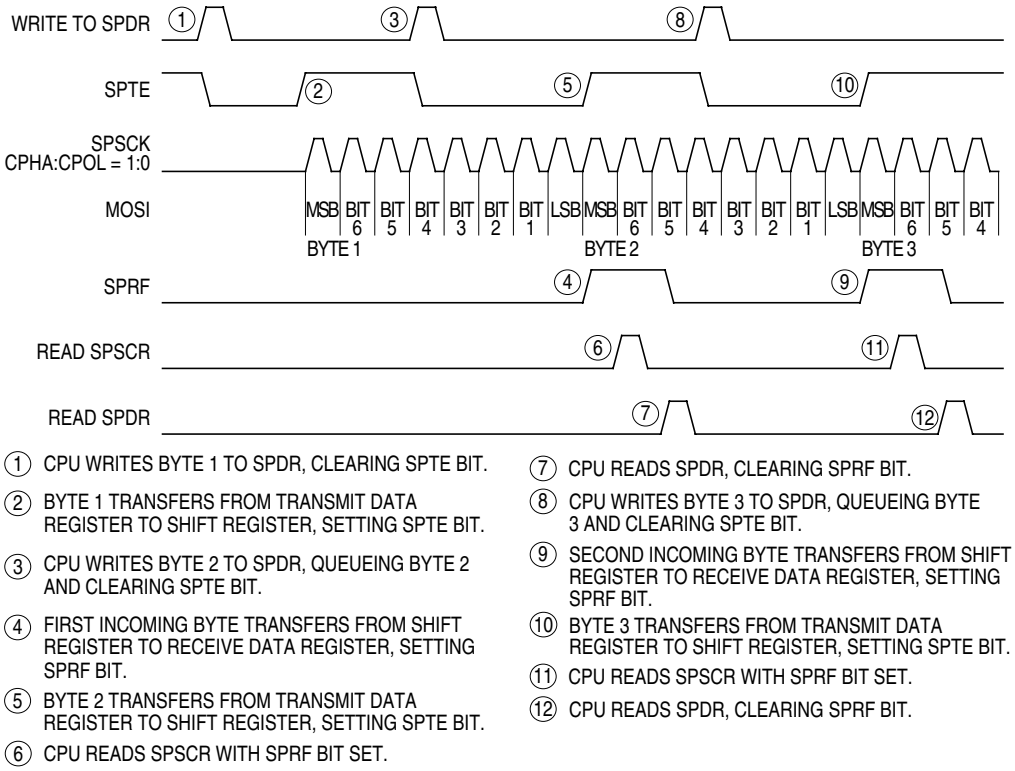


Figure 13-8. SPRF/SPTE CPU Interrupt Timing

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.

13.7 Error Conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — Failing to read the SPI data register before the next full byte enters the shift register sets the OVRF bit. The new byte does not transfer to the receive data register, and the unread byte still can be read. OVRF is in the SPI status and control register.
- Mode fault error (MODF) — The MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI status and control register.

