

Welcome to [E-XFL.COM](https://www.e-xfl.com)

What is "[Embedded - Microcontrollers](#)"?

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

Applications of "[Embedded - Microcontrollers](#)"

Details

Product Status	Active
Core Processor	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	SCI
Peripherals	LED, LVD, POR, PWM
Number of I/O	26
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 13x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	32-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68hc908jl8mfae

Chapter 11 Input/Output (I/O) Ports

11.1	Introduction	151
11.2	Port A	153
11.2.1	Port A Data Register (PTA)	153
11.2.2	Data Direction Register A (DDRA)	153
11.2.3	Port A Input Pull-Up Enable Registers	155
11.3	Port B	156
11.3.1	Port B Data Register (PTB)	156
11.3.2	Data Direction Register B (DDRB)	156
11.4	Port D	157
11.4.1	Port D Data Register (PTD)	158
11.4.2	Data Direction Register D (DDRD)	158
11.4.3	Port D Control Register (PDCR)	160
11.5	Port E	160
11.5.1	Port E Data Register (PTE)	160
11.5.2	Data Direction Register E (DDRE)	161

Chapter 12 External Interrupt (IRQ)

12.1	Introduction	163
12.2	Features	163
12.3	Functional Description	163
12.3.1	$\overline{\text{IRQ}}$ Pin	164
12.4	IRQ Module During Break Interrupts	165
12.5	IRQ Status and Control Register (INTSCR)	166

Chapter 13 Keyboard Interrupt Module (KBI)

13.1	Introduction	167
13.2	Features	167
13.3	I/O Pins	167
13.4	Functional Description	168
13.4.1	Keyboard Initialization	169
13.5	Keyboard Interrupt Registers	169
13.5.1	Keyboard Status and Control Register	169
13.5.2	Keyboard Interrupt Enable Register	170
13.6	Low-Power Modes	171
13.6.1	Wait Mode	171
13.6.2	Stop Mode	171
13.7	Keyboard Module During Break Interrupts	171

Chapter 14 Computer Operating Properly (COP)

14.1	Introduction	173
14.2	Functional Description	173

General Description

- 11 LED drivers (sink)
- 2 × 25mA open-drain I/O with pull-up
- Resident routines for in-circuit programming and EEPROM emulation
- System protection features:
 - Optional computer operating properly (COP) reset, driven by internal RC oscillator
 - Optional low-voltage detection with reset and selectable trip points for 3V and 5V operation
 - Illegal opcode detection with reset
 - Illegal address detection with reset
- Master reset pin with internal pull-up and power-on reset
- $\overline{\text{IRQ}}$ with schmitt-trigger input and programmable pull-up
- 20-pin dual in-line package (PDIP), 20-pin small outline integrated package (SOIC), 28-pin PDIP, 28-pin SOIC, 32-pin shrink dual in-line package (SDIP), and 32-pin low-profile quad flat pack (LQFP)
- Specific features of the MC68HC908JL8 in 28-pin packages are:
 - 23 general-purpose I/Os only
 - 7 keyboard interrupt with internal pull-up
 - 10 LED drivers (sink)
 - 12-channel ADC
 - Timer I/O pins on TIM1 only
- Specific features of the MC68HC908JL8 in 20-pin packages are:
 - 15 general-purpose I/Os only
 - 1 keyboard interrupt with internal pull-up
 - 4 LED drivers (sink)
 - 10-channel ADC
 - Timer I/O pins on TIM1 only

Features of the CPU08 include the following:

- Enhanced HC05 programming model
- Extensive loop control functions
- 16 addressing modes (eight more than the HC05)
- 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
- Efficient C language support

1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908JL8.

Memory

8. Wait for time, t_{prog} (30 μs).
9. Repeat steps 7 and 8 until all bytes within the row are programmed.
10. Clear the PGM bit.
11. Wait for time, t_{nvh} (5 μs).
12. Clear the HVEN bit.
13. After time, t_{rcv} (1 μs), the memory can be accessed in read mode again.

This program sequence is repeated throughout the memory until all data is programmed.

NOTE

The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH addressed programmed to clearing the PGM bit (step 7 to step 10), must not exceed the maximum programming time, $t_{\text{prog max}}$.

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 shown, other unrelated operations may occur between the steps.

