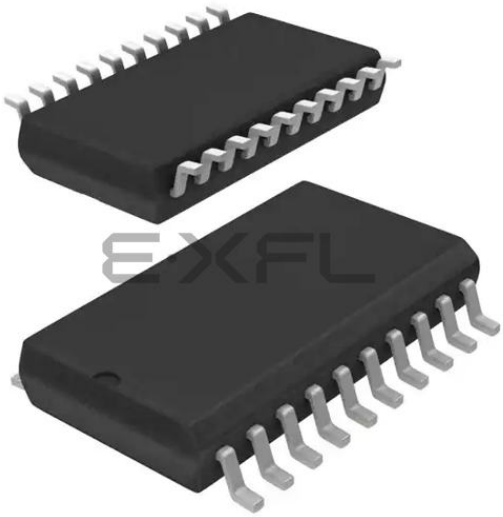


Welcome to E-XFL.COM

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	3MHz
Connectivity	USB
Peripherals	LVD, POR, PWM
Number of I/O	13
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908jb8jdwer

Figure	Title	Page

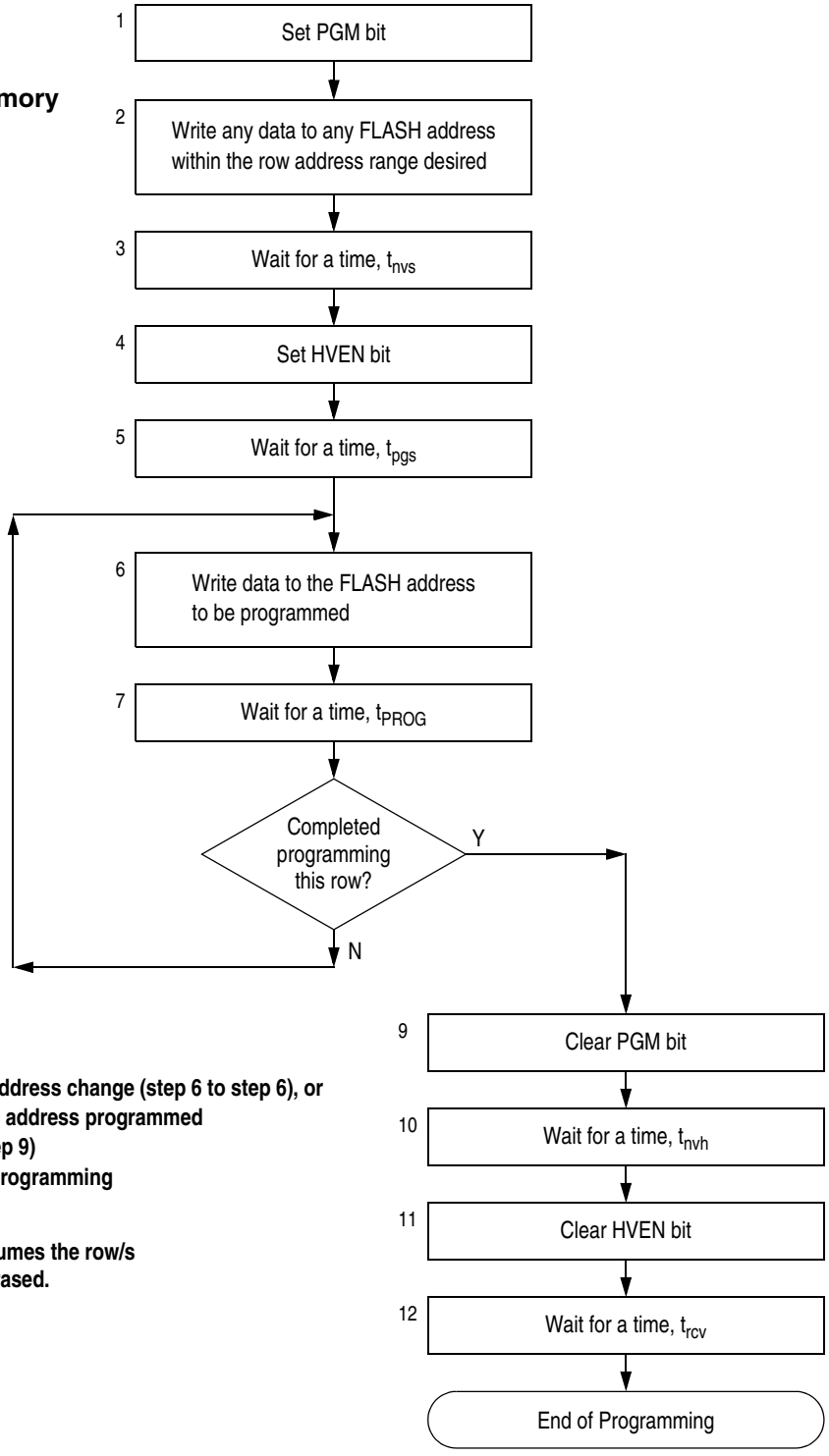
Features of the CPU08 include the following:

- 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

1.4 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908JB8.

