



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	Not For New Designs
Core Processor	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	-
Peripherals	LVD, POR, PWM
Number of I/O	13
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	16-SOIC (0.295", 7.50mm Width)
Supplier Device Package	16-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908qy4amdwer

MC68HC908QY4A	MC68HC908QT4A
MC68HC908QY2A	MC68HC908QT2A
MC68HC908QY1A	MC68HC908QT1A

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://freescale.com/>

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
This product incorporates SuperFlash® technology licensed from SST.

© Freescale Semiconductor, Inc., 2007–2010. All rights reserved.

Chapter 11

Oscillator (OSC) Module

11.1	Introduction	93
11.2	Features	93
11.3	Functional Description	93
11.3.1	Internal Signal Definitions	94
11.3.1.1	Oscillator Enable Signal (SIMOSCEN)	94
11.3.1.2	XTAL Oscillator Clock (XTALCLK)	95
11.3.1.3	RC Oscillator Clock (RCCLK)	95
11.3.1.4	Internal Oscillator Clock (INTCLK)	95
11.3.1.5	Bus Clock Times 4 (BUSCLKX4)	95
11.3.1.6	Bus Clock Times 2 (BUSCLKX2)	95
11.3.2	Internal Oscillator	95
11.3.2.1	Internal Oscillator Trimming	95
11.3.2.2	Internal to External Clock Switching	96
11.3.2.3	External to Internal Clock Switching	96
11.3.3	External Oscillator	96
11.3.4	XTAL Oscillator	96
11.3.5	RC Oscillator	98
11.4	Interrupts	98
11.5	Low-Power Modes	98
11.5.1	Wait Mode	98
11.5.2	Stop Mode	98
11.6	OSC During Break Interrupts	99
11.7	I/O Signals	99
11.7.1	Oscillator Input Pin (OSC1)	99
11.7.2	Oscillator Output Pin (OSC2)	99
11.8	Registers	100
11.8.1	Oscillator Status and Control Register	100
11.8.2	Oscillator Trim Register (OSCTRIM)	101

Chapter 12

Input/Output Ports (PORTS)

12.1	Introduction	103
12.2	Unused Pin Termination	103
12.3	Port A	103
12.3.1	Port A Data Register	104
12.3.2	Data Direction Register A	104
12.3.3	Port A Input Pullup Enable Register	105
12.3.4	Port A Summary Table	106
12.4	Port B	106
12.4.1	Port B Data Register	106
12.4.2	Data Direction Register B	107
12.4.3	Port B Input Pullup Enable Register	108
12.4.4	Port B Summary Table	108

Chapter 1

General Description

1.1 Introduction

The MC68HC908QY4A is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

Table 1-1. Summary of Device Variations

Device	FLASH Memory Size	ADC	Pin Count
MC68HC908QT1A	1536 bytes	—	8 pins
MC68HC908QT2A	1536 bytes	6 channel, 10 bit	8 pins
MC68HC908QT4A	4096 bytes	6 channel, 10 bit	8 pins
MC68HC908QY1A	1536 bytes	—	16 pins
MC68HC908QY2A	1536 bytes	6 channel, 10 bit	16 pins
MC68HC908QY4A	4096 bytes	6 channel, 10 bit	16 pins

1.2 Features

Features include:

- High-performance M68HC08 CPU core
- Fully upward-compatible object code with M68HC05 Family
- 5-V and 3-V operating voltages (V_{DD})
- 8-MHz internal bus operation at 5 V, 4-MHz at 3 V
- Trimmable internal oscillator
 - Software selectable 1 MHz, 2 MHz, or 3.2 MHz internal bus operation
 - 8-bit trim capability
 - $\pm 25\%$ untrimmed
 - Trimmable to approximately 0.4%⁽¹⁾
- Software selectable crystal oscillator range, 32–100 kHz, 1–8 MHz and 8–32 MHz
- Software configurable input clock from either internal or external source
- Auto wakeup from STOP capability using dedicated internal 32-kHz RC or bus clock source
- On-chip in-application programmable FLASH memory
 - Internal program/erase voltage generation
 - Monitor ROM containing user callable program/erase routines
 - FLASH security⁽²⁾

1. See [16.11 Oscillator Characteristics](#) for internal oscillator specifications

2. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

1.4 Pin Assignments

The MC68HC908QT4A, MC68H908QT2A, and MC68HC098QT1A are available in 8-pin packages. The MC68HC908QY4A, MC68HC908QY2A, and MC68HC908QY1A are available in 16-pin packages.