Table 4-1. Instruction Set Summary

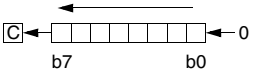
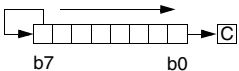
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
AND #opr AND opr AND opr, X AND opr, X AND ,X AND opr, SP AND opr, SP	Logical AND	$A \leftarrow (A) \& (M)$	0	–	–	–	–	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A4 B4 C4 D4 E4 F4 9EE4 9ED4	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
ASL opr ASLA ASLX ASL opr, X ASL ,X ASL opr, SP	Arithmetic Shift Left (Same as LSL)		–	–	–	–	–	–	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
ASR opr ASRA ASRX ASR opr, X ASR opr, X ASR opr, SP	Arithmetic Shift Right		–	–	–	–	–	–	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	4 1 1 4 3 5
BCC rel	Branch if Carry Bit Clear	$PC \leftarrow (PC) + 2 + rel ? (C) = 0$	–	–	–	–	–	–	REL	24	rr	3
BCLR n, opr	Clear Bit n in M	$M_n \leftarrow 0$	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BCS rel	Branch if Carry Bit Set (Same as BLO)	$PC \leftarrow (PC) + 2 + rel ? (C) = 1$	–	–	–	–	–	–	REL	25	rr	3
BEQ rel	Branch if Equal	$PC \leftarrow (PC) + 2 + rel ? (Z) = 1$	–	–	–	–	–	–	REL	27	rr	3
BGE opr	Branch if Greater Than or Equal To (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 0$	–	–	–	–	–	–	REL	90	rr	3
BGT opr	Branch if Greater Than (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (Z) \mid (N \oplus V) = 0$	–	–	–	–	–	–	REL	92	rr	3
BHCC rel	Branch if Half Carry Bit Clear	$PC \leftarrow (PC) + 2 + rel ? (H) = 0$	–	–	–	–	–	–	REL	28	rr	3
BHCS rel	Branch if Half Carry Bit Set	$PC \leftarrow (PC) + 2 + rel ? (H) = 1$	–	–	–	–	–	–	REL	29	rr	3
BHI rel	Branch if Higher	$PC \leftarrow (PC) + 2 + rel ? (C) \mid (Z) = 0$	–	–	–	–	–	–	REL	22	rr	3
BHS rel	Branch if Higher or Same (Same as BCC)	$PC \leftarrow (PC) + 2 + rel ? (C) = 0$	–	–	–	–	–	–	REL	24	rr	3
BIH rel	Branch if \overline{IRQ} Pin High	$PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 1$	–	–	–	–	–	–	REL	2F	rr	3
BIL rel	Branch if \overline{IRQ} Pin Low	$PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 0$	–	–	–	–	–	–	REL	2E	rr	3

Table 4-1. Instruction Set Summary

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BIT #opr BIT opr BIT opr BIT opr,X BIT opr,X BIT ,X BIT opr,SP BIT opr,SP	Bit Test	(A) & (M)	0	–	–	–	–	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
BLE opr	Branch if Less Than or Equal To (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (Z) \vee (N \oplus V) = 1$	–	–	–	–	–	–	REL	93	rr	3
BLO rel	Branch if Lower (Same as BCS)	$PC \leftarrow (PC) + 2 + rel ? (C) = 1$	–	–	–	–	–	–	REL	25	rr	3
BLS rel	Branch if Lower or Same	$PC \leftarrow (PC) + 2 + rel ? (C) \vee (Z) = 1$	–	–	–	–	–	–	REL	23	rr	3
BLT opr	Branch if Less Than (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 1$	–	–	–	–	–	–	REL	91	rr	3
BMC rel	Branch if Interrupt Mask Clear	$PC \leftarrow (PC) + 2 + rel ? (I) = 0$	–	–	–	–	–	–	REL	2C	rr	3
BMI rel	Branch if Minus	$PC \leftarrow (PC) + 2 + rel ? (N) = 1$	–	–	–	–	–	–	REL	2B	rr	3
BMS rel	Branch if Interrupt Mask Set	$PC \leftarrow (PC) + 2 + rel ? (I) = 1$	–	–	–	–	–	–	REL	2D	rr	3
BNE rel	Branch if Not Equal	$PC \leftarrow (PC) + 2 + rel ? (Z) = 0$	–	–	–	–	–	–	REL	26	rr	3
BPL rel	Branch if Plus	$PC \leftarrow (PC) + 2 + rel ? (N) = 0$	–	–	–	–	–	–	REL	2A	rr	3
BRA rel	Branch Always	$PC \leftarrow (PC) + 2 + rel$	–	–	–	–	–	–	REL	20	rr	3
BRCLR n,opr,rel	Branch if Bit n in M Clear	$PC \leftarrow (PC) + 3 + rel ? (Mn) = 0$	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN rel	Branch Never	$PC \leftarrow (PC) + 2$	–	–	–	–	–	–	REL	21	rr	3
BRSET n,opr,rel	Branch if Bit n in M Set	$PC \leftarrow (PC) + 3 + rel ? (Mn) = 1$	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET n,opr	Set Bit n in M	$Mn \leftarrow 1$	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4

