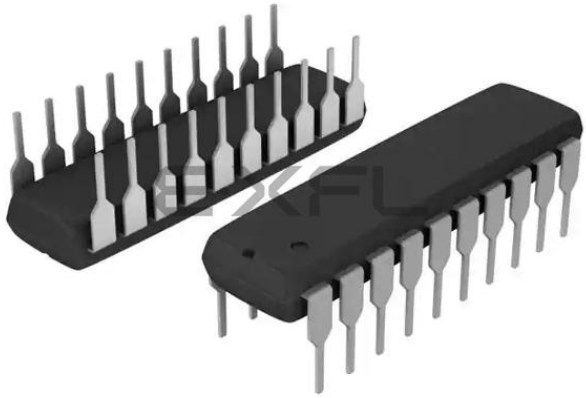


Welcome to E-XFL.COM

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	-
Peripherals	LED, LVD, POR, PWM
Number of I/O	15
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.3V
Data Converters	A/D 12x8b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	20-DIP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mcr908jk3ecpe

MC68HC908JL3/JK3E/JK1E
MC68HRC908JL3/JK3E/JK1E
MC68HLC908JL3/JK3E/JK1E
MC68HC908KL3E/KK3E
MC68HC08JL3E/JK3E
MC68HRC08JL3E/JK3E

Data Sheet

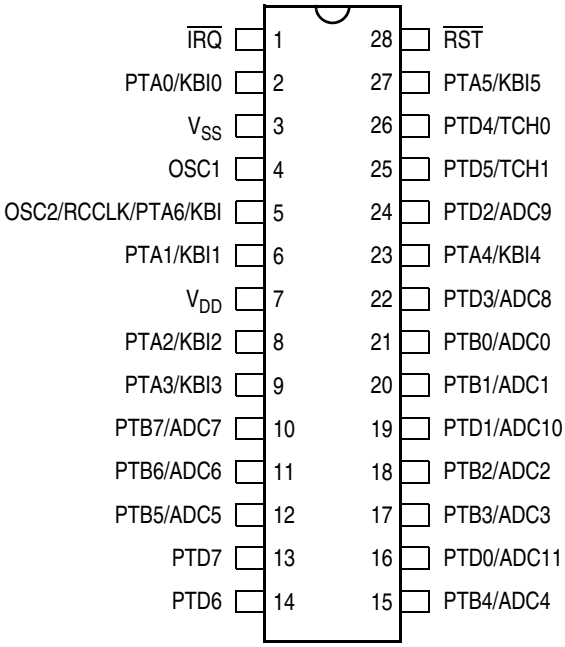
To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://www.freescale.com

List of Chapters

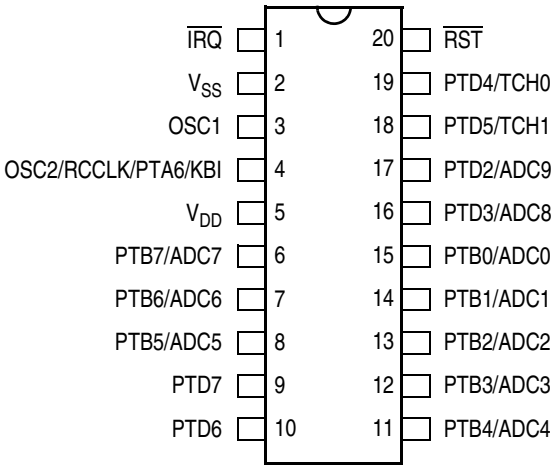
Table of Contents

1.4 Pin Assignments



MC68H(R)C908JL3E

Figure 1-2. 28-Pin PDIP/SOIC Pin Assignment



Pins not available on 20-pin packages	
PTA0/KBI0	PTD0/ADC11
PTA1/KBI1	PTD1/ADC10
PTA2/KBI2	
PTA3/KBI3	
PTA4/KBI4	
PTA5/KBI5	
Internal pads are unconnected.	

MC68H(R)C908JK3E/JK1E

Figure 1-3. 20-Pin PDIP/SOIC Pin Assignment

2.10 Flash Program Operation

Programming of the Flash memory is done on a row basis. A row consists of 32 consecutive bytes starting from addresses $XX00, $XX20, $XX40, $XX60, $XX80, $XXA0, $XXC0 or $XXE0. Use this step-by-step procedure to program a row of Flash memory (Figure 2-5 shows a flowchart of the programming algorithm):

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Write any data to any Flash location within the address range of the row to be programmed.
3. Wait for a time, t_{nvs} (10μs).
4. Set the HVEN bit.
5. Wait for a time, t_{pgs} (5μs).
6. Write data to the byte being programmed.
7. Wait for time, t_{PROG} (30μs).
8. Repeat step 6 and 7 until all the bytes within the row are programmed.
9. Clear the PGM bit.
10. Wait for time, t_{nvh} (5μs).
11. Clear the HVEN bit.
12. After time, t_{rcv} (1μs), 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 6 to step 6), or the time between the last Flash addressed programmed to clearing the PGM bit (step 6 to step 10), must not exceed the maximum programming time, t_{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.

2.11 Flash Protection

Due to the ability of the on-board charge pump to erase and program the Flash memory in the target application, provision is made to protect blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by use of a Flash Block Protect Register (FLBPR). The FLBPR determines the range of the Flash memory which is to be protected. The range of the protected area starts from a location defined by FLBPR and ends to the bottom of the Flash memory ($FFFF). When the memory is protected, the HVEN bit cannot be set in either ERASE or PROGRAM operations.

2.12 Flash Block Protect Register

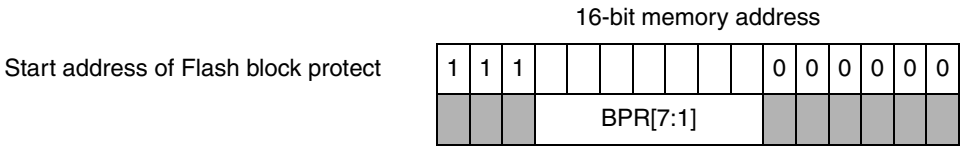
The Flash Block Protect Register is implemented as an 8-bit I/O register. The value in this register determines the starting address of the protected range within the Flash memory.

Address:	$FE09							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
Reset:	0	0	0	0	0	0	0	0

Figure 2-6. Flash Block Protect Register (FLBPR)

BPR[7:0] — Flash Block Protect Register Bit 7 to Bit 0

BPR[7:1] represent bits [12:6] of a 16-bit memory address. Bits [15:13] are 1's and bits [5:0] are 0's.



BPR0 is used only for BPR[7:0] = $FF, for no block protection.
The resultant 16-bit address is used for specifying the start address of the Flash memory for block protection. The Flash is protected from this start address to the end of Flash memory, at $FFFF. With this mechanism, the protect start address can be XX00, XX40, XX80, or XXC0 (at page boundaries — 64 bytes) within the Flash memory.

Examples of protect start address:

BPR[7:0]	Start of Address of Protect Range
$00–$60	The entire Flash memory is protected.
$62 or $63 (**0110 001x**)	$EC40 (111**0 1100 01**00 0000)
$64 or $65 (**0110 010x**)	$EC80 (111**0 1100 10**00 0000)
$68 or $69 (**0110 100x**)	$ED00 (111**0 1101 00**00 0000)
and so on...	
$DE or $DF (**1101 111x**)	$FBC0 (111**1 1011 11**00 0000)
$FE (**1111 1110**)	$FFC0 (111**1 1111 11**00 0000)
$FF	The entire Flash memory is not protected.

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

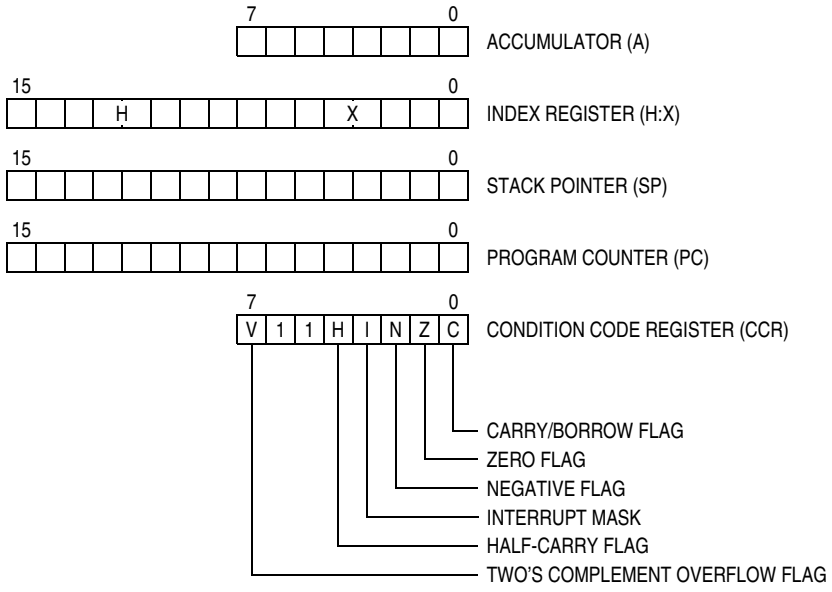


Figure 4-1. CPU Registers

4.3.1 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.

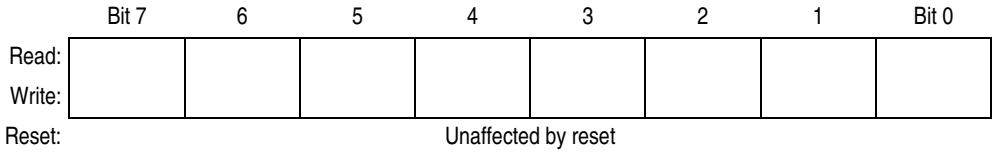


Figure 4-2. Accumulator (A)

4.3.2 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.

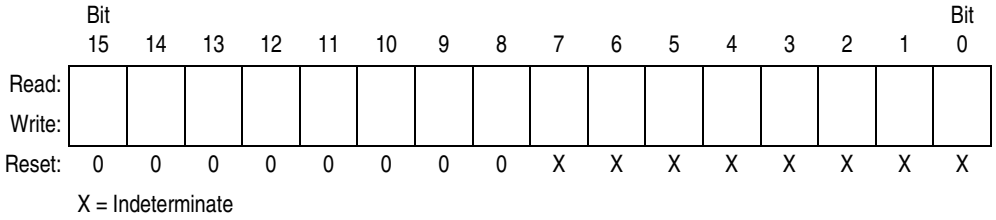


Figure 4-3. Index Register (H:X)

4.3.3 Stack Pointer

The stack pointer is a 16-bit register that contains the address of the next location on the stack. During a reset, the stack pointer is preset to $00FF. The reset stack pointer (RSP) instruction sets the least significant byte to $FF and does not affect the most significant byte. The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

In the stack pointer 8-bit offset and 16-bit offset addressing modes, the stack pointer can function as an index register to access data on the stack. The CPU uses the contents of the stack pointer to determine the conditional address of the operand.

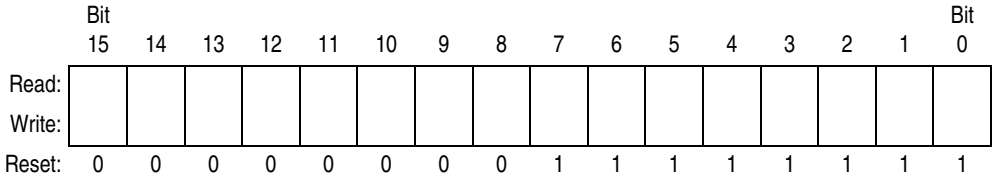


Figure 4-4. Stack Pointer (SP)

NOTE

The location of the stack is arbitrary and may be relocated anywhere in random-access memory (RAM). Moving the SP out of page 0 ($0000 to $00FF) frees direct address (page 0) space. For correct operation, the stack pointer must point only to RAM locations.

4.3.4 Program Counter

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched.

Normally, the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, and interrupt operations load the program counter with an address other than that of the next sequential location.

During reset, the program counter is loaded with the reset vector address located at $FFFE and $FFFF. The vector address is the address of the first instruction to be executed after exiting the reset state.

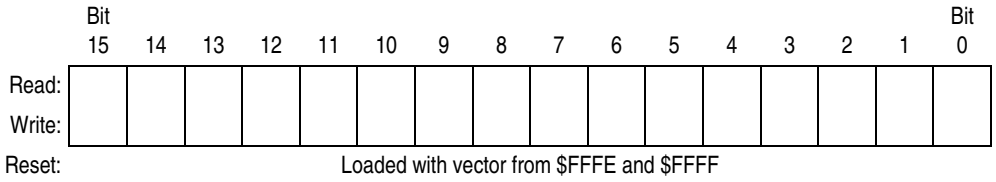
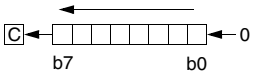
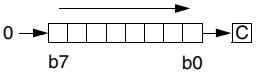


Figure 4-5. Program Counter (PC)

Table 4-1. Instruction Set Summary (Sheet 4 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	4 5 6 5 4
LDA #*opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X LDA *opr*,SP LDA *opr*,SP	Load A from M	A ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A6 B6 C6 D6 E6 F6 9EE6 9ED6	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LDHX #*opr* LDHX *opr*	Load H:X from M	H:X ← (M:M + 1)	0	–	–	↕	↕	–	IMM DIR	45 55	ii jj dd	3 4
LDX #*opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X LDX *opr*,SP LDX *opr*,SP	Load X from M	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9EEE 9EDE	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LSL *opr* LSLA LSLX LSL *opr*,X LSL ,X LSL *opr*,SP	Logical Shift Left (Same as ASL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
LSR *opr* LSRA LSR*X* LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr*,*opr* MOV *opr*,X+ MOV #*opr*,*opr* MOV X+,*opr*	Move	$(M)_{Destination} \leftarrow (M)_{Source}$ H:X ← (H:X) + 1 (IX+D, DIX+)	0	–	–	↕	↕	–	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	X:A ← (X) × (A)	–	0	–	–	–	0	INH	42		5
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X NEG *opr*,SP	Negate (Two's Complement)	M ← –(M) = $00 – (M) A ← –(A) = $00 – (A) X ← –(X) = $00 – (X) M ← –(M) = $00 – (M) M ← –(M) = $00 – (M)	↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	None	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap A	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		3
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X ORA *opr*,SP ORA *opr*,SP	Inclusive OR A and M	A ← (A) | (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AA BA CA DA EA FA 9EEA 9EDA	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
PSHA	Push A onto Stack	Push (A); SP ← (SP) – 1	–	–	–	–	–	–	INH	87		2
PSHH	Push H onto Stack	Push (H); SP ← (SP) – 1	–	–	–	–	–	–	INH	8B		2
PSHX	Push X onto Stack	Push (X); SP ← (SP) – 1	–	–	–	–	–	–	INH	89		2

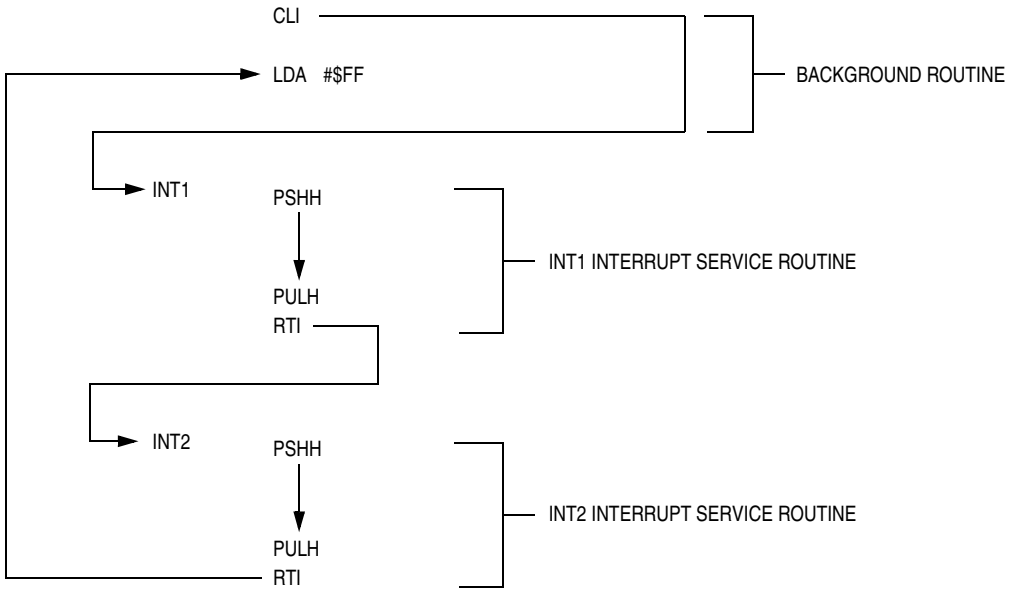


Figure 5-11. Interrupt Recognition Example

The LDA opcode is prefetched by both the INT1 and INT2 RTI instructions. However, in the case of the INT1 RTI prefetch, this is a redundant operation.

NOTE

To maintain compatibility with the M6805 Family, the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.

5.5.1.2 SWI Instruction

The SWI instruction is a non-maskable instruction that causes an interrupt regardless of the state of the interrupt mask (I bit) in the condition code register.

NOTE

A software interrupt pushes PC onto the stack. A software interrupt does ***not*** *push PC – 1, as a hardware interrupt does.*

5.5.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. Table 5-3 summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

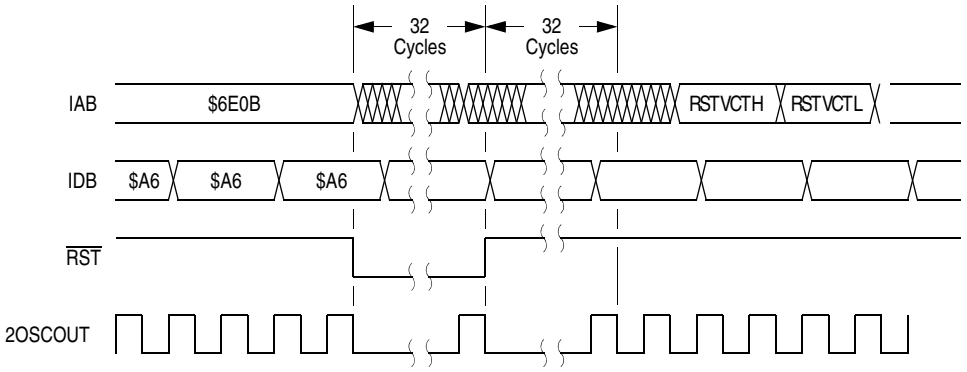


Figure 5-17. Wait Recovery from Internal Reset

5.6.2 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 oscillator signals (OSCOUT and 2OSCOUT) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the configuration register (CONFIG). If SSREC is set, stop recovery is reduced from the normal delay of 4096 2OSCOUT cycles down to 32. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode.

NOTE

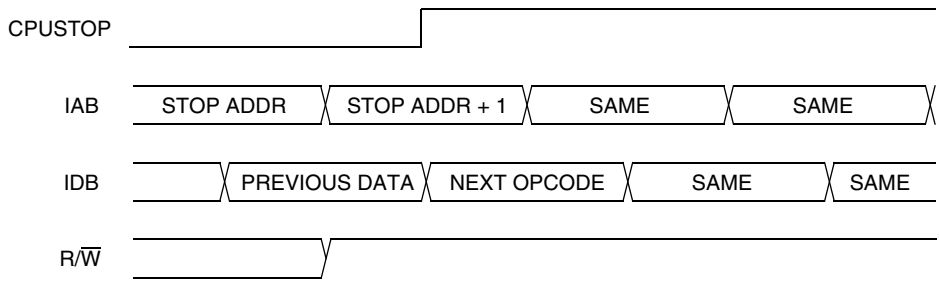
External crystal applications should use the full stop recovery time by clearing the SSREC bit.

A break interrupt during stop mode sets the SIM break stop/wait bit (SBSW) in the break status register (BSR).

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. Figure 5-18 shows stop mode entry timing.

NOTE

To minimize stop current, all pins configured as inputs should be driven to a logic 1 or logic 0.



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

Figure 5-18. Stop Mode Entry Timing

8.9.2 TIM Counter Registers (TCNTH:TCNTL)

The two read-only TIM counter registers contain the high and low bytes of the value in the TIM counter. Reading the high byte (TCNTH) latches the contents of the low byte (TCNTL) into a buffer. Subsequent reads of TCNTH do not affect the latched TCNTL value until TCNTL is read. Reset clears the TIM counter registers. Setting the TIM reset bit (TRST) also clears the TIM counter registers.

NOTE

If you read TCNTH during a break interrupt, be sure to unlatch TCNTL by reading TCNTL before exiting the break interrupt. Otherwise, TCNTL retains the value latched during the break.

Address: $0021 TCNTH

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Write:								
Reset:	0	0	0	0	0	0	0	0

Address: $0022 TCNTL

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 8-5. TIM Counter Registers (TCNTH:TCNTL)

8.9.3 TIM Counter Modulo Registers (TMODH:TMODL)

The read/write TIM modulo registers contain the modulo value for the TIM counter. When the TIM counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIM counter resumes counting from $0000 at the next timer clock. Writing to the high byte (TMODH) inhibits the TOF bit and overflow interrupts until the low byte (TMODL) is written. Reset sets the TIM counter modulo registers.

Address: $0023 TMODH

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Reset:	1	1	1	1	1	1	1	1

Address: $0024 TMODL

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reset:	1	1	1	1	1	1	1	1

Figure 8-6. TIM Counter Modulo Registers (TMODH:TMODL)

NOTE

Reset the TIM counter before writing to the TIM counter modulo registers.

12.4 Functional Description

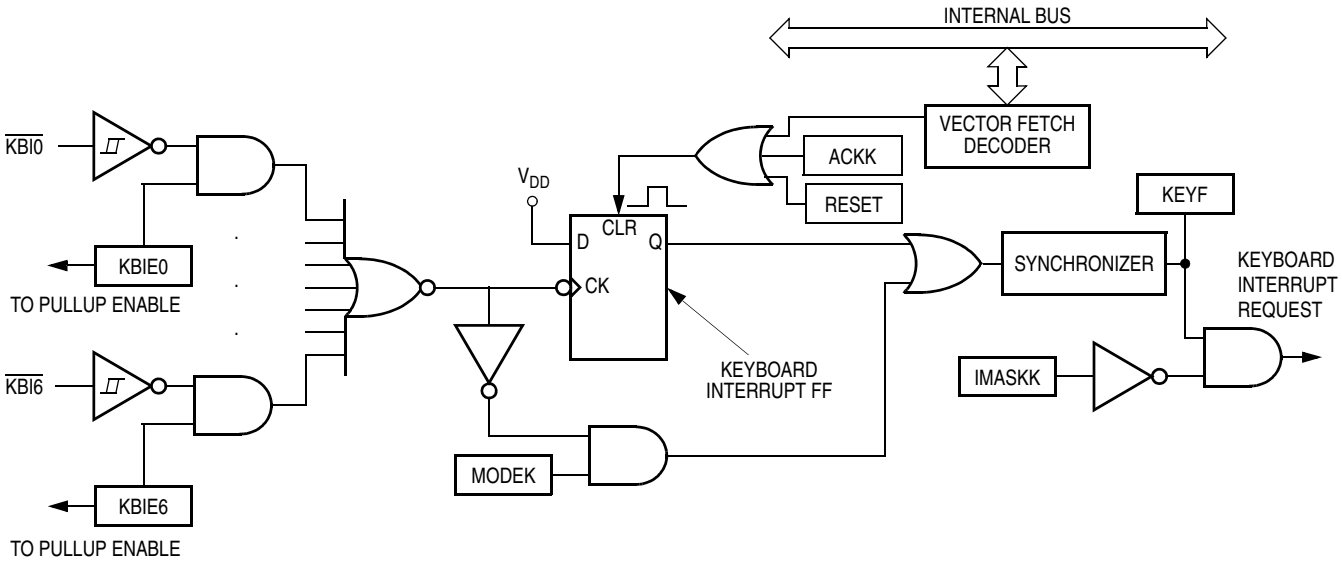


Figure 12-2. Keyboard Interrupt Block Diagram

Writing to the KBIE6–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port A pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin in port A also enables its internal pull-up device irrespective of PTAPUEx bits in the port A input pull-up enable register (see 10.2.3 Port A Input Pull-up Enable Register (PTAPUE)). A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

A keyboard interrupt is latched when one or more keyboard pins goes low after all were high. The MODEK bit in the keyboard status and control register controls the triggering mode of the keyboard interrupt.

- If the keyboard interrupt is edge-sensitive only, a falling edge on a keyboard pin does not latch an interrupt request if another keyboard pin is already low. To prevent losing an interrupt request on one pin because another pin is still low, software can disable the latter pin while it is low.
- If the keyboard interrupt is falling edge- and low level-sensitive, an interrupt request is present as long as any keyboard pin is low.

If the MODEK bit is set, the keyboard interrupt pins are both falling edge- and low level-sensitive, and both of the following actions must occur to clear a keyboard interrupt request:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a 1 to the ACKK bit in the keyboard status and control register KBSCR. The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations $FFE0 and $FFE1.
- Return of all enabled keyboard interrupt pins to logic 1 — As long as any enabled keyboard interrupt pin is at 0, the keyboard interrupt remains set.

15.4.2 Break Address Registers

The break address registers contain the high and low bytes of the desired breakpoint address. Reset clears the break address registers.

Address: $FE0C

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

Figure 15-4. Break Address Register High (BRKH)

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 15-5. Break Address Register Low (BRKL)

15.4.3 Break Status Register

The break status register 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

1. Writing a zero clears SBSW.

Figure 15-6. Break Status Register (BSR)

SBSW — SIM Break Stop/Wait

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.

1 = Wait mode was exited by break interrupt

0 = Wait mode was not exited by break interrupt

15.4.4 Break Flag Control Register (BFCR)

The break control register contains a bit that enables software to clear status bits while the MCU is in a break state.

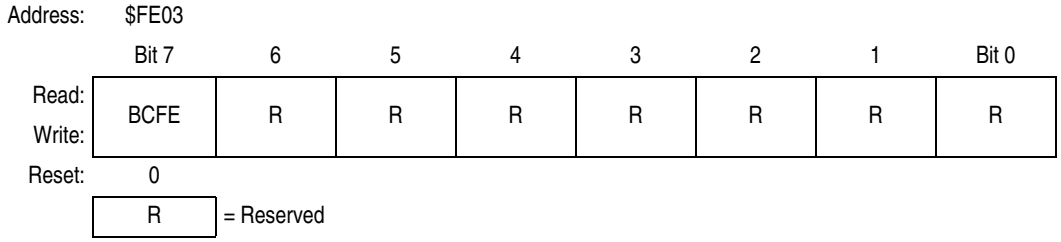


Figure 15-7. Break Flag Control Register (BFCR)

BCFE — Break Clear Flag Enable Bit

This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.

1 = Status bits clearable during break
0 = Status bits not clearable during break

15.5 Low-Power Modes

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

15.5.1 Wait Mode

If enabled, the break module is active in wait mode. In the break routine, the user can subtract one from the return address on the stack if SBSW is set (see 5.6 Low-Power Modes). Clear the SBSW bit by writing zero to it.

15.5.2 Stop Mode

A break interrupt causes exit from stop mode and sets the SBSW bit in the break status register. See 5.7 SIM Registers.

16.13 Memory Characteristics

Table 16-11. Memory Characteristics

Characteristic	Symbol	Min	Max	Unit
RAM data retention voltage	V_{RDR}	1.3	—	V
Flash program bus clock frequency	—	1	—	MHz
Flash read bus clock frequency	f_{Read} (1)	32k	8M	Hz
Flash page erase time	t_{Erase} (2)	1	—	ms
Flash mass erase time	t_{MErase} (3)	4	—	ms
Flash PGM/ERASE to HVEN set up time	t_{nvs}	10	—	µs
Flash high-voltage hold time	t_{nvh}	5	—	µs
Flash high-voltage hold time (mass erase)	t_{nvh1}	100	—	µs
Flash program hold time	t_{pgs}	5	—	µs
Flash program time	t_{PROG}	30	40	µs
Flash return to read time	t_{rcv} (4)	1	—	µs
Flash cumulative program hv period	t_{HV} (5)	—	4	ms
Flash row erase endurance (6)	—	10k	—	cycles
Flash row program endurance (7)	—	10k	—	cycles
Flash data retention time (8)	—	10	—	years

1. f_{Read} is defined as the frequency range for which the Flash memory can be read.
2. If the page erase time is longer than t_{Erase} (Min), there is no erase-disturb, but it reduces the endurance of the Flash memory.
3. If the mass erase time is longer than t_{MErase} (Min), there is no erase-disturb, but it reduces the endurance of the Flash memory.
4. t_{rcv} is defined as the time it needs before the Flash can be read after turning off the high voltage charge pump, by clearing HVEN to 0.
5. tHV is defined as the cumulative high voltage programming time to the same row before next erase. t_{HV} must satisfy this condition: $t_{nvs} + t_{nvh} + t_{pgs} + (t_{PROG} \times 32) \leq t_{HV}$ max.
6. The minimum row endurance value specifies each row of the Flash memory is guaranteed to work for at least this many erase / program cycles.
7. The minimum row endurance value specifies each row of the Flash memory is guaranteed to work for at least this many erase / program cycles.
8. The Flash is guaranteed to retain data over the entire operating temperature range for at least the minimum time specified.

B.4 Reserved Registers

The two registers at $FE08 and $FE09 are reserved locations on the MC68H(R)C08JL3E/JK3E.

On the MC68H(R)C908JL3E/JK3E, these two locations are the Flash control register and the Flash block protect register respectively.