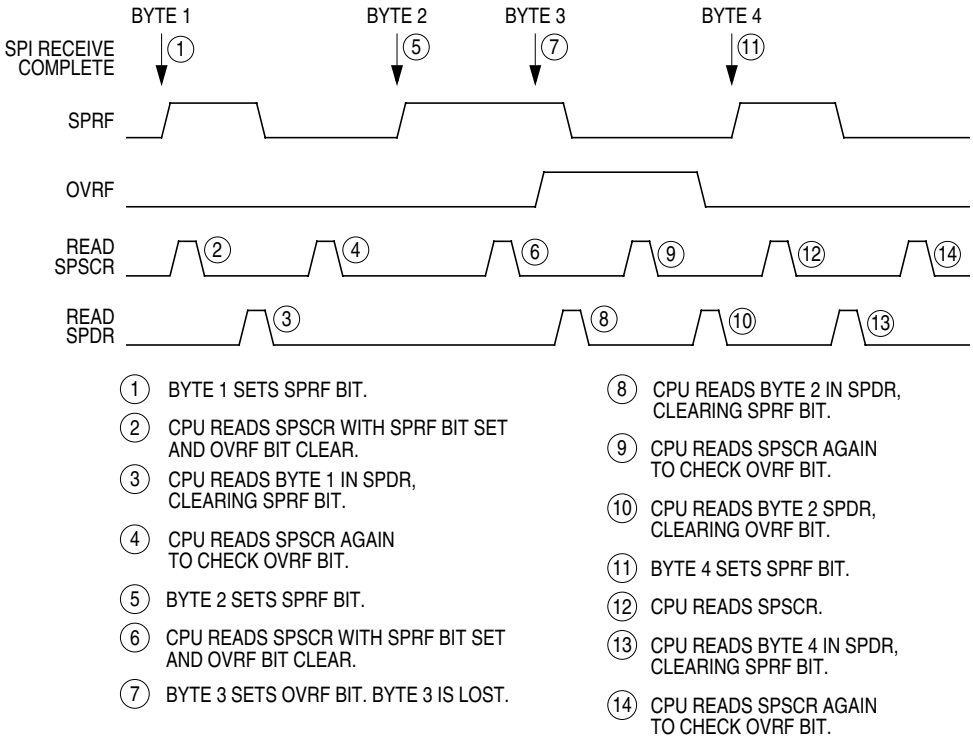


Figure 13-10. Clearing SPRF When OVRF Interrupt Is Not Enabled

13.7.2 Mode Fault Error

Setting the SPMSTR bit selects master mode and configures the SPSCK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SPSCK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, becomes set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR.

To prevent SPI pin contention and damage to the MCU, a mode fault error occurs if:

- The $\overline{SS}$ pin of a slave SPI goes high during a transmission
- The $\overline{SS}$ pin of a master SPI goes low at any time

For the MODF flag to be set, the mode fault error enable bit (MODFEN) must be set. Clearing the MODFEN bit does not clear the MODF flag but does prevent MODF from being set again after MODF is cleared.

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. (See Figure 13-11.) It is not possible to enable MODF or 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 the following 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.

14.3 I/O Pins

The MMIIC module uses two I/O pins, shared with standard port I/O pins. The full name of the MMIIC I/O pins are listed in Table 14-1. The generic pin name appear in the text that follows.

The SDA and SDL pins are open-drain. When configured as general purpose output pins (PTB0 and PTB1), pullup resistors must be connected to these pins.

Table 14-1. Pin Name Conventions

MMIIC Generic Pin Names:	Full MCU Pin Names:	Pin Selected for MMIIC Function By:
SDA	PTB0/SDA	MMEN bit in MMCR1 ($0049)
SCL	PTB1/SCL	

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0048	MMIIC Address Register (MMADR)	Read:	MMAD7	MMAD6	MMAD5	MMAD4	MMAD3	MMAD2	MMAD1	MMEXTAD
		Write:								
		Reset:	1	0	1	0	0	0	0	0
$0049	MMIIC Control Register 1 (MMCR1)	Read:	MMEN	MMIEN	0	0	MMTXAK	REPSEN	MMCRCBYTE	0
		Write:			MMCLRBB					
		Reset:	0	0	0	0	0	0	0	0
$004A	MMIIC Control Register 2 (MMCR2)	Read:	MMALIF	MMNAKIF	MMBB	MMAST	MMRW	0	0	MMCRCEF
		Write:	0	0						
		Reset:	0	0	0	0	0	0	0	Unaffected
$004B	MMIIC Status Register (MMSR)	Read:	MMRXIF	MMTXIF	MMATCH	MMSRW	MMRXAK	MMCRCBF	MMTXBE	MMRXBF
		Write:	0	0						
		Reset:	0	0	0	0	1	0	1	0
$004C	MMIIC Data Transmit Register (MMDTR)	Read:	MMTD7	MMTD6	MMTD5	MMTD4	MMTD3	MMTD2	MMTD1	MMTD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004D	MMIIC Data Receive Register (MDDRR)	Read:	MMRD7	MMRD6	MMRD5	MMRD4	MMRD3	MMRD2	MMRD1	MMRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004E	MMIIC CRC Data Register (MMCRDR)	Read:	MMCRCD7	MMCRCD6	MMCRCD5	MMCRCD4	MMCRCD3	MMCRCD2	MMCRCD1	MMCRCD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004F	MMIIC Frequency Divider Register (MMFDR)	Read:	0	0	0	0	0	MMBR2	MMBR1	MMBR0
		Write:								
		Reset:	0	0	0	0	0	1	0	0

= Unimplemented

Figure 14-1. MMIIC I/O Register Summary

14.4 Multi-Master IIC System Configuration

The multi-master IIC system uses a serial data line SDA and a serial clock line SCL for data transfer. All devices connected to it must have open collector (drain) outputs and the logical-AND function is performed on both lines by two pull-up resistors.