Figure 1-2 shows the pin assignment for these packages.

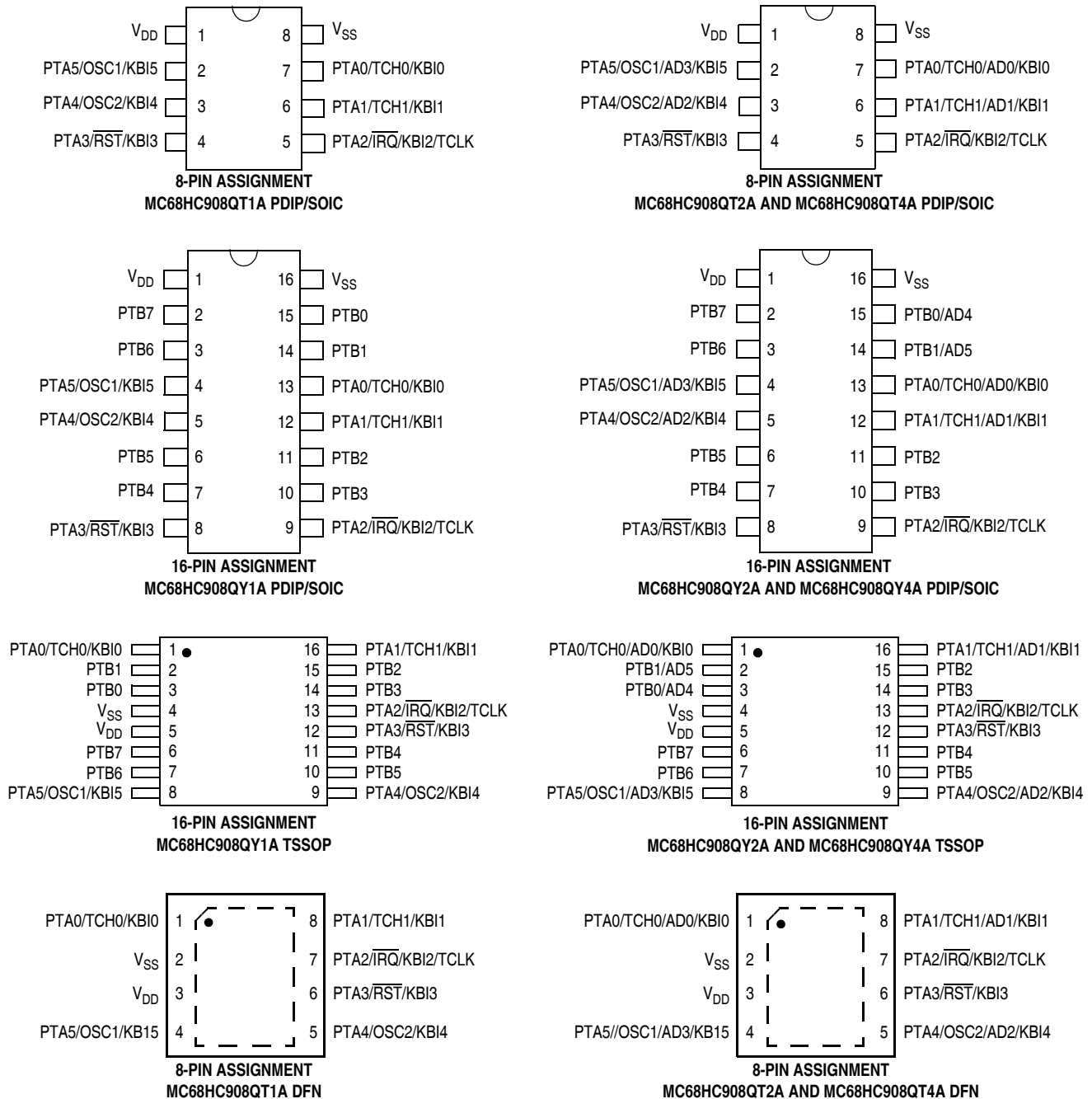


Figure 1-2. MCU Pin Assignments

1.5 Pin Functions

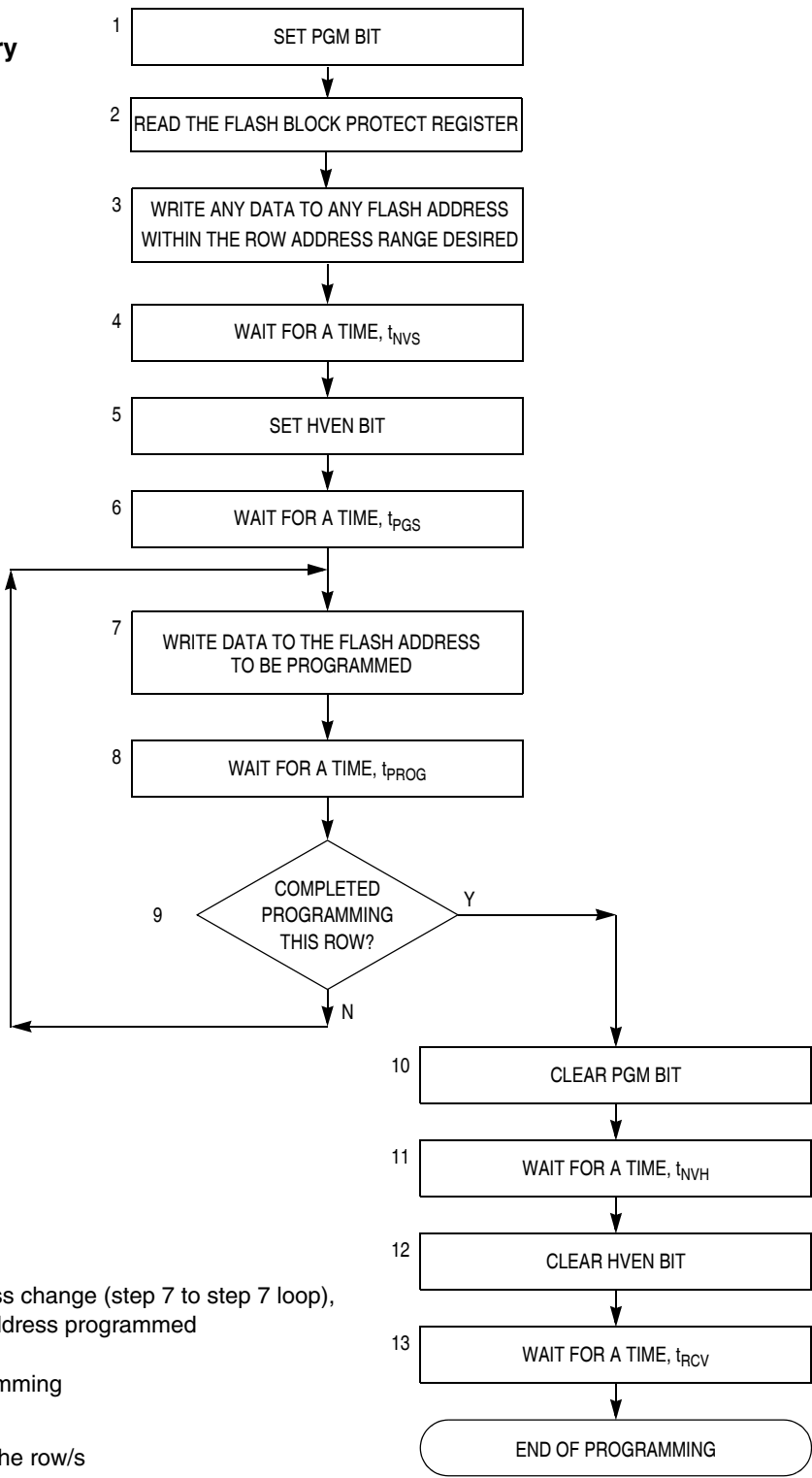
Table 1-2 provides a description of the pin functions.

Table 1-2. Pin Functions

Pin Name	Description	Input/Output
V _{DD}	Power supply	Power
V _{SS}	Power supply ground	Power
PTA0	PTA0 — General purpose I/O port	Input/Output
	TCH0 — Timer Channel 0 I/O	Input/Output
	AD0 — A/D channel 0 input	Input
	KBI0 — Keyboard interrupt input 0	Input
PTA1	PTA1 — General purpose I/O port	Input/Output
	TCH1 — Timer Channel 1 I/O	Input/Output
	AD1 — A/D channel 1 input	Input
	KBI1 — Keyboard interrupt input 1	Input
PTA2	PTA2 — General purpose input-only port	Input
	IRQ — External interrupt with programmable pullup and Schmitt trigger input	Input
	KBI2 — Keyboard interrupt input 2	Input
	TCLK — Timer clock input	Input
PTA3	PTA3 — General purpose I/O port	Input/Output
	RST — Reset input, active low with internal pullup and Schmitt trigger	Input
	KBI3 — Keyboard interrupt input 3	Input
PTA4	PTA4 — General purpose I/O port	Input/Output
	OSC2 — XTAL oscillator output (XTAL option only) RC or internal oscillator output (OSC2EN = 1 in PTAPUE register)	Output Output
	AD2 — A/D channel 2 input	Input
	KBI4 — Keyboard interrupt input 4	Input
PTA5	PTA5 — General purpose I/O port	Input/Output
	OSC1 — XTAL, RC, or external oscillator input	Input
	AD3 — A/D channel 3 input	Input
	KBI5 — Keyboard interrupt input 5	Input
PTB0 ⁽¹⁾	PTB0 — General-purpose I/O port	Input/Output
	AD4 — A/D channel 4 input	Input
PTB1 ⁽¹⁾	PTB1 — General-purpose I/O port	Input/Output
	AD5 — A/D channel 5 input	Input
PTB2-PTB7 ⁽¹⁾	6 General-purpose I/O port	Input/Output

1. The PTB pins are not available on the 8-pin packages.

**Algorithm for Programming
a Row (32 Bytes) of FLASH Memory**



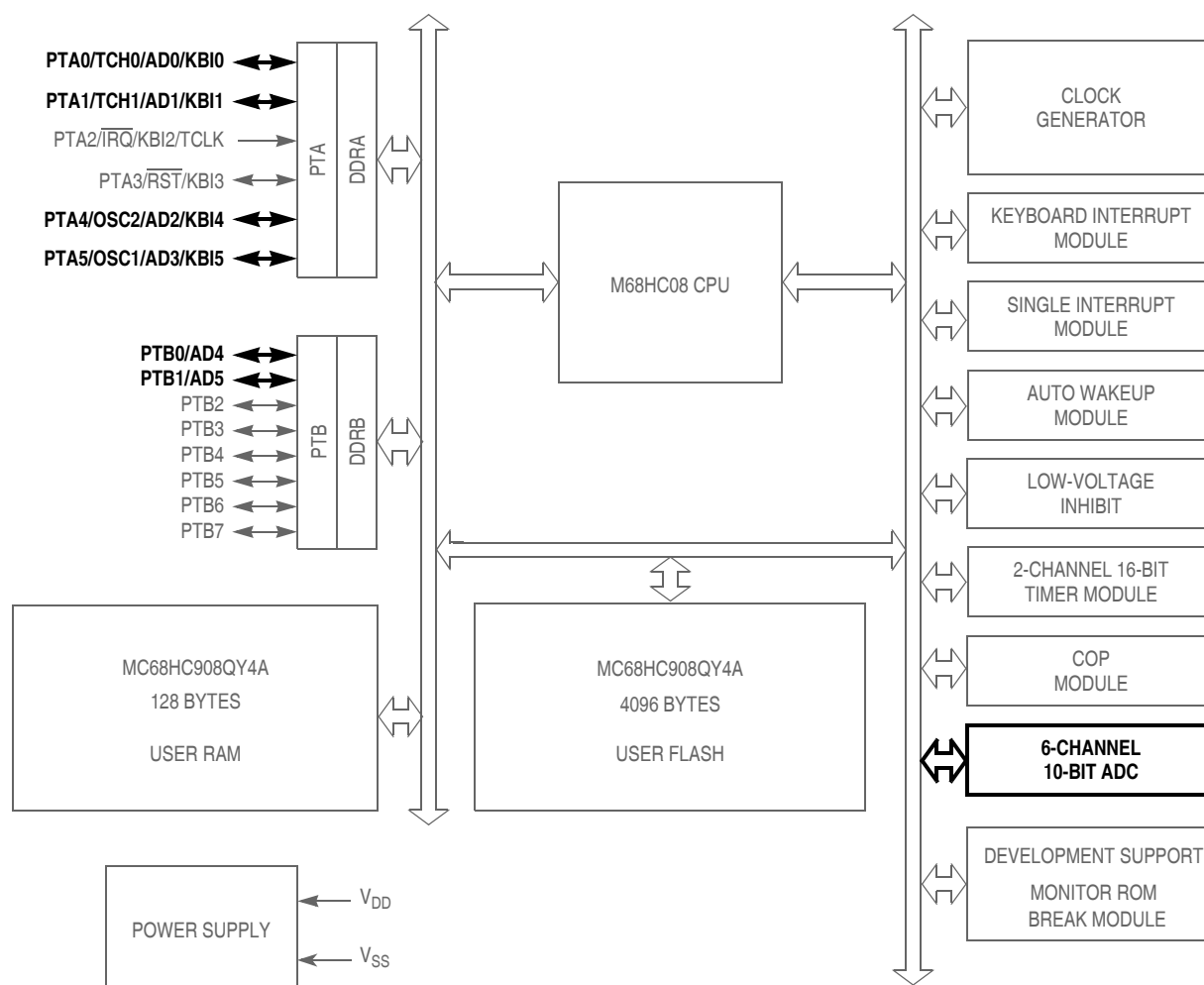
NOTES:

The time between each FLASH address change (step 7 to step 7 loop), or the time between the last FLASH address programmed to clearing PGM bit (step 7 to step 10) 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 2-4. FLASH Programming Flowchart

Analog-to-Digital Converter (ADC10) Module



\overline{RST} , \overline{IRQ} : Pins have internal pull up device
 All port pins have programmable pull up device
 PTA[0:5]: Higher current sink and source capability
 PTB[0:7]: Not available on 8-pin devices

Figure 3-1. Block Diagram Highlighting ADC10 Block and Pins

3.3.4.4 Code Width and Quantization Error

The ADC10 quantizes the ideal straight-line transfer function into 1024 steps (in 10-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points from one code to the next. The ideal code width for an N bit converter (in this case N can be 8 or 10), defined as 1LSB, is:

$$1\text{LSB} = (V_{\text{REFH}} - V_{\text{REFL}}) / 2^N$$

Because of this quantization, there is an inherent quantization error. Because the converter performs a conversion and then rounds to 8 or 10 bits, the code will transition when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be $\pm 1/2\text{LSB}$ in 8- or 10-bit mode. As a consequence, however, the code width of the first (\$000) conversion is only 1/2LSB and the code width of the last (\$FF or \$3FF) is 1.5LSB.

3.3.4.5 Linearity Errors

The ADC10 may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the user should be aware of them because they affect overall accuracy. These errors are:

- Zero-Scale Error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2LSB). Note, if the first conversion is \$001, then the difference between the actual \$001 code width and its ideal (1LSB) is used.
- Full-Scale Error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5LSB). Note, if the last conversion is \$3FE, then the difference between the actual \$3FE code width and its ideal (1LSB) is used.
- Differential Non-Linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral Non-Linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total Unadjusted Error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function, and therefore includes all forms of error.

3.3.4.6 Code Jitter, Non-Monotonicity and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

- Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around $\pm 1/2\text{LSB}$ but will increase with noise.
- Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Non-monotonicity is present if the apparent code jitter covers three codes (when the converter's output is indeterminate between three values for a given input voltage) or is greater than 1LSB.
- Missing codes are those which are never converted for any input value. In 8-bit or 10-bit mode, the ADC10 is guaranteed to be monotonic and to have no missing codes.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0


 = Unimplemented

Figure 3-4. ADC10 Data Register High (ADRH), 8-Bit Mode

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	AD9	AD8
Write:								
Reset:	0	0	0	0	0	0	0	0