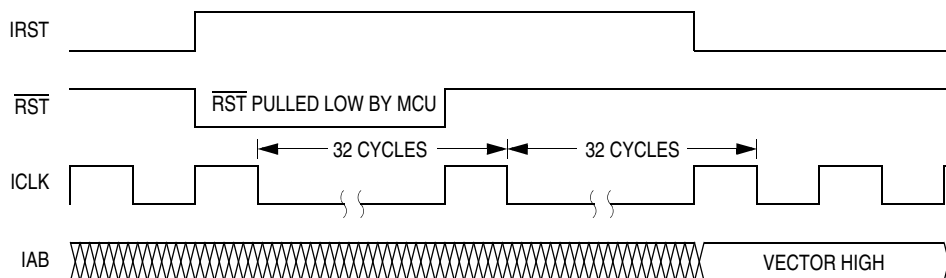


Figure 5-5. Internal Reset Timing

The COP reset is asynchronous to the bus clock.

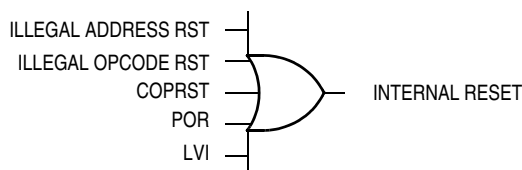


Figure 5-6. Sources of Internal Reset

The active reset feature allows the part to issue a reset to peripherals and other chips within a system built around the MCU.

5.3.2.1 Power-On Reset

When power is first applied to the MCU, the power-on reset module (POR) generates a pulse to indicate that power-on has occurred. The external reset pin ($\overline{\text{RST}}$) is held low while the SIM counter counts out 4096 ICLK cycles. Sixty-four ICLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, the following events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.
- The SIM enables OSCOUT.
- Internal clocks to the CPU and modules are held inactive for 4096 ICLK cycles to allow stabilization of the oscillator.
- The $\overline{\text{RST}}$ pin is driven low during the oscillator stabilization time.
- The POR bit of the reset status register (RSR) is set and all other bits in the register are cleared.

5.5.5 Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select whether flags are protected from being cleared by properly initializing the break clear flag enable bit (BCFE) in the break flag control register (BFCR).

Protecting flags in break mode ensures that set flags will not be cleared while in break mode. This protection allows registers to be freely read and written during break mode without losing status flag information.

Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a two-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

5.6 Low-Power Modes

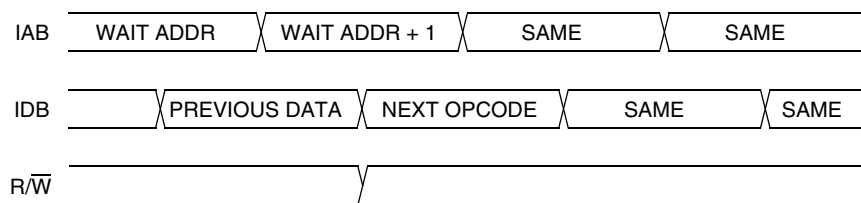
Executing the WAIT or STOP instruction puts the MCU in a low-power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

5.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. [Figure 5-15](#) shows the timing for wait mode entry.

A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the break status register (BSR). If the COP disable bit, COPD, in the mask option register is logic zero, then the computer operating properly module (COP) is enabled and remains active in wait mode.



NOTE: Previous data can be operand data or the WAIT opcode, depending on the last instruction.

Figure 5-15. Wait Mode Entry Timing

[Figure 5-16](#) and [Figure 5-17](#) show the timing for WAIT recovery.

SBSW — SIM Break Stop/Wait

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

- 1 = Stop mode or wait mode was exited by break interrupt
- 0 = Stop mode or wait mode was not exited by break interrupt

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it. The following code is an example of this. Writing zero to the SBSW bit clears it.

```

; This code works if the H register has been pushed onto the stack in the break
; service routine software. This code should be executed at the end of the
; break service routine software.

HIBYTE EQU 5
LOBYTE EQU 6

; If not SBSW, do RTI
BRCLR SBSW,BSR, RETURN ; See if wait mode or stop mode was exited
; by break.

TST LOBYTE,SP ; If RETURNLO is not zero,
BNE DOLO ; then just decrement low byte.
DEC HIBYTE,SP ; Else deal with high byte, too.
DOLO DEC LOBYTE,SP ; Point to WAIT/STOP opcode.
RETURN PULH ; Restore H register.
RTI

```

5.7.2 Reset Status Register (RSR)

This register contains six flags that show the source of the last reset. Clear the SIM reset status register by reading it. A power-on reset sets the POR bit and clears all other bits in the register.