14.5 Multi-Master IIC Bus Protocol

Normally a standard communication is composed of four parts:

1. START signal,
2. slave address transmission,
3. data transfer, and
4. STOP signal.

These are described briefly in the following sections and illustrated in Figure 14-2.

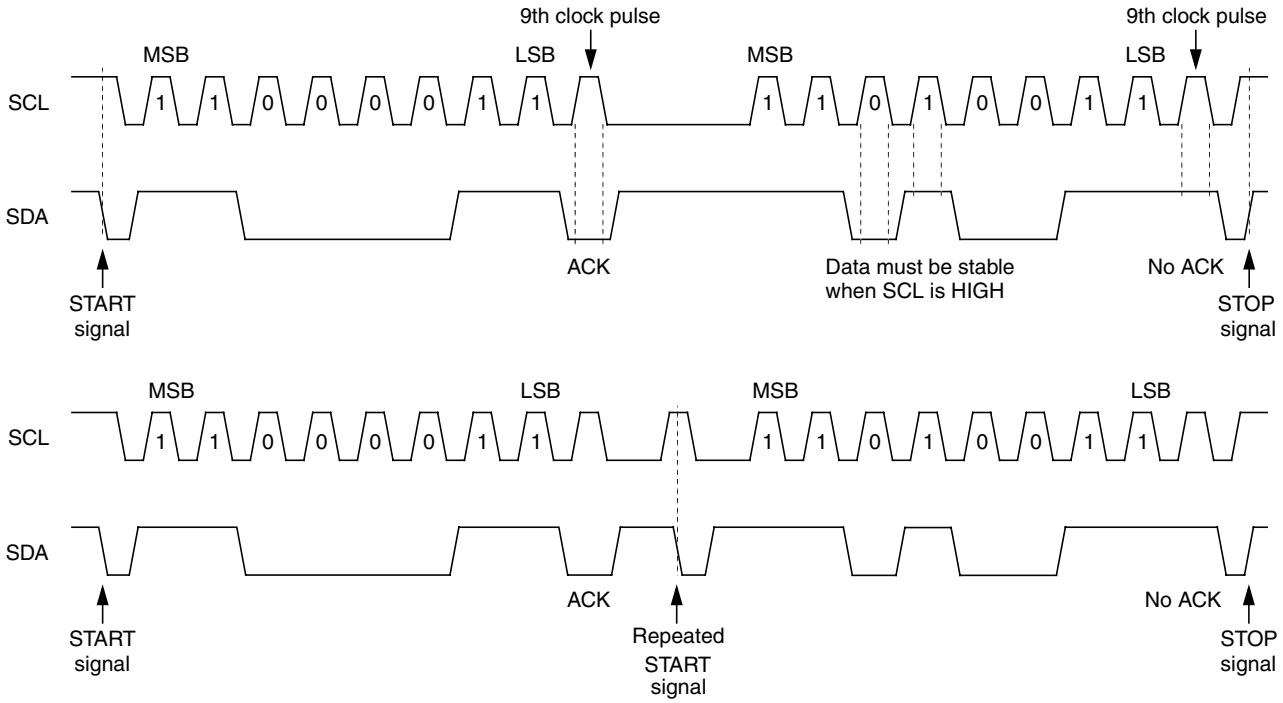


Figure 14-2. Multi-Master IIC Bus Transmission Signal Diagram

14.5.1 START Signal

When the bus is free, (i.e. no master device is engaging the bus — both SCL and SDA lines are at logic high) a master may initiate communication by sending a START signal. As shown in Figure 14-2, a START signal is defined as a high to low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and wakes up all slaves.

14.5.2 Slave Address Transmission

The first byte transferred immediately after the START signal is the slave address transmitted by the master. This is a 7-bit calling address followed by a R/W-bit. The R/W-bit dictates to the slave the desired direction of the data transfer. A logic 0 indicates that the master wishes to transmit data to the slave; a logic 1 indicates that the master wishes to receive data from the slave.

