

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	SCI, SPI
Peripherals	LVD, POR, PWM
Number of I/O	16
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc908qc16vdse

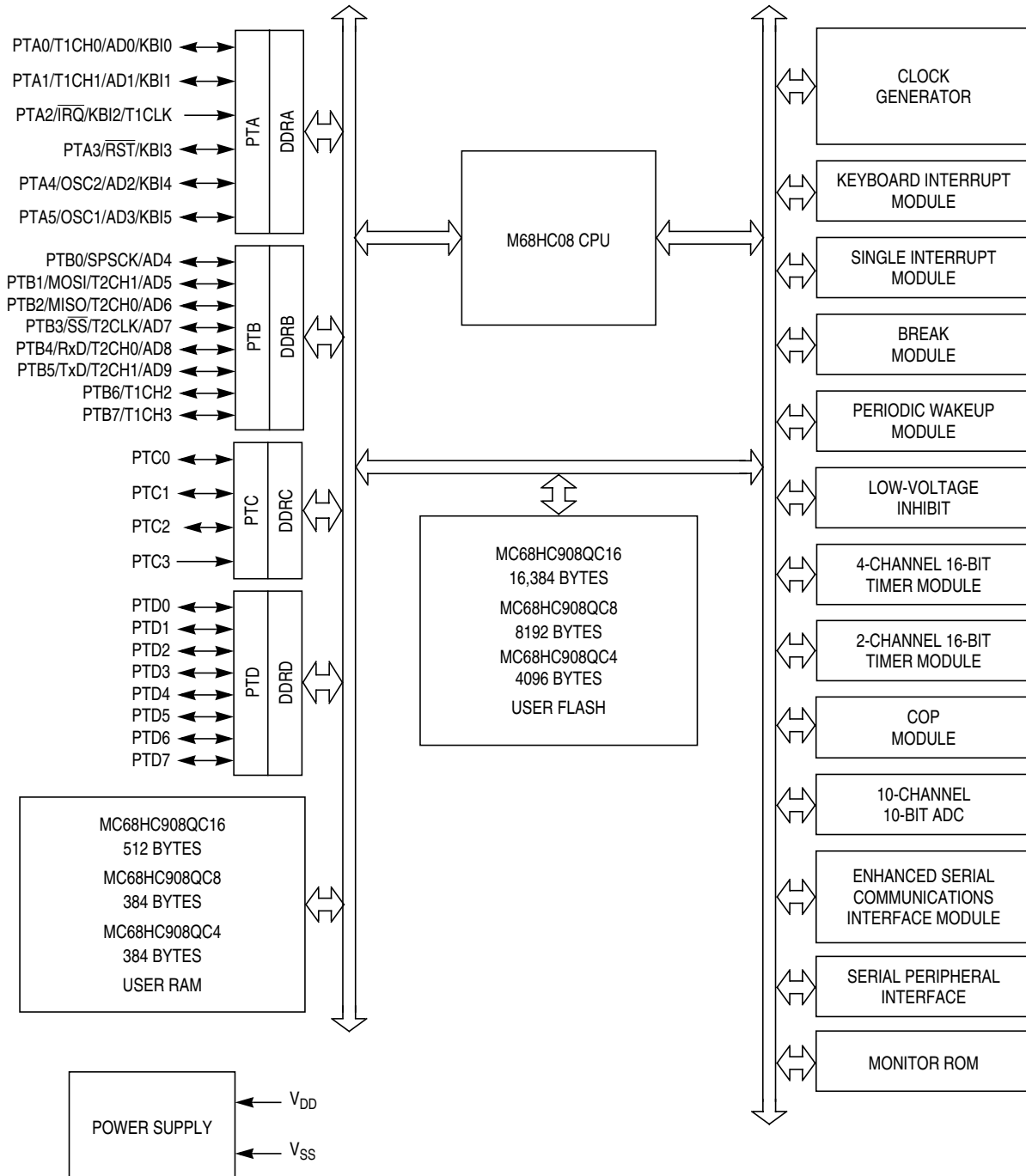
Table of Contents

13.3.2.2	Character Transmission	126
13.3.2.3	Break Characters	127
13.3.2.4	Idle Characters	128
13.3.2.5	Inversion of Transmitted Output	128
13.3.3	Receiver	128
13.3.3.1	Character Length	128
13.3.3.2	Character Reception	128
13.3.3.3	Data Sampling	130
13.3.3.4	Framing Errors	131
13.3.3.5	Baud Rate Tolerance	131
13.3.3.6	Receiver Wakeup	133
13.4	Interrupts	134
13.4.1	Transmitter Interrupts	134
13.4.2	Receiver Interrupts	134
13.4.3	Error Interrupts	134
13.5	Low-Power Modes	135
13.5.1	Wait Mode	135
13.5.2	Stop Mode	135
13.6	ESCI During Break Interrupts	135
13.7	I/O Signals	135
13.7.1	ESCI Transmit Data (TxD)	135
13.7.2	ESCI Receive Data (RxD)	136
13.8	Registers	136
13.8.1	ESCI Control Register 1	136
13.8.2	ESCI Control Register 2	138
13.8.3	ESCI Control Register 3	140
13.8.4	ESCI Status Register 1	141
13.8.5	ESCI Status Register 2	143
13.8.6	ESCI Data Register	144
13.8.7	ESCI Baud Rate Register	144
13.8.8	ESCI Prescaler Register	145
13.9	ESCI Arbiter	149
13.9.1	ESCI Arbiter Control Register	149
13.9.2	ESCI Arbiter Data Register	150
13.9.3	Bit Time Measurement	150
13.9.4	Arbitration Mode	151