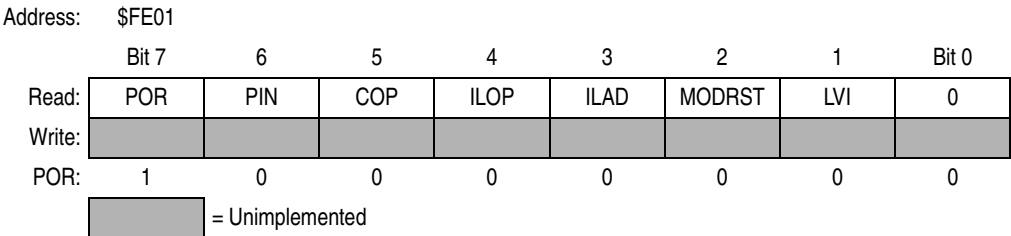


Figure 5-21. Reset Status Register (RSR)

POR — Power-On Reset Bit

- 1 = Last reset caused by POR circuit
- 0 = Read of RSR

Chapter 8

Timer Interface Module (TIM)

8.1 Introduction

This section describes the timer interface (TIM) module. The TIM is a two-channel timer that provides a timing reference with Input capture, output compare, and pulse-width-modulation functions. [Figure 8-1](#) is a block diagram of the TIM.

This particular MCU has two timer interface modules which are denoted as TIM1 and TIM2.

8.2 Features

Features of the TIM include:

- Two input capture/output compare channels:
 - Rising-edge, falling-edge, or any-edge input capture trigger
 - Set, clear, or toggle output compare action
- Buffered and unbuffered pulse-width-modulation (PWM) signal generation
- Programmable TIM clock input
 - 7-frequency internal bus clock prescaler selection
 - External clock input on timer 2 (bus frequency $\div 2$ maximum)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIM counter stop and reset bits

8.3 Pin Name Conventions

The text that follows describes both timers, TIM1 and TIM2. The TIM input/output (I/O) pin names are T[1,2]CH0 (timer channel 0) and T[1,2]CH1 (timer channel 1), where “1” is used to indicate TIM1 and “2” is used to indicate TIM2. The two TIMs share four I/O pins with four I/O port pins. The external clock input for TIM2 is shared with the an ADC channel pin. The full names of the TIM I/O pins are listed in [Table 8-1](#). The generic pin names appear in the text that follows.

Table 8-1. Pin Name Conventions

TIM Generic Pin Names:		T[1,2]CH0	T[1,2]CH1	T2CLK
Full TIM Pin Names:	TIM1	PTD4/T1CH0	PTD5/T1CH1	—
	TIM2	PTE0/T2CH0	PTE1/T2CH1	ADC12/T2CLK

NOTE

References to either timer 1 or timer 2 may be made in the following text by omitting the timer number. For example, TCH0 may refer generically to T1CH0 and T2CH0, and TCH1 may refer to T1CH1 and T2CH1.

Fast Data Tolerance

Figure 9-8 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

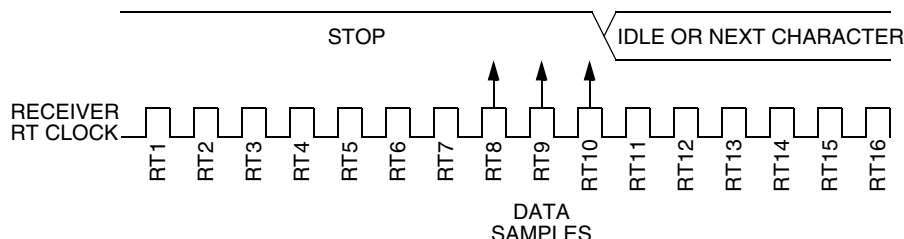


Figure 9-8. Fast Data

For an 8-bit character, data sampling of the stop bit takes the receiver
 $9 \text{ bit times} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 154 \text{ RT cycles}$.

With the misaligned character shown in Figure 9-8, the receiver counts 154 RT cycles at the point when the count of the transmitting device is
 $10 \text{ bit times} \times 16 \text{ RT cycles} = 160 \text{ RT cycles}$.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is

$$\left| \frac{154 - 160}{154} \right| \times 100 = 3.90\%$$

For a 9-bit character, data sampling of the stop bit takes the receiver
 $10 \text{ bit times} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 170 \text{ RT cycles}$.

With the misaligned character shown in Figure 9-8, the receiver counts 170 RT cycles at the point when the count of the transmitting device is
 $11 \text{ bit times} \times 16 \text{ RT cycles} = 176 \text{ RT cycles}$.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is

$$\left| \frac{170 - 176}{170} \right| \times 100 = 3.53\%$$

9.4.3.6 Receiver Wakeup

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

Depending on the state of the WAKE bit in SCC1, either of two conditions on the RxD pin can bring the receiver out of the standby state:

9.5 Low-Power Modes

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

9.5.1 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 [5.6 Low-Power Modes](#) for information on exiting wait mode.

9.5.2 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 [5.6 Low-Power Modes](#) for information on exiting stop mode.

9.6 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 break flag control register (BFCR) 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.

9.7 I/O Signals

The two SCI I/O pins are:

- PTD6/TxD — Transmit data
- PTD7/RxD — Receive data

9.7.1 TxD (Transmit Data)

The PTD6/TxD pin is the serial data output from the SCI transmitter.

9.7.2 RxD (Receive Data)

The PTD7/RxD pin is the serial data input to the SCI receiver.

9.8.3 SCI Control Register 3

SCI control register 3:

- Stores the ninth SCI data bit received and the ninth SCI data bit to be transmitted
- Enables these interrupts:
 - Receiver overrun interrupts
 - Noise error interrupts
 - Framing error interrupts
- Parity error interrupts

Address:	\$0015							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
Write:								
Reset:	U	U	0	0	0	0	0	0
	= Unimplemented		U = Unaffected					

Figure 9-11. SCI Control Register 3 (SCC3)

R8 — Received Bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.

When the SCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

T8 — Transmitted Bit 8

When the SCI is transmitting 9-bit characters, T8 is the read/write ninth bit (bit 8) of the transmitted character. T8 is loaded into the transmit shift register at the same time that the SCDR is loaded into the transmit shift register. Reset has no effect on the T8 bit.

DMARE — DMA Receive Enable Bit

CAUTION

The DMA module is not included on this MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

1 = DMA not enabled to service SCI receiver DMA service requests generated by the SCRF bit (SCI receiver CPU interrupt requests enabled)

0 = DMA not enabled to service SCI receiver DMA service requests generated by the SCRF bit (SCI receiver CPU interrupt requests enabled)

DMATE — DMA Transfer Enable Bit

CAUTION

The DMA module is not included on this MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

1 = SCTE DMA service requests enabled; SCTE CPU interrupt requests disabled

0 = SCTE DMA service requests disabled; SCTE CPU interrupt requests enabled

ORIE — Receiver Overrun Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the receiver overrun bit, OR.

1 = SCI error CPU interrupt requests from OR bit enabled

0 = SCI error CPU interrupt requests from OR bit disabled

9.8.7 SCI Baud Rate Register

The baud rate register (SCBR) selects the baud rate for both the receiver and the transmitter.

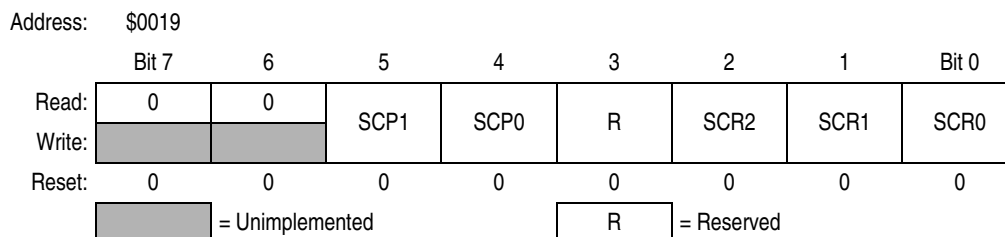


Figure 9-16. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in [Table 9-6](#). Reset clears SCP1 and SCP0.

Table 9-6. SCI Baud Rate Prescaling

SCP1 and SCP0	Prescaler Divisor (PD)
00	1
01	3
10	4
11	13

SCR2–SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in [Table 9-7](#). Reset clears SCR2–SCR0.

Table 9-7. SCI Baud Rate Selection

SCR2, SCR1, and SCR0	Baud Rate Divisor (BD)
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Use this formula to calculate the SCI baud rate:

$$\text{baud rate} = \frac{\text{SCI clock source}}{64 \times \text{PD} \times \text{BD}}$$

where:

SCI clock source = bus clock

PD = prescaler divisor

BD = baud rate divisor

[Table 9-8](#) shows the SCI baud rates that can be generated with a 4.9152MHz bus clock.

11.4.1 Port D Data Register (PTD)

The port D data register contains a data latch for each of the eight port D pins.

Address:	\$0003							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
Reset:	Unaffected by reset							
Additional Functions	LED (Sink)	LED (Sink)			LED (Sink)	LED (Sink)		
	25mA sink (Slow Edge)	25mA sink (Slow Edge)						
	pull-up	pull-up						
Alternative Functions:	RxD	TxD	T1CH1	T1CH0	ADC8	ADC9	ADC10	ADC11

Figure 11-10. Port D Data Register (PTD)

PTD[7:0] — Port D Data Bits

These read/write bits are software programmable. Data direction of each port D pin is under the control of the corresponding bit in data direction register D. Reset has no effect on port D data.

ADC11–ADC8 — ADC channels 11 to 8

ADC[11:8] are pins used for the input channels to the analog-to-digital converter module. The channel select bits, ADCH[4:0], in the ADC status and control register define which port pin will be used as an ADC input and overrides any control from the port I/O logic. See [Chapter 10 Analog-to-Digital Converter \(ADC\)](#).