Algorithm for programming a row (64 bytes) of FLASH memory



NOTE:

The time between each FLASH address change (step 6 to step 6), or the time between the last FLASH address programmed to clearing PGM bit (step 6 to step 9) must not exceed the maximum programming time, t_{PROG} max.

This row program algorithm assumes the row/s to be programmed are initially erased.

Figure 4-3. FLASH Programming Flowchart

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 X000, X200, X400, X600, X800, XA00, XC00, or XE00 within the FLASH memory.

Examples of protect start address:

BPR[7:0]	Start of Address of Protect Range
$00 to $DC	The entire FLASH memory is protected.
$DE (**1101 1110**)	$DE00 (**1101 1110** 0000 0000)
$E0 (**1110 0000**)	$E000 (**1110 0000** 0000 0000)
$E2 (**1110 0010**)	$E200 (**1110 0010** 0000 0000)
$E4 (**1110 0100**)	$E400 (**1110 0100** 0000 0000)
and so on...	
$FE	$FFE0–$FFFF (User vectors)
$FF	The entire FLASH memory is not protected.

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

4.9 ROM-Resident Routines

ROM-resident routines can be called by a program running in user mode or in monitor mode (see **Section 10. Monitor ROM (MON)**) for FLASH programming, erasing, and verifying. The range of the FLASH memory must be unprotected (see **4.8 FLASH Protection**) before calling the erase or programming routine.

Table 4-1. ROM-Resident Routines

Routine Name	Call Address	Routine Function
VERIFY	$FC03	FLASH verify routine
ERASE	$FC06	FLASH mass erase routine
PROGRAM	$FC09	FLASH program routine

8.4.2 Active Resets from Internal Sources

All internal reset sources actively pull the $\overline{\text{RST}}$ pin low for 32 OSCXCLK cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles. (See Figure 8-5.) An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, the USB module or POR. (See **Figure 8-6 . Sources of Internal Reset**.)

NOTE: *For LVI or POR resets, the SIM cycles through 4096 OSCXCLK cycles during which the SIM forces the $\overline{RST}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{RST}$ shown in* **Figure 8-5**.

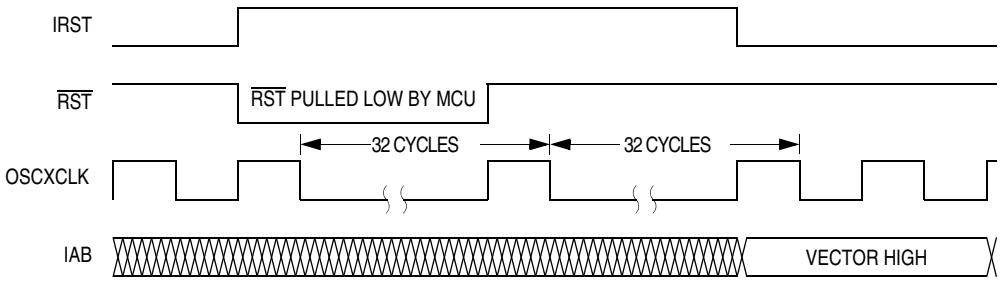


Figure 8-5. Internal Reset Timing

The COP reset is asynchronous to the bus clock.

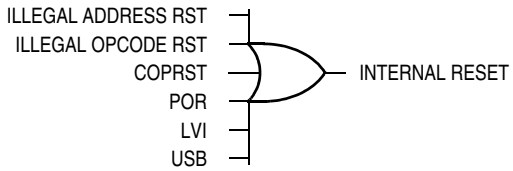


Figure 8-6. Sources of Internal Reset

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

8.7.2 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.