Chapter 14 System Integration Module (SIM)

14.1	Introduction	153
14.2	\overline{RST} and \overline{IRQ} Pins Initialization	153
14.3	SIM Bus Clock Control and Generation	153
14.3.1	Bus Timing	155
14.3.2	Clock Start-Up from POR	155
14.3.3	Clocks in Stop Mode and Wait Mode	155
14.4	Reset and System Initialization	155
14.4.1	External Pin Reset	155

General Description



All port pins can be configured with internal pullup
 PTC not available on 16-pin devices (see note in 11.1 Introduction)
 PTD not available on 16-pin or 20-pin devices (see note in 11.1 Introduction)

Figure 1-1. Block Diagram

1.6 Pin Function Priority

Table 1-3 is meant to resolve the priority if multiple functions are enabled on a single pin.

NOTE

Upon reset all pins come up as input ports regardless of the priority table.

Table 1-3. Function Priority in Shared Pins

Pin Name	Highest-to-Lowest Priority Sequence
PTA0 ⁽¹⁾	AD0 → T1CH0 → KBI0 → PTA0
PTA1 ⁽¹⁾	AD1 → T1CH1 → KBI1 → PTA1
PTA2	$\overline{\text{IRQ}}$ → T1CLK → KBI2 → PTA2
PTA3	$\overline{\text{RST}}$ → KBI3 → PTA3
PTA4 ⁽¹⁾	OSC2 → AD2 → KBI4 → PTA4
PTA5 ⁽¹⁾	OSC1 → AD3 → KBI5 → PTA5
PTB0 ⁽¹⁾	AD4 → SPCK → PTB0
PTB1 ⁽¹⁾	AD5 → MOSI → T2CH1 ⁽²⁾ → PTB1
PTB2 ⁽¹⁾	AD6 → MISO → T2CH0 ⁽²⁾ → PTB2
PTB3 ⁽¹⁾	AD7 → $\overline{\text{SS}}$ → T2CLK → PTB3
PTB4 ⁽¹⁾	AD8 → RxD → T2CH0 ⁽²⁾ → PTB4
PTB5 ⁽¹⁾	AD9 → TxD → T2CH1 ⁽²⁾ → PTB5
PTB6	T1CH2 → PTB6
PTB7	T1CH3 → PTB7
PTCx	PTCx
PTDx	PTDx

1. When a pin is to be used as an ADC pin, the I/O port function should be left as an input and all other shared modules should be disabled. The ADC does not override additional modules using the pin.
2. T2CH0 and T2CH1 can be repositioned using TIM2POS in CONFIG2 (see Figure 2-2. Control, Status, and Data Registers).

1.7 Unused Pin Termination

Input pins and I/O port pins that are not used in the application must be terminated. This prevents excess current caused by floating inputs, and enhances immunity during noise or transient events. Termination methods include:

1. Configuring unused pins as outputs and driving high or low;
2. Configuring unused pins as inputs and enabling internal pull-ups;
3. Configuring unused pins as inputs and using external pull-up or pull-down resistors.

Never connect unused pins directly to V_{DD} or V_{SS} .

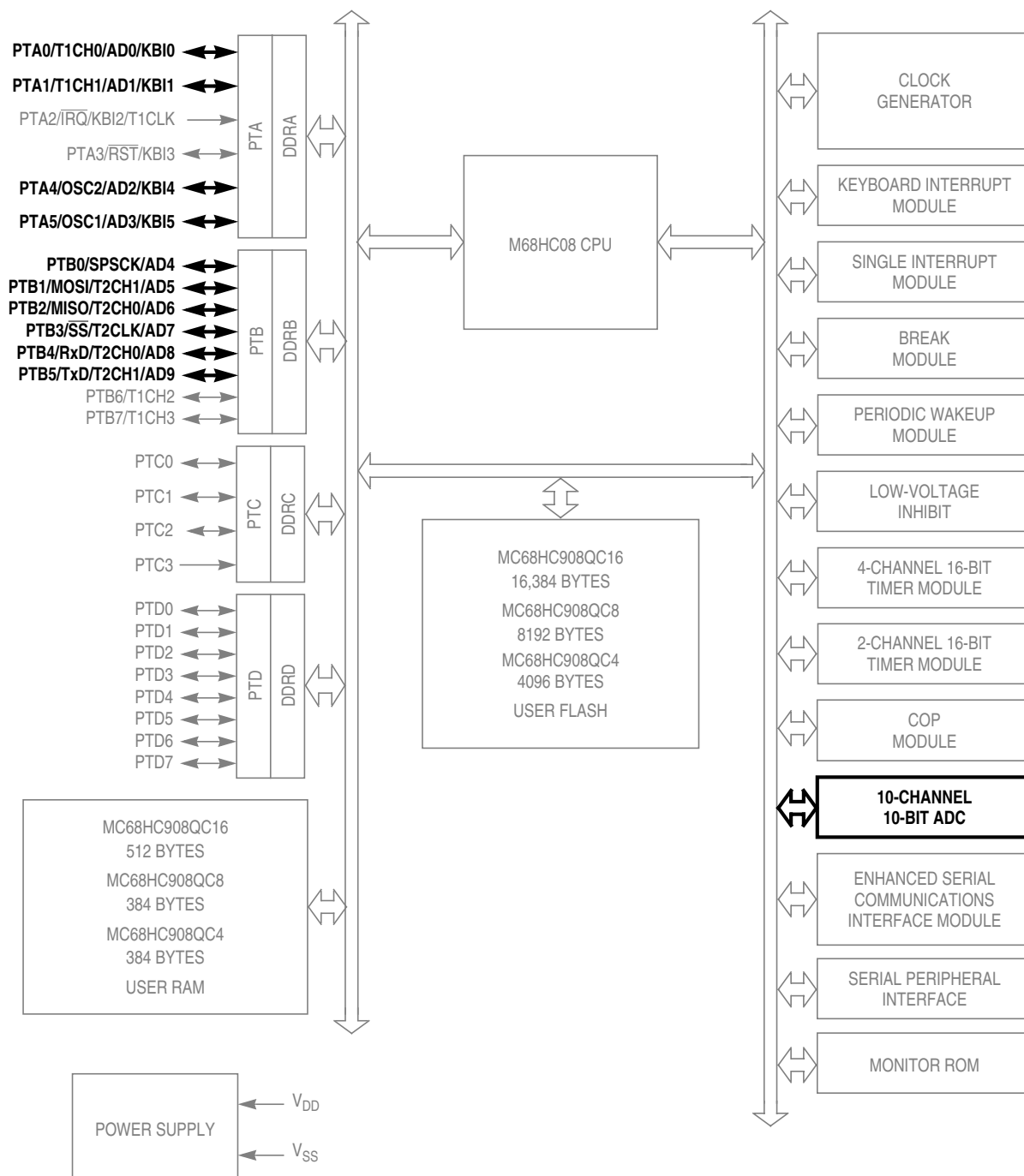
Since some general-purpose I/O pins are not available on all packages, these pins must be terminated as well. Either method 1 or 2 above are appropriate.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0034	TIM1 Channel 3 Register High (T1CH3H) See page 204.	Read:								
		Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	Indeterminate after reset							
\$0035	TIM1 Channel 3 Register Low (T1CH3L) See page 204.	Read:								
		Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	Indeterminate after reset							
\$0036	Oscillator Status and Control Register (OSCS) See page 104.	Read:								
		Write:	OSCOPT1	OSCOPT0	ICFS1	ICFS0	ECFS1	ECFS0	ECGON	ECGST
		Reset:	0	0	0	0	0	0	0	0
\$0037	Reserved									
\$0038	Oscillator Trim Register (OSCTRIM) See page 105.	Read:								
		Write:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Reset:	1	0	0	0	0	0	0	0
\$0039 ↓ \$003B	Reserved									
\$003C	ADC10 Status and Control Register (ADSCR) See page 54.	Read:	COCO							
		Write:		AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Reset:	0	0	0	1	1	1	1	1
\$003D	ADC10 Data Register High (ADRH) See page 56.	Read:	0	0	0	0	0	0	AD9	AD8
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
\$003E	ADC10 Data Register Low (ADRL) See page 56.	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
\$003F	ADC10 Clock Register (ADCLK) See page 56.	Read:								
		Write:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ADACKEN
		Reset:	0	0	0	0	0	0	0	0
\$0240	TIM2 Status and Control Register (T2SC) See page 213.	Read:	TOF			0	0			
		Write:	0	TOIE	TSTOP	TRST		PS2	PS1	PS0
		Reset:	0	0	1	0	0	0	0	0
\$0241	TIM2 Counter Register High (T2CNTH) See page 214.	Read:								
		Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented
R = Reserved
U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 8)

Analog-to-Digital Converter (ADC10) Module



All port pins can be configured with internal pullup
 PTC not available on 16-pin devices (see note in 11.1 Introduction)
 PTD not available on 16-pin or 20-pin devices (see note in 11.1 Introduction)

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

clocks are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by ADIV[1:0] and can