T1CH1, T1CH0 — Timer 1 Channel I/Os

The T1CH1 and T1CH0 pins are the TIM1 input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTD4/T1CH0 and PTD5/T1CH1 pins are timer channel I/O pins or general-purpose I/O pins. See [Chapter 8 Timer Interface Module \(TIM\)](#).

TxD, RxD — SCI Data I/O Pins

The TxD and RxD pins are the transmit data output and receive data input for the SCI module. The enable SCI bit, ENSCI, in the SCI control register 1 enables the PTD6/TxD and PTD7/RxD pins as SCI TxD and RxD pins and overrides any control from the port I/O logic. See [Chapter 9 Serial Communications Interface \(SCI\)](#).

11.4.2 Data Direction Register D (DDRD)

Data direction register D determines whether each port D pin is an input or an output. Writing a logic 1 to a DDRD bit enables the output buffer for the corresponding port D pin; a logic 0 disables the output buffer.

NOTE

For those devices packaged in a 20-pin package, PTD0–PTD1 and are not connected. DDRD0–DDRD1 should be set to a 1 to configure PTD0–PTD1 as outputs.

If the MODEK bit is clear, the keyboard interrupt pin is falling-edge-sensitive only. With MODEK clear, a vector fetch or software clear immediately clears the keyboard interrupt request.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic 0.

The keyboard flag bit (KEYF) in the keyboard status and control register can be used to see if a pending interrupt exists. The KEYF bit is not affected by the keyboard interrupt mask bit (IMASKK) which makes it useful in applications where polling is preferred.

To determine the logic level on a keyboard interrupt pin, disable the pull-up device, use the data direction register to configure the pin as an input and then read the data register.

NOTE

Setting a keyboard interrupt enable bit (KBIE_x) forces the corresponding keyboard interrupt pin to be an input, overriding the data direction register. However, the data direction register bit must be a logic 0 for software to read the pin.

13.4.1 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pull-up to reach a logic 1. Therefore a false interrupt can occur as soon as the pin is enabled.

To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register.
2. Enable the KBI pins by setting the appropriate KBIE_x bits in the keyboard interrupt enable register.
3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts.
4. Clear the IMASKK bit.

An interrupt signal on an edge-triggered pin can be acknowledged immediately after enabling the pin. An interrupt signal on an edge- and level-triggered interrupt pin must be acknowledged after a delay that depends on the external load.

Another way to avoid a false interrupt:

1. Configure the keyboard pins as outputs by setting the appropriate DDRA bits in the data direction register A.
2. Write logic 1's to the appropriate port A data register bits.
3. Enable the KBI pins by setting the appropriate KBIE_x bits in the keyboard interrupt enable register.

13.5 Keyboard Interrupt Registers

Two registers control the operation of the keyboard interrupt module:

- Keyboard status and control register
- Keyboard interrupt enable register

13.5.1 Keyboard Status and Control Register

- Flags keyboard interrupt requests
- Acknowledges keyboard interrupt requests
- Masks keyboard interrupt requests
- Controls keyboard interrupt triggering sensitivity



Chapter 16

Break Module (BREAK)

16.1 Introduction

This section describes the break module. The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

16.2 Features

Features of the break module include the following:

- Accessible I/O registers during the break Interrupt
- CPU-generated break interrupts
- Software-generated break interrupts
- COP disabling during break interrupts

16.3 Functional Description

When the internal address bus matches the value written in the break address registers, the break module issues a breakpoint signal ($\overline{\text{BKPT}}$) to the SIM. The SIM then causes the CPU to load the instruction register with a software interrupt instruction (SWI) after completion of the current CPU instruction. The program counter vectors to \$FFFC and \$FFFD (\$FEFC and \$FEFD in monitor mode).

The following events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a logic one to the BRKA bit in the break status and control register.

When a CPU generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return from interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. [Figure 16-1](#) shows the structure of the break module.

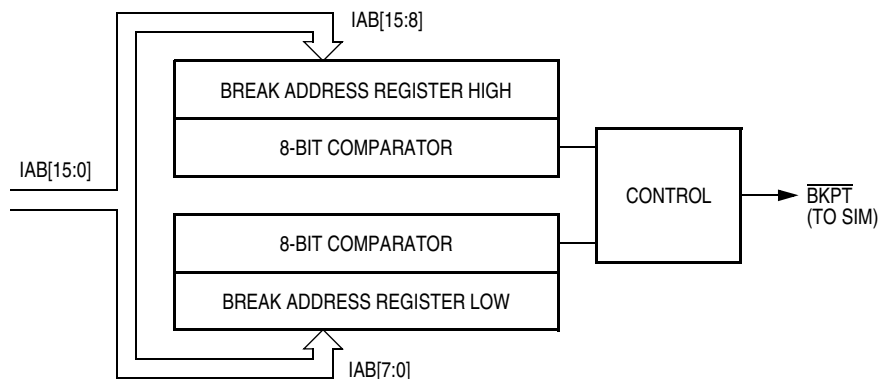


Figure 16-1. Break Module Block Diagram

Break Module (BREAK)