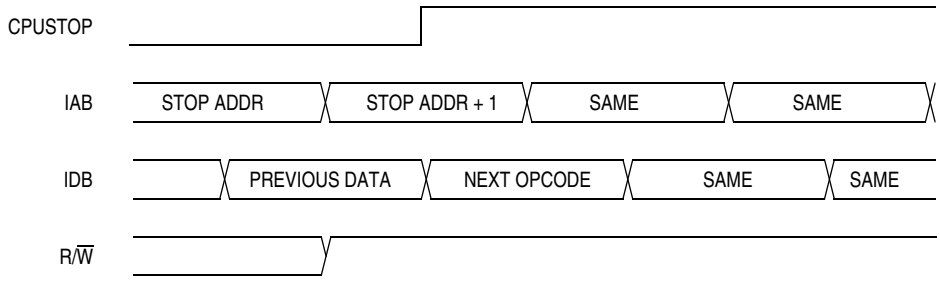
The SIM disables the oscillator signals (OSCOUT and OSCXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the configuration register (CONFIG). If SSREC is set, stop recovery is reduced from the normal delay of 4096 OSCXCLK cycles down to 2048. This is ideal for applications using canned oscillators that do not require long startup 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 8-16** 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 8-16. Stop Mode Entry Timing

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 0 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
```

8.8.2 Reset Status Register

This register contains seven flags that show the source of the last reset. All flag bits are cleared automatically following a read of the register. The register is initialized on power-up as shown with the POR bit set and all other bits cleared. However, during a POR or any other internal reset, the $\overline{\text{RST}}$ pin is pulled low. After the pin is released, it will be sampled 32 XCLK cycles later. If the pin is not above a V_{IH} at that time, then the PIN bit in the RSR may be set in addition to whatever other bits are set.

9.5.4 Resume After Suspend

The MCU can be activated from the suspend state by normal bus activity, a USB reset signal, or by a forced resume driven from the MCU.

9.5.4.1 Host Initiated Resume

The host signals resume by initiating resume signalling (K state) for at least 20ms followed by a standard low-speed EOP signal. This 20ms ensures that all devices in the USB network are awakened.

After resuming the bus, the host must begin sending bus traffic within 37ms to prevent the device from re-entering suspend mode.

9.5.4.2 USB Reset Signalling

Reset can wake a device from the suspended mode.

9.5.4.3 Remote Wakeup

The MCU also supports the remote wakeup feature. The firmware has the ability to exit suspend mode by signaling a resume state to the upstream host or hub. A non-idle state (K state) on the USB data lines is accomplished by asserting the FRESUM bit in the UCR1 register.

When using the remote wakeup capability, the firmware must wait for at least 5ms after the bus is in the idle state before sending the remote wakeup resume signaling. This allows the upstream devices to get into their suspend state and prepare for propagating resume signaling. The FRESUM bit should be asserted to cause the resume state on the USB data lines for at least 10ms, but not more than 15ms. Note that the resume signalling is controlled by the FRESUM bit and meeting the timing specifications is dependent on the firmware. When FRESUM is cleared by firmware, the data lines will return to their high-impedance state.
Refer to the register definitions (see **9.8.6 USB Control Register 1**) for more information about how the force resume (FRESUM) bit can be used to initiate the remote wakeup feature.

STALL2 — Endpoint 2 Force Stall Bit

This read/write bit causes endpoint 2 to return a STALL handshake when polled by either an IN or OUT token by the USB host controller. Reset clears this bit.

1 = Send STALL handshake