22.13 MMIIC Electrical Characteristics

Table 22-12. MMIIC DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ	Max	Unit	Comments
Input low	V_{IL}	–0.5	—	0.8	V	Data, clock input low.
Input high	V_{IH}	2.1	—	5.5	V	Data, clock input high.
Output low	V_{OL}	—	—	0.4	V	Data, clock output low; @$I_{PULLUP,MAX}$
Input leakage	I_{LEAK}	—	—	± 5	µA	Input leakage current
Pullup current	I_{PULLUP}	100	—	350	µA	Current through pull-up resistor or current source. See note.[2]

1. V_{DD} = 2.7 to 5.5Vdc, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted.
2. The I_{PULLUP} (max) specification is determined primarily by the need to accommodate a maximum of 1.1kΩ equivalent series resistor of removable SMBus devices, such as the smart battery, while maintaining the V_{OL} (max) of the bus.

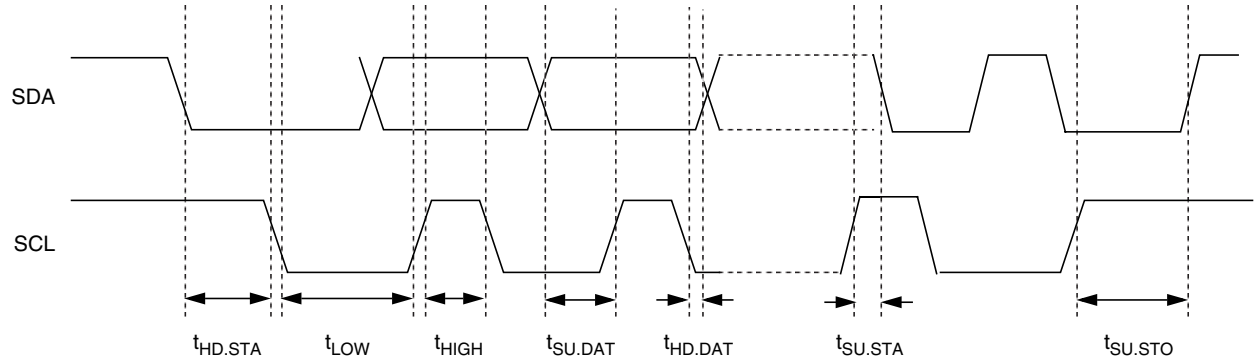


Figure 22-2. MMIIC Signal Timings

See Table 22-13 for MMIIC timing parameters.

22.15 5V SPI Characteristics

Table 22-15. SPI Characteristics (5V)

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 MHz
1	Cycle time Master Slave	 $t_{CYC(M)}$ $t_{CYC(S)}$	 2 1	 128 —	 t_{CYC} t_{CYC}
2	Enable lead time	$t_{Lead(S)}$	1	—	t_{CYC}
3	Enable lag time	$t_{Lag(S)}$	1	—	t_{CYC}
4	Clock (SPSCK) high time Master Slave	 $t_{SCKH(M)}$ $t_{SCKH(S)}$	 t_{CYC} –25 1/2 t_{CYC} –25	 64 t_{CYC} —	 ns ns
5	Clock (SPSCK) low time Master Slave	 $t_{SCKL(M)}$ $t_{SCKL(S)}$	 t_{CYC} –25 1/2 t_{CYC} –25	 64 t_{CYC} —	 ns ns
6	Data setup time (inputs) Master Slave	 $t_{SU(M)}$ $t_{SU(S)}$	 30 30	 — —	 ns ns
7	Data hold time (inputs) Master Slave	 $t_{H(M)}$ $t_{H(S)}$	 30 30	 — —	 ns ns
8	Access time, slave[3] CPHA = 0 CPHA = 1	 $t_{A(CP0)}$ $t_{A(CP1)}$	 0 0	 40 40	 ns ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	40	ns
10	Data valid time, after enable edge Master Slave[5]	 $t_{V(M)}$ $t_{V(S)}$	 — —	 50 50	 ns ns
11	Data hold time, outputs, after enable edge Master Slave	 $t_{HO(M)}$ $t_{HO(S)}$	 0 0	 — —	 ns ns

1. Numbers refer to dimensions in Figure 22-3 and Figure 22-4.
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins