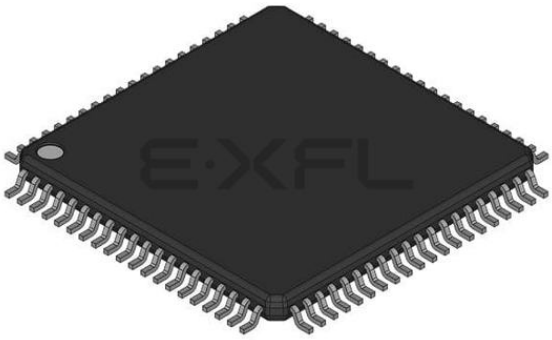

Welcome to E-XFL.COM

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	Active
Core Processor	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	I²C, IRSCI, SPI
Peripherals	LCD, LVD, POR, PWM
Number of I/O	40
Program Memory Size	24KB (24K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-LQFP
Supplier Device Package	80-FQFP (12x12)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68hc908lk24cpk

Revision History

Date	Revision Level	Description	Page Number(s)
8/2003	2	First general release.	—

Section 12. Real Time Clock (RTC)

Section 1. General Description

1.1 Contents

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0058	LCD Data Register 7 (LDAT7)	Read: Write:	F13B3	F13B2	F13B1	F13B0	F12B3	F12B2	F12B1	F12B0
		Reset:	U	U	U	U	U	U	U	U
$0059	LCD Data Register 8 (LDAT8)	Read: Write:	F15B3	F15B2	F15B1	F15B0	F14B3	F14B2	F14B1	F14B0
		Reset:	U	U	U	U	U	U	U	U
$005A	LCD Data Register 9 (LDAT9)	Read: Write:	F17B3	F17B2	F17B1	F17B0	F16B3	F16B2	F16B1	F16B0
		Reset:	U	U	U	U	U	U	U	U
$005B	LCD Data Register 10 (LDAT10)	Read: Write:	F19B3	F19B2	F19B1	F19B0	F18B3	F18B2	F18B1	F18B0
		Reset:	U	U	U	U	U	U	U	U
$005C	LCD Data Register 11 (LDAT11)	Read: Write:	F21B3	F21B2	F21B1	F21B0	F20B3	F20B2	F20B1	F20B0
		Reset:	U	U	U	U	U	U	U	U
$005D	LCD Data Register 12 (LDAT12)	Read: Write:	F23B3	F23B2	F23B1	F23B0	F22B3	F22B2	F22B1	F22B0
		Reset:	U	U	U	U	U	U	U	U
$005E	LCD Data Register 13 (LDAT13)	Read: Write:	F25B3	F25B2	F25B1	F25B0	F24B3	F24B2	F24B1	F24B0
		Reset:	U	U	U	U	U	U	U	U
$005F	LCD Data Register 14 (LDAT14)	Read: Write:	F27B3	F27B2	F27B1	F27B0	F26B3	F26B2	F26B1	F26B0
		Reset:	U	U	U	U	U	U	U	U
$0060	LCD Data Register 15 (LDAT15)	Read: Write:	F29B3	F29B2	F29B1	F29B0	F28B3	F28B2	F28B1	F28B0
		Reset:	U	U	U	U	U	U	U	U
$0061	LCD Data Register 16 (LDAT16)	Read: Write:	F31B3	F31B2	F31B1	F31B0	F30B3	F30B2	F30B1	F30B0
		Reset:	U	U	U	U	U	U	U	U

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 10 of 13)

4.8 FLASH Block 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 pages 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.

NOTE: *The 48 bytes of user interrupt vectors are always protected, regardless of the value in the FLASH block protect register. A mass erase is required to erase the vectors.*

When the FLBPR is program with $20, the entire memory is protected from being programmed and erased. When the FLBPR is erased ($FF), the entire memory is accessible for program and erase.

Once the FLBPR is programmed with a value other than $FF, the FLBPR itself is protected. It can only be erased using a mass erase operation.

NOTE: *In performing a program or erase operation, the FLASH block protect register must be read after setting the PGM or ERASE bit and before asserting the HVEN bit*

6.6.2 Stop Mode

The STOP instruction:

- Clears the interrupt mask (I bit) in the condition code register, enabling external interrupts. After exit from stop mode by external interrupt, the I bit remains clear. After exit by reset, the I bit is set.
- Disables the CPU clock.

After exiting stop mode, the CPU clock begins running after the oscillator stabilization delay.

6.7 CPU During Break Interrupts

If the break module is enabled, a break interrupt causes the CPU to execute the software interrupt instruction (SWI) at the completion of the current CPU instruction. (See **Section 23. Break Module (BRK)**.) The program counter vectors to $FFFC–$FFFD ($FEFC–$FEFD in monitor mode).

A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation if the break interrupt has been deasserted.

6.8 Instruction Set Summary

Table 6-1 provides a summary of the M68HC08 instruction set.

6.9 Opcode Map

The opcode map is provided in **Table 6-2**.

PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). (See **8.4.8 Base Clock Selector Circuit**.) Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the oscillator output, CGMXCLK, or the divided VCO clock, CGMPCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMPCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. (See **8.4.8 Base Clock Selector Circuit**.) Reset clears the BCS bit.

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

NOTE: *PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMPCLK requires two writes to the PLL control register. (See **8.4.8 Base Clock Selector Circuit**.)*

PRE1 and PRE0 — Prescaler Program Bits

These read/write bits control a prescaler that selects the prescaler power-of-two multiplier, P. (See **8.4.3 PLL Circuits** and **8.4.6 Programming the PLL**.) PRE1 and PRE0 cannot be written when the PLLON bit is set. Reset clears these bits.

These prescaler bits affects the relationship between the VCO clock and the final system bus clock.

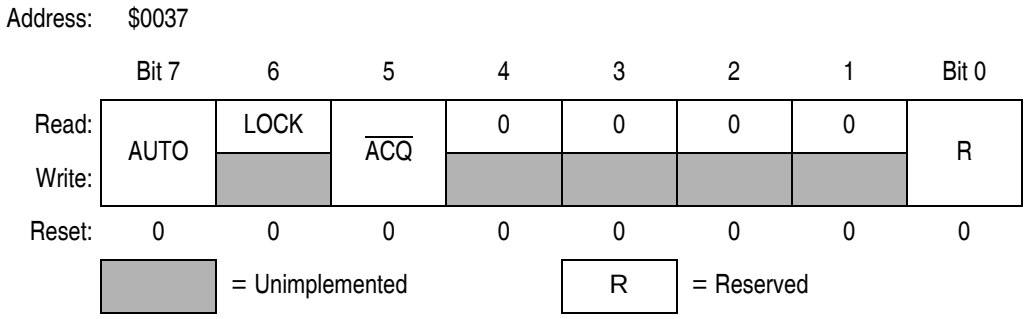

Figure 8-5. PLL Bandwidth Control Register (PBWCR)

AUTO — Automatic Bandwidth Control Bit

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

1 = Automatic bandwidth control
0 = Manual bandwidth control

LOCK — Lock Indicator Bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as logic 0 and has no meaning. The write one function of this bit is reserved for test, so this bit must *always* be written a 0. Reset clears the LOCK bit.

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

$\overline{ACQ}$ — Acquisition Mode Bit

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

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

1 = Tracking mode
0 = Acquisition mode

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							Note	
		Reset:							0	
Note: Writing a logic 0 clears SBSW.										
$FE01	SIM Reset Status Register (SRSR)	Read:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	0	0
$FE03	SIM Break Flag Control Register (SBFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE04	Interrupt Status Register 1 (INT1)	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
$FE05	Interrupt Status Register 2 (INT2)	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
$FE06	Interrupt Status Register 3 (INT3)	Read:	0	0	0	0	IF18	IF17	IF16	IF15
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
			= Unimplemented				R	= Reserved		

Figure 9-2. SIM I/O Register Summary

9.3 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in **Figure 9-3**. This clock can come from either the oscillator module or from the on-chip PLL. (See **Section 8. Clock Generator Module (CGM)**.)

9.6 Exception Control

Normal, sequential program execution can be changed in three different ways:

- Interrupts:
  - Maskable hardware CPU interrupts
  - Non-maskable software interrupt instruction (SWI)
- Reset
- Break interrupts

9.6.1 Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. **Figure 9-8** shows interrupt entry timing, and **Figure 9-9** shows interrupt recovery timing.

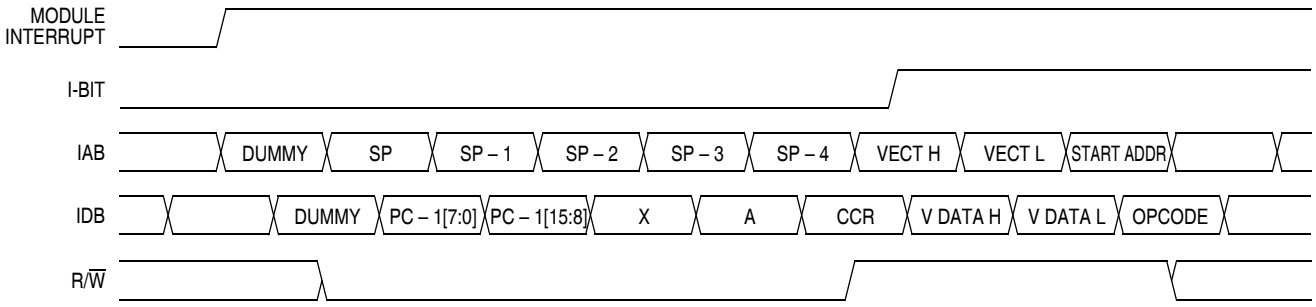

Figure 9-8. Interrupt Entry Timing

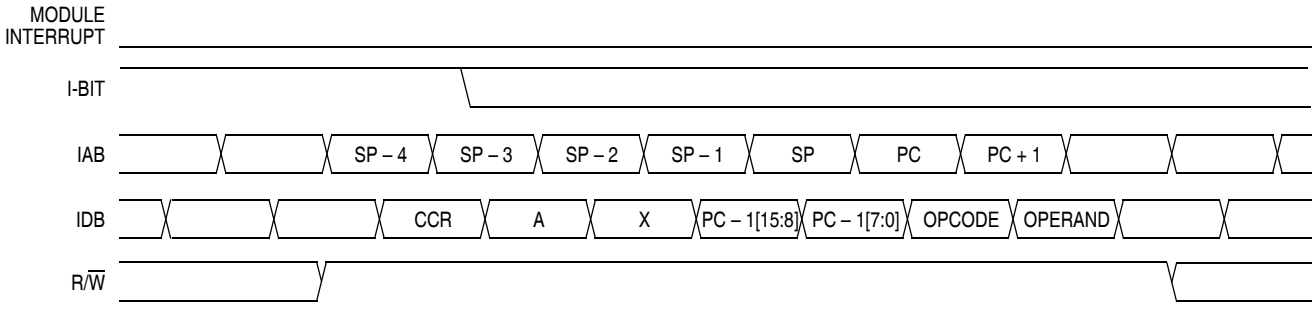

Figure 9-9. Interrupt Recovery Timing

9.6.1.1 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 9-11** 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.

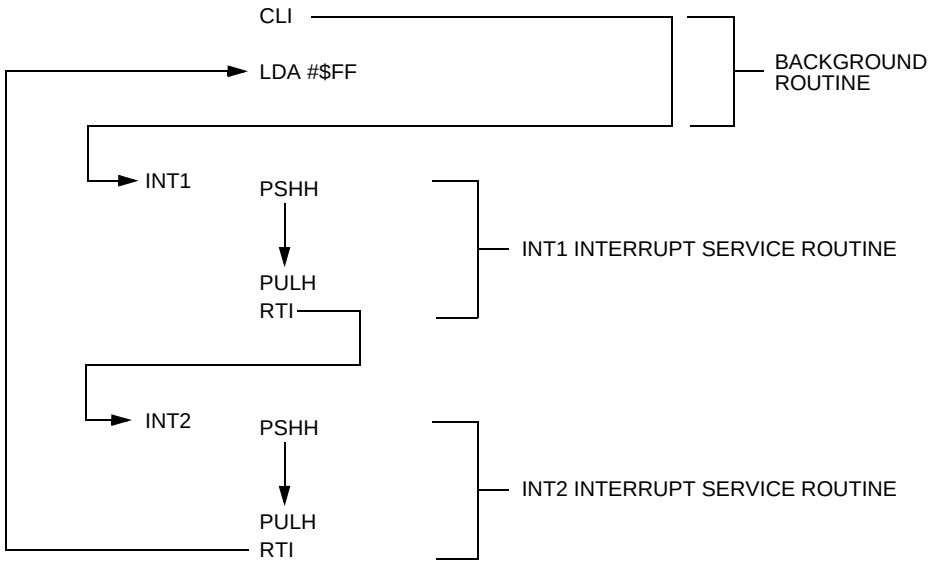

Figure 9-11. Interrupt Recognition Example

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.*

Section 10. Monitor ROM (MON)

10.1 Contents

10.4 Functional Description

The monitor ROM receives and executes commands from a host computer. **Figure 10-1** shows an example circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

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

The monitor code allows enabling the PLL to generate the internal clock, provided the reset vector is blank, when the device is being clocked by a low-frequency crystal. This entry method, which is enabled when $\overline{\text{IRQ}}$ is held low out of reset, is intended to support serial communication/ programming at 9600 baud in monitor mode by stepping up the external frequency (assumed to be 32.768 kHz) by a fixed amount to generate the desired internal frequency (2.4576 MHz). Since this feature is enabled only when $\overline{\text{IRQ}}$ is held low out of reset, it cannot be used when the reset vector is non-zero because entry into monitor mode in this case requires V_{TST} on $\overline{\text{IRQ}}$.

14.7 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the transmit data register only when the SPTE bit is high. **Figure 14-8** shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

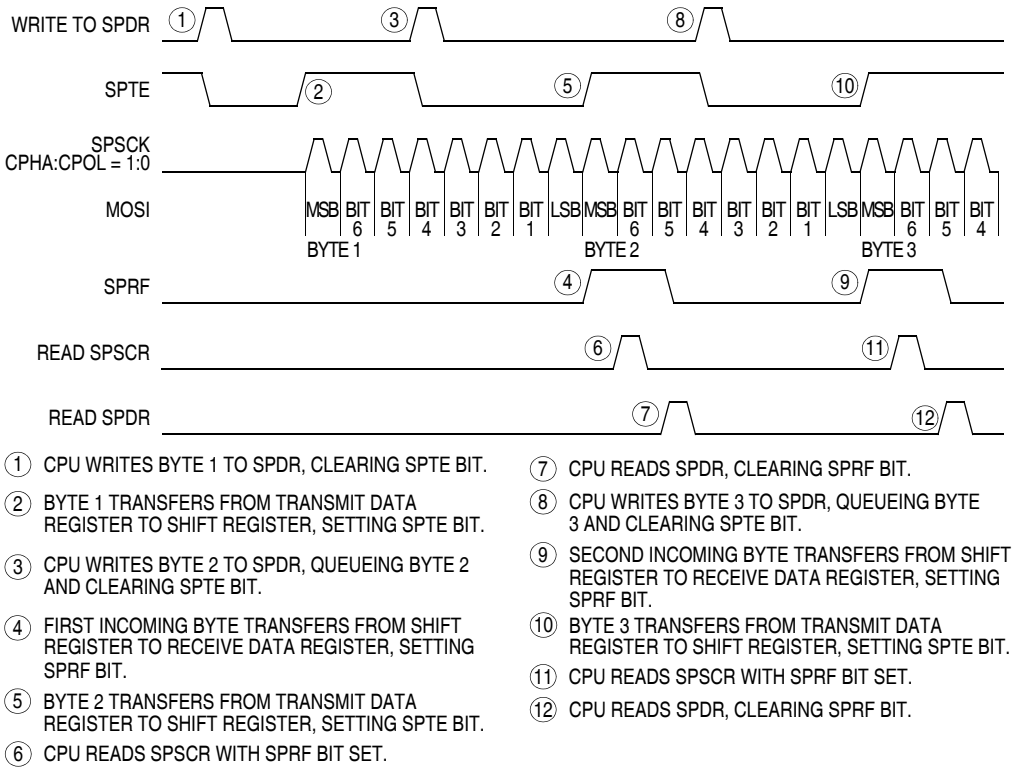

Figure 14-8. SPRF/SPTE CPU Interrupt Timing

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.

Table 18-1. Port Control Register Bits Summary (Sheet 2 of 2)

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
D[1]	0	DDRD0	SPI RTC	SPCR ($0010)	SPE	PTD0/$\overline{SS}$/CALIN
				RTCCOMR ($0040)	CAL	
	1	DDRD1		SPCR ($0010)	SPE	PTD1/MISO
	2	DDRD2				PTD2/MOSI
	3	DDRD3		SPCR ($0010)	SPE	PTD3/SPSCK/CALOUT
				RTCCOMR ($0040)	CAL	
	4	DDRD4	KBI TIM	KBIER ($001C)	KBIE4	PTD4/KBI4/T1CLK
				T1SC ($0020)	PS[2:0]	
	5	DDRD5		KBIER ($001C)	KBIE5	PTD5/KBI5/T2CLK
				T2SC ($002B)	PS[2:0]	
	6	DDRD6	KBI MMIIC	KBIER ($001C)	KBIE6	PTD6/KBI6/SCL
				MMCR ($006C)	MMEN	
	7	DDRD7		KBIER ($001C)	KBIE7	PTD7/KBI7/SDA
				MMCR ($006C)	MMEN	
E	0	DDRE0	LCD	CONFIG2 ($001D)	PEE	PTE0/FP11
	1	DDRE1				PTE1/FP12
	2	DDRE2				PTE2/FP13
	3	DDRE3				PTE3/FP14
	4	DDRE4				PTE4/FP15
	5	DDRE5				PTE5/FP16
	6	DDRE6				PTE6/FP17
	7	DDRE7				PTE7/FP18
F	0	DDRF0	—	—	—	PTF0
	1	DDRF1				PTF1
	2	DDRF2				PTF2
	3	DDRF3				PTF3
	4	DDRF4				PTF4
	5	DDRF5				PTF5
	6	DDRF6				PTF6
	7	DDRF7				PTF7

Notes:
1. In addition to the standard I/O function on PTD0 and PTD3–PTD7 pins, these pins are shared with two other modules. For each of the pins, ONLY enable ONE module at any one time to avoid pin contention.

NOTE: *Setting a keyboard interrupt enable bit (KBIEx) 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.*

20.5.1 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup 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 KBIEx 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 DDR bits in data direction register.
2. Write logic 1s to the appropriate data register bits.
3. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

Section 21. Computer Operating Properly (COP)

21.1 Contents

21.2 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. The COP module can be disabled through the COPD bit in the configuration register 1 (CONFIG1).

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE0F	Low-Voltage Inhibit Status Register (LVISR)	Read:	LVIOUT	LVIIE	LVIIF	0	0	0	0	0
		Write:				LVIIACK				
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 22-1. LVI I/O Register Summary

22.4 Functional Description

Figure 22-2 shows the structure of the LVI module.

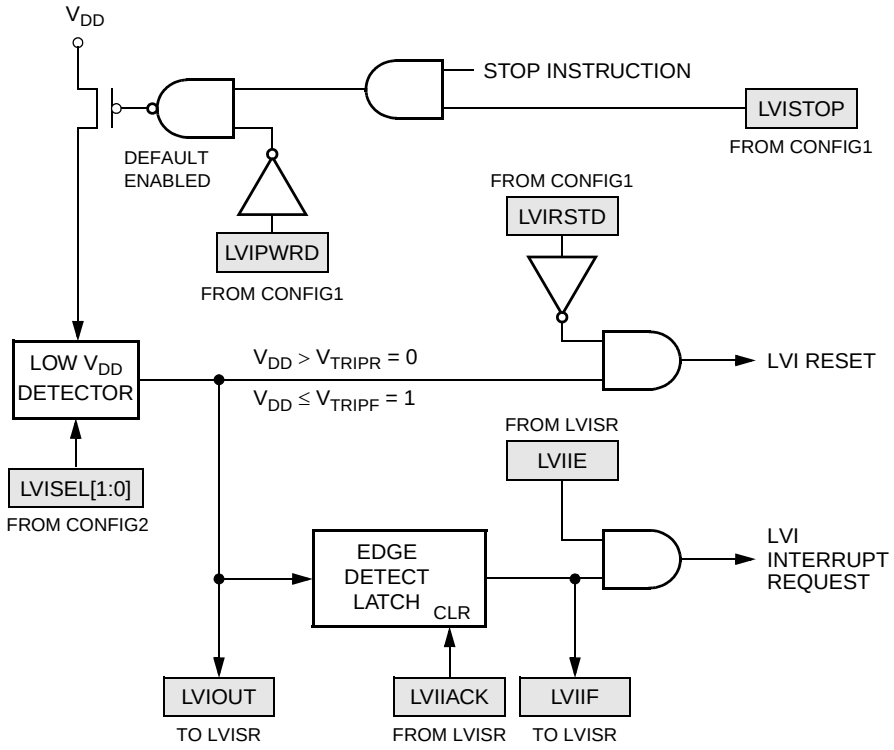

Figure 22-2. LVI Module Block Diagram

The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. Clearing the LVI power disable bit, LVIPWRD, enables the LVI to monitor V_{DD} voltage. Clearing the LVI reset disable bit, LVIRSTD, enables the LVI module to generate a reset when V_{DD} falls below a voltage, V_{TRIPF}. Setting the LVI enable in stop mode bit, LVISTOP, enables the LVI to operate in stop mode.

BRKA — Break Active Bit

This read/write status and control bit is set when a break address match occurs. Writing a logic 1 to BRKA generates a break interrupt. Clear BRKA by writing a logic 0 to it before exiting the break routine. Reset clears the BRKA bit.

1 = (When read) Break address match

0 = (When read) No break address match

23.6.2 Break Address Registers

The break address registers (BRKH and BRKL) 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 23-4. 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 23-5. Break Address Register Low (BRKL)

23.6.3 SIM Break Status Register

The SIM break status register (SBSR) contains a flag to indicate that a break caused an exit from wait mode. The flag is useful in applications requiring a return to wait mode after exiting from a break interrupt.