0 = Default

TX2E — Endpoint 2 Transmit Enable

This read/write bit enables a transmit to occur when the USB host controller sends an IN token to endpoint 2. The appropriate endpoint enable bit, ENABLE2 bit in the UCR3 register, also should be set. Software should set the TX2E bit when data is ready to be transmitted. It must be cleared by software when no more data needs to be transmitted.

If this bit is 0 or the TXD2F is set, the USB will respond with a NAK handshake to any endpoint 2 directed IN tokens. Reset clears this bit.

1 = Data is ready to be sent
0 = Data is not ready. Respond with NAK

RX2E — Endpoint 2 Receive Enable

This read/write bit enables a receive to occur when the USB host controller sends an OUT token to endpoint 2. Software should set this bit when data is ready to be received. It must be cleared by software when data cannot be received.

If this bit is 0 or the RXD2F is set, the USB will respond with a NAK handshake to any endpoint 2 OUT tokens. Reset clears this bit.

1 = Data is ready to be received
0 = Not ready for data. Respond with NAK

TP2SIZ3–TP2SIZ0 — Endpoint 2 Transmit Data Packet Size

These read/write bits store the number of transmit data bytes for the next IN token request for endpoint 2. These bits are cleared by reset.

9.8.12 USB Endpoint 0 Data Registers

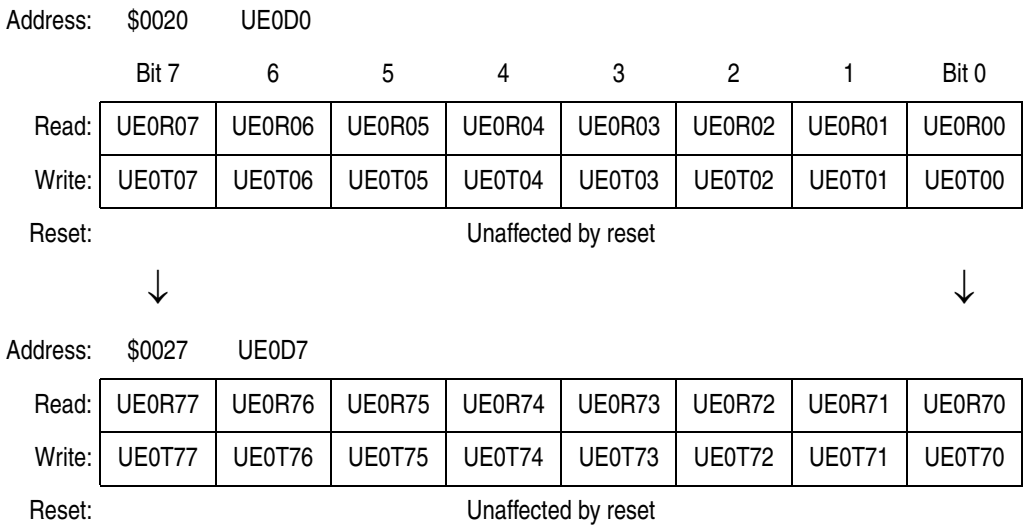


Figure 9-26. USB Endpoint 0 Data Registers (UE0D0–UE0D7)

UE0Rx7–UE0Rx0 — Endpoint 0 Receive Data Buffer

These read-only bits are serially loaded with OUT token or SETUP token data directed at endpoint 0. The data is received over the USB's D+ and D– pins.

UE0Tx7–UE0Tx0 — Endpoint 0 Transmit Data Buffer

These write-only buffers are loaded by software with data to be sent on the USB bus on the next IN token directed at endpoint 0.

9.8.14 USB Endpoint 2 Data Registers

Address:	$0030	UE2D0						
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	UE2R07	UE2R06	UE2R05	UE2R04	UE2R03	UE2R02	UE2R01	UE2R00
Write:	UE2T07	UE2T06	UE2T05	UE2T04	UE2T03	UE2T02	UE2T01	UE2T00
Reset:	Unaffected by reset							

Address:	$0037	UE2D7						
Read:	UE2R77	UE2R76	UE2R75	UE2R74	UE2R73	UE2R72	UE2R71	UE2R70
Write:	UE2T77	UE2T76	UE2T75	UE2T74	UE2T73	UE2T72	UE2T71	UE2T70
Reset:	Unaffected by reset							

Figure 9-28. USB Endpoint 2 Data Registers (UE2D0–UE2D7)

UE2Rx7–UE2Rx0 — Endpoint 2 Receive Data Buffer

These read-only bits are serially loaded with OUT token data directed at endpoint 2. The data is received over the USB's D+ and D– pins.

UE2Tx7–UE2Tx0 — Endpoint 2 Transmit Data Buffer

These write-only buffers are loaded by software with data to be sent on the USB bus on the next IN token directed at endpoint 2.

10.5 Security

A security feature discourages unauthorized reading of FLASH locations while in monitor mode. The host can bypass the security feature at monitor mode entry by sending eight security bytes that match the bytes at locations $FFF6–$FFFD. Locations $FFF6–$FFFD contain user-defined data.

NOTE: *Do not leave locations $FFF6–$FFFD blank. For security reasons, program locations $FFF6–$FFFD even if they are not used for vectors.*

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. (See **Figure 10-7**.)

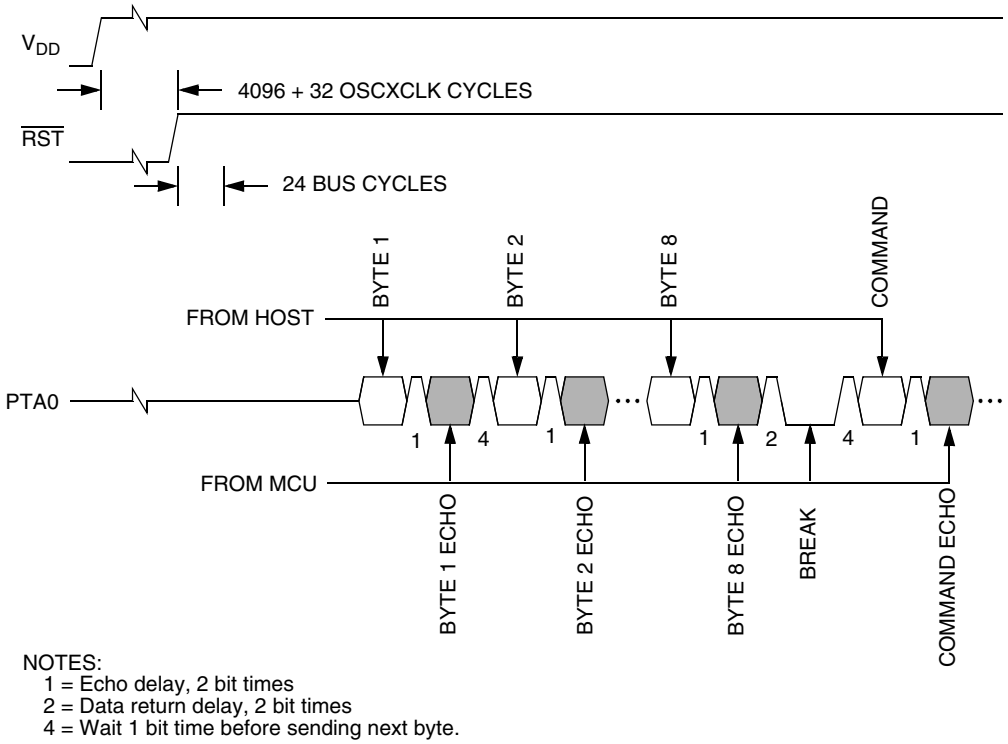


Figure 10-7. Monitor Mode Entry Timing

12.3 Port A

Port A is an 8-bit general-purpose bidirectional I/O port with software configurable pullups, and it shares its pins with the keyboard interrupt module (KBI).

12.3.1 Port A Data Register

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

Address: $0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
Reset:	Unaffected by reset							
Alternative Function:	$\overline{KBA7}$	$\overline{KBA6}$	$\overline{KBA5}$	$\overline{KBA4}$	$\overline{KBA3}$	$\overline{KBA2}$	$\overline{KBA1}$	$\overline{KBA0}$
Additional Function:	Optional pullup	Optional pullup	Optional pullup	Optional pullup	Optional pullup	Optional pullup	Optional pullup	Optional pullup

Figure 12-2. Port A Data Register (PTA)

PTA[7:0] — Port A Data Bits

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

The port A pullup enable bit, PAP, in the port option control register (POCR) enables pullups on port A pins if the respective pin is configured as an input. (See **12.8 Port Options**.)

$\overline{KBA7}$–$\overline{KBA0}$ — Keyboard Interrupts

The keyboard interrupt enable bits, KBIE7–KBIE0, in the keyboard interrupt enable register (KBIER), enable the port A pins as external interrupt pins. (See **Section 14. Keyboard Interrupt Module (KBI)**.)

13.5 $\overline{\text{IRQ}}$ Pin

The $\overline{\text{IRQ}}$ pin has a low leakage for input voltages ranging from 0V to V_{DD}; suitable for applications using RC discharge circuitry to wake up the MCU.

A logic 0 on the $\overline{\text{IRQ}}$ pin can latch an interrupt request into the IRQ latch. A vector fetch, software clear, or reset clears the IRQ latch.

If the MODE bit is set, the $\overline{\text{IRQ}}$ pin is both falling-edge-sensitive and low-level-sensitive. With MODE set, both of the following actions must occur to clear IRQ:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACK bit in the interrupt status and control register (ISCR). The ACK bit is useful in applications that poll the $\overline{\text{IRQ}}$ pin and require software to clear the IRQ latch. Writing to the ACK bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACK does not affect subsequent transitions on the $\overline{\text{IRQ}}$ pin. A falling edge that occurs after writing to the ACK bit latches another interrupt request. If the IRQ mask bit, IMASK, is clear, the CPU loads the program counter with the vector address at locations $FFF8 and $FFF9.
- Return of the $\overline{\text{IRQ}}$ pin to logic one — As long as the $\overline{\text{IRQ}}$ pin is at logic zero, IRQ remains active.

The vector fetch or software clear and the return of the $\overline{\text{IRQ}}$ pin to logic one may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ}}$ pin is at logic zero. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

If the MODE bit is clear, the $\overline{\text{IRQ}}$ pin is falling-edge-sensitive only. With MODE clear, a vector fetch or software clear immediately clears the IRQ latch.