 = Unimplemented

Figure 3-5. ADC10 Data Register High (ADRH), 10-Bit Mode

3.8.3 ADC10 Result Low Register (ADRL)

This register holds the LSBs of the result. This register is updated each time a conversion completes. Reading ADRH prevents the ADC10 from transferring subsequent conversion results into the result registers until ADRL is read. If ADRL is not read until the after next conversion is completed, then the intermediate conversion result will be lost. In 8-bit mode, there is no interlocking with ADRH.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
Write:								
Reset:	0	0	0	0	0	0	0	0


 = Unimplemented

Figure 3-6. ADC10 Data Register Low (ADRL)

3.8.4 ADC10 Clock Register (ADCLK)

This register selects the clock frequency for the ADC10 and the modes of operation.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ACLKEN
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 3-7. ADC10 Clock Register (ADCLK)

ADLPC — ADC10 Low-Power Configuration Bit

ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required.

- 1 = Low-power configuration: The power is reduced at the expense of maximum clock speed.
- 0 = High-speed configuration

9.7 I/O Signals

The KBI module can share its pins with the general-purpose I/O pins. See [Figure 9-1](#) for the port pins that are shared.

9.7.1 KBI Input Pins (KBIx:KBI0)

Each KBI pin is independently programmable as an external interrupt source. KBI pin polarity can be controlled independently. Each KBI pin when enabled will automatically configure the appropriate pullup/pulldown device based on polarity.

9.8 Registers

The following registers control and monitor operation of the KBI module:

- KBSCR (keyboard interrupt status and control register)
- KBIER (keyboard interrupt enable register)
- KBIPR (keyboard interrupt polarity register)

9.8.1 Keyboard Status and Control Register (KBSCR)

Features of the KBSCR:

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

	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

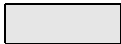
 = Unimplemented

Figure 9-3. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending.

- 1 = Keyboard interrupt pending
- 0 = No keyboard interrupt pending

ACKK — Keyboard Acknowledge Bit

Writing a 1 to this write-only bit clears the KBI request. ACKK always reads 0.

IMASKK — Keyboard Interrupt Mask Bit

Writing a 1 to this read/write bit prevents the output of the KBI latch from generating interrupt requests.

- 1 = Keyboard interrupt requests disabled
- 0 = Keyboard interrupt requests enabled

MODEK — Keyboard Triggering Sensitivity Bit

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

- 1 = Keyboard interrupt requests on edge and level
- 0 = Keyboard interrupt requests on edge only

The value in the TIM channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing \$0080 (128) to the TIM channel registers produces a duty cycle of 128/256 or 50%.

14.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in [14.3.4 Pulse Width Modulation \(PWM\)](#). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIM overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM may pass the new value before it is written to the timer channel (TCHxH:TCHxL).

Use the following methods to synchronize unbuffered changes in the PWM pulse width on channel x:

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

NOTE

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

14.3.4.2 Buffered PWM Signal Generation

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

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

NOTE

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

14.4 Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the counter reaches the modulo value programmed in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow interrupt requests. TOF and TOIE are in the TSC register.
- TIM channel flags (CH1F:CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM interrupt requests are controlled by the channel x interrupt enable bit, CHxIE. Channel x TIM interrupt requests are enabled when CHxIE = 1. CHxF and CHxIE are in the TSCx register.

14.5 Low-Power Modes

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

14.5.1 Wait Mode

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

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

14.5.2 Stop Mode

The TIM module is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions. TIM operation resumes after an external interrupt. If stop mode is exited by reset, the TIM is reset.

14.6 TIM During Break Interrupts

A break interrupt stops the counter.

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. See BFCR in the SIM section of this data sheet.

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

Chapter 15

Development Support

15.1 Introduction

This section describes the break module, the monitor module (MON), and the monitor mode entry methods.

15.2 Break Module (BRK)

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

Features include:

- Accessible input/output (I/O) registers during the break Interrupt
- Central processor unit (CPU) generated break interrupts
- Software-generated break interrupts
- Computer operating properly (COP) disabling during break interrupts

15.2.1 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 system integration module (SIM). The SIM then causes the CPU to load the instruction register with a software interrupt instruction (SWI). 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 1 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 is generated. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the microcontroller unit (MCU) to normal operation.

Figure 15-2 shows the structure of the break module.