B.5 Mask Option Registers

This section describes the mask option registers (MOR1 and MOR2). The mask option registers enable or disable the following options:

- Stop mode recovery time (32 × 2OSCOUT cycles or 4096 × 2OSCOUT cycles)
- STOP instruction
- Computer operating properly module (COP)
- COP reset period (COPRS), 8176 × 2OSCOUT or 262,128 × 2OSCOUT
- Enable LVI circuit
- Select LVI trip voltage

B.5.1 Functional Description

The mask options are hard-wired connections, specified at the same time as the ROM code, which allow the user to customize the MCU.

B.5.2 Mask Option Register 1 (MOR1)

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	COPRS	0	0	LVID	0	SSREC	STOP	COPD
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-1. Mask Option Register 1 (MOR1)

COPRS — COP reset period selection bit

1 = COP reset cycle is 8176 × 2OSCOUT
0 = COP reset cycle is 262,128 × 2OSCOUT

LVID — Low Voltage Inhibit Disable Bit

1 = Low Voltage Inhibit disabled
0 = Low Voltage Inhibit enabled

SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 × 2OSCOUT cycles instead of a 4096 × 2OSCOUT cycle delay.

1 = Stop mode recovery after 32 × 2OSCOUT cycles
0 = Stop mode recovery after 4096 × 2OSCOUT cycles

NOTE

Exiting stop mode by pulling reset will result in the long stop recovery.

If using an external crystal, do not set the SSREC bit.

STOP — STOP Instruction Enable

STOP enables the STOP instruction.

1 = STOP instruction enabled
0 = STOP instruction treated as illegal opcode

COPD — COP Disable Bit

COPD disables the COP module. (See **Chapter 13 Computer Operating Properly (COP)**.)

1 = COP module disabled
0 = COP module enabled

B.5.3 Mask Option Register 2 (MOR2)

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IRQPUD	0	0	LVIT1	LVIT0	0	0	0
Write:								
Reset:	0	0	0	Not affected	Not affected	0	0	0
POR:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-2. Mask Option Register 2 (MOR2)

IRQPUD — $\overline{\text{IRQ}}$ Pin Pull-up control bit

1 = Internal pull-up is disconnected
0 = Internal pull-up is connected between $\overline{\text{IRQ}}$ pin and V_{DD}

LVIT1, LVIT0 — Low Voltage Inhibit trip voltage selection bits

Detail description of the LVI control signals is given in Chapter 14 Low Voltage Inhibit (LVI)

B.6 Monitor ROM

The monitor program (monitor ROM: $FE10–$FFCF) on the MC68H(R)C08JL3E/JK3E is for device testing only. $FC00–$FDFF are unused.

Table B-3. DC Electrical Characteristics (3V)

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
V_{DD} supply current, f_{OP} = 2MHz					
Run[3]					
MC68HC08JL3E/JK3E		—	2.8	3.5	mA
MC68HRC08JL3E/JK3E		—	1.4	2	mA
Wait[4]					
MC68HC08JL3E/JK3E	I_{DD}	—	1.5	2	mA
MC68HRC08JL3E/JK3E		—	0.19	0.3	mA
Stop[5]					
(–40°C to 85°C)					
MC68HC08JL3E/JK3E		—	1.4	5	µA
MC68HRC08JL3E/JK3E		—	1.4	5	µA
Pullup resistors[6]					
PTD6, PTD7	R_{PU1}	1.8	4.3	4.8	kΩ
$\overline{RST}$, $\overline{IRQ}$, PTA0–PTA6	R_{PU2}	16	31	36	kΩ

1. V_{DD} = 2.7 to 3.3 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.
2. Typical values reflect average measurements at midpoint of voltage range, 25 °C only.
3. Run (operating) I_{DD} measured using external square wave clock source (f_{OP} = 2MHz). All inputs 0.2V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I_{DD}. Measured with all modules enabled.
4. Wait I_{DD} measured using external square wave clock source (f_{OP} = 2MHz). All inputs 0.2V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait I_{DD}.
5. Stop I_{DD} measured with OSC1 grounded; no port pins sourcing current. LVI is disabled.
6. R_{PU1} and R_{PU2} are measured at V_{DD} = 5.0V.

B.7.2 5V Oscillator Characteristics

Table B-4. Oscillator Component Specifications (5V)

Characteristic	Symbol	Min	Typ	Max	Unit
RC oscillator external R	R_{EXT}	See Figure B-3 and Figure B-4			
RC oscillator external C	C_{EXT}	—	10	—	pF