The IRQF bit in the ISCR register can be used to check for pending interrupts. The IRQF bit is not affected by the IMASK bit, which makes it useful in applications where polling is preferred.

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 data direction register A.
2. Write logic 1s to the appropriate port A data register bits.
3. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

14.7 Low-Power Modes

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

14.7.1 Wait Mode

The keyboard module remains active in wait mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of wait mode.

14.7.2 Stop Mode

The keyboard module remains active in stop mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of stop mode.

Address: $0016

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

= Unimplemented

Figure 14-2. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

These read-only bits always read as logic 0s.

KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending. Reset clears the KEYF bit.

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

ACKK — Keyboard Acknowledge Bit

Writing a logic 1 to this write-only bit clears the keyboard interrupt request. ACKK always reads as logic 0. Reset clears ACKK.

IMASKK — Keyboard Interrupt Mask Bit

Writing a logic 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests. Reset clears the IMASKK bit.

1 = Keyboard interrupt requests masked
0 = Keyboard interrupt requests not masked

MODEK — Keyboard Triggering Sensitivity Bit

This read/write bit controls the triggering sensitivity of the keyboard interrupt pins. Reset clears MODEK.

1 = Keyboard interrupt requests on falling edges and low levels
0 = Keyboard interrupt requests on falling edges only

Appendix B. MC68HC08JT8

B.1 Contents

B.11.2 Functional Operating Range

Characteristic	Symbol	Value	Unit
Operating temperature range	T_A	0 to 70	°C
Operating voltage range	V_{DD}	2.0 to 3.6	V

B.11.3 DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage (I_{Load} = −1.6 mA) PTA0–PTA7, PTB0–PTB7, PTC0–PTC7, PTE0–PTE2	V_{OH}	V_{DD}–0.4	—	—	V
Output low voltage (I_{Load} = 1.6 mA) All I/O pins (I_{Load} = 15 mA) PTD0–PTD1 in ILDD mode (I_{Load} = 5 mA) PTE3–PTE4	V_{OL}	 — — —	 — — —	 0.4 0.5 0.4	V
Input high voltage All ports, OSC1, $\overline{IRQ}$, $\overline{RST}$	V_{IH}	0.7 × V_{DD}	—	V_{DD}	V
Input low voltage All ports, OSC1, $\overline{IRQ}$, $\overline{RST}$	V_{IL}	V_{SS}	—	0.3 × V_{DD}	V
Output low current (V_{OL} = 2.0 V)[3] PTD2–PTD5 in LDD mode (V_{DD} = 2V) PTD2–PTD5 in LDD mode (V_{DD} = 3V)	I_{OL}	 — —	 6 16	 — —	mA
V_{DD} supply current, V_{DD} = 3V, f_{OP} = 3MHz Run[4] Wait[5] Stop[6] 0 °C to 70°C	I_{DD}	 — — —	 3.5 2.5 20	 6.5 4.5 30	 mA mA µA
I/O ports Hi-Z leakage current	I_{IL}	—	—	± 10	µA
Input current	I_{IN}	—	—	± 1	µA
Capacitance Ports (as input or output)	C_{Out} C_{In}	— —	— —	12 8	pF
POR re-arm voltage[7]	V_{POR}	0	—	100	mV
POR rise-time ramp rate	R_{POR}	0.02	—	—	V/ms
Monitor mode entry voltage	V_{DD}+V_{HI}	1.4 × V_{DD}		2 × V_{DD}	V
Pullup resistors Port A, port B, port C, PTE0–PTE2, $\overline{RST}$, $\overline{IRQ}$ PTE3–PTE4	R_{PU}	 25 4	 40 5	 55 6	 kΩ kΩ