When the internal address bus matches the value written in the break address registers or when software writes a 1 to the BRKA bit in the break status and control register, the CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with \$FFFC and \$FFFD (\$FEFC and \$FEFD in monitor mode)

16.11 Oscillator Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Internal oscillator frequency ⁽¹⁾ ICFS1:ICFS0 = 00 ICFS1:ICFS0 = 01 ICFS1:ICFS0 = 10 (not allowed if $V_{DD} < 2.7V$)	f_{INTCLK}	— — —	4 8 12.8	— — —	MHz
Trim accuracy ⁽²⁾⁽³⁾	Δ_{TRIM_ACC}	—	± 0.4	—	%
Deviation from trimmed Internal oscillator ⁽³⁾⁽⁴⁾ 4, 8, 12.8MHz, $V_{DD} \pm 10\%$, 0 to 70°C 4, 8, 12.8MHz, $V_{DD} \pm 10\%$, -40 to 125°C	Δ_{INT_TRIM}	— —	± 2 —	— ± 5	%
External RC oscillator frequency, RCCLK ⁽¹⁾⁽³⁾	f_{RCCLK}	2	—	10	MHz
External clock reference frequency ⁽¹⁾⁽⁵⁾⁽⁶⁾ $V_{DD} \geq 4.5V$ $V_{DD} < 4.5V$	f_{OSCCLK}	dc dc	—	32 16	MHz
RC oscillator external resistor ⁽³⁾ $V_{DD} = 5V$ $V_{DD} = 3V$	R_{EXT}	See Figure 16-7 See Figure 16-8			—
Crystal frequency, XTALCLK ⁽¹⁾⁽⁷⁾⁽⁸⁾ ECFS1:ECFS0 = 00 ($V_{DD} \geq 4.5V$) ECFS1:ECFS0 = 00 ECFS1:ECFS0 = 01 ECFS1:ECFS0 = 10	f_{OSCCLK}	8 8 1 30	—	32 16 8 100	MHz MHz MHz kHz
ECFS1:ECFS0 = 00 ⁽⁹⁾ Feedback bias resistor Crystal load capacitance ⁽¹⁰⁾ Crystal capacitors ⁽¹⁰⁾	R_B C_L C_1, C_2	— — —	1 20 (2 x C_L) – 5pF	— — —	MΩ pF pF
ECFS1:ECFS0 = 01 ⁽⁹⁾ Crystal series damping resistor $f_{OSCCLK} = 1MHz$ $f_{OSCCLK} = 4MHz$ $f_{OSCCLK} = 8MHz$ Feedback bias resistor Crystal load capacitance ⁽¹⁰⁾ Crystal capacitors ⁽¹⁰⁾	R_S R_B C_L C_1, C_2	— — — — — — —	20 10 0 5 18 (2 x C_L) – 10 pF	— — — — — —	kΩ kΩ kΩ MΩ pF pF
AWU module internal RC oscillator frequency	f_{INTRC}	—	32	—	kHz

1. Bus frequency, f_{OP} , is oscillator frequency divided by 4.
2. Factory trimmed to provided 12.8MHz accuracy requirement ($\pm 5\%$, @25°C) for forced monitor mode communication. User should trim in-circuit to obtain the most accurate internal oscillator frequency for the application.
3. Values are based on characterization results, not tested in production.
4. Deviation values assumes trimming in target application @25°C and midpoint of voltage range, for example 5.0 V for 5 V $\pm 10\%$ operation.
5. No more than 10% duty cycle deviation from 50%.
6. When external oscillator clock is greater than 1MHz, ECFS1:ECFS0 must be 00 or 01
7. Use fundamental mode only, do **not** use overtone crystals or overtone ceramic resonators
8. Due to variations in electrical properties of external components such as, ESR and Load Capacitance, operation above 16 MHz is not guaranteed for all crystals or ceramic resonators. Operation above 16 MHz requires that a Negative Resistance Margin (NRM) characterization and component optimization be performed by the crystal or ceramic resonator vendor for every different type of crystal or ceramic resonator which will be used. This characterization and optimization must be performed at the extremes of voltage and temperature which will be applied to the microcontroller in the application. The NRM must meet or exceed 10x the maximum ESR of the crystal or ceramic resonator for acceptable performance.
9. Do not use damping resistor when ECFS1:ECFS0 = 00 or 10
10. Consult crystal vendor data sheet.