Address:	\$FE0D							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:								
Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 16-5. Break Address Register Low (BRKL)

16.4.3 Break Status Register

The break status register contains a flag to indicate that a break caused an exit from stop or 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

1. Writing a logic zero clears SBSW.

Figure 16-6. Break Status Register (BSR)

SBSW — SIM Break Stop/Wait

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

1 = Stop mode or wait mode was exited by break interrupt

0 = Stop mode or wait mode was not exited by break interrupt

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it. The following code is an example of this.

```
; This code works if the H register has been pushed onto the stack in the break
; service routine software. This code should be executed at the end of the
; break service routine software.

HIBYTE EQU 5
LOBYTE EQU 6

; If not SBSW, do RTI

BRCLR SBSW,BSR, RETURN ; See if wait mode or stop mode was exited
; by break.

TST LOBYTE,SP ; If RETURNLO is not zero,
BNE DOLO ; then just decrement low byte.
DEC HIBYTE,SP ; Else deal with high byte, too.
DOLO DEC LOBYTE,SP ; Point to WAIT/STOP opcode.
RETURN PULH ; Restore H register.
RTI
```

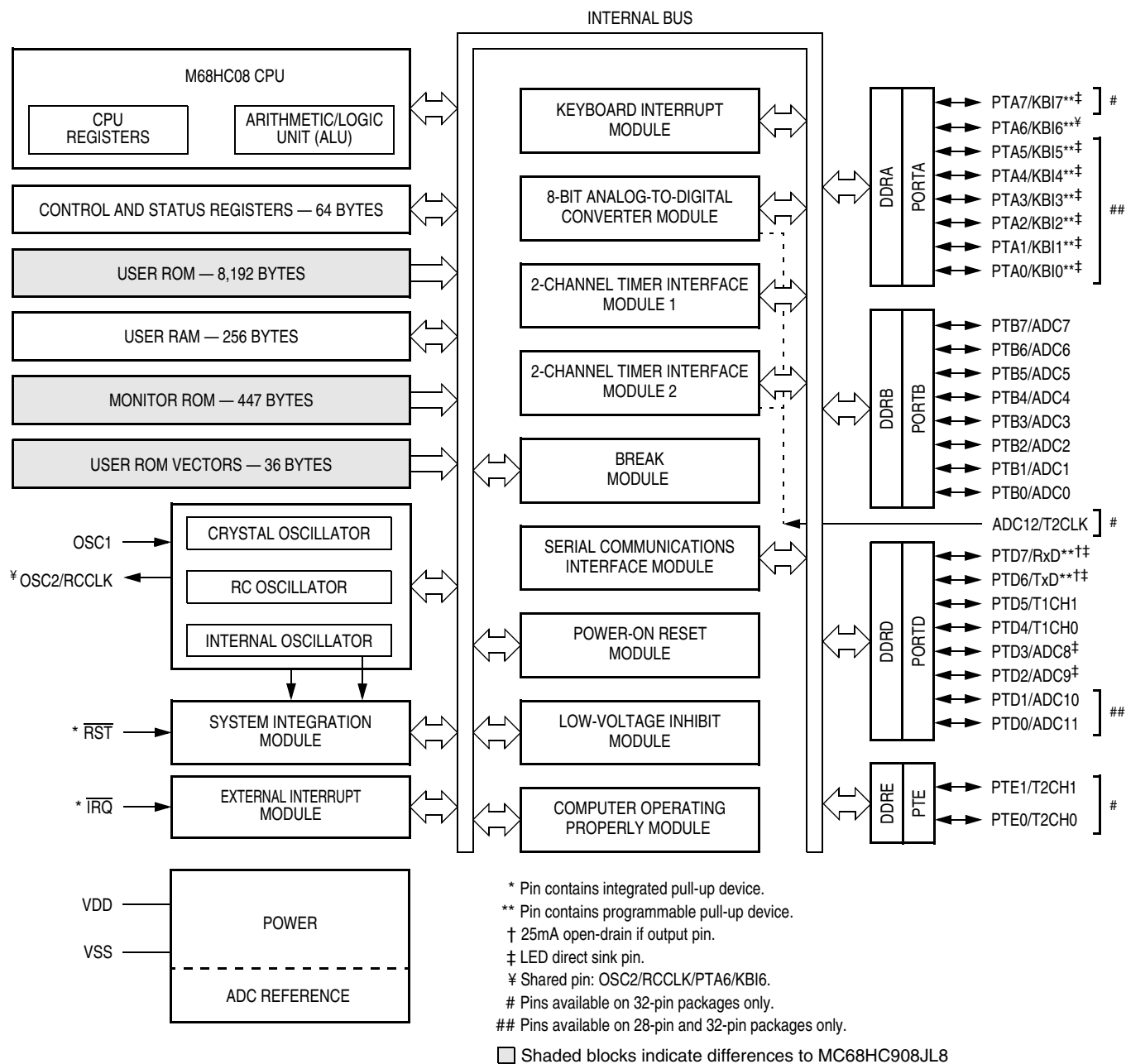


Figure A-1. MC68HC08JL8 Block Diagram

A.4 Reserved Registers

The two registers at \$FE08 and \$FFCF are reserved locations on the MC68HC08JL8.

On the MC68HC908JL8, these two locations are the FLASH control register and the FLASH block protect register respectively.

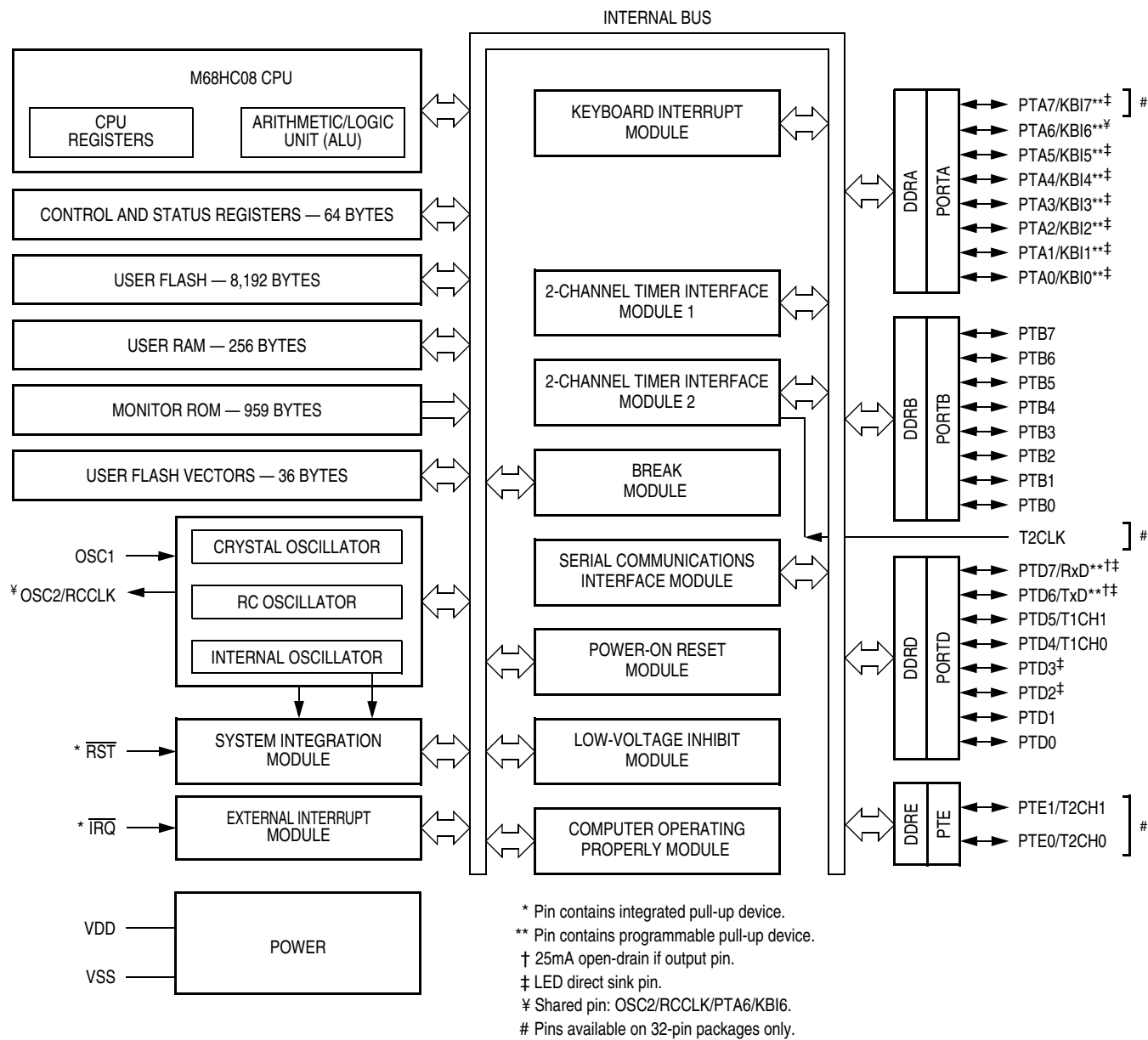


Figure B-1. MC68HC908KL8 Block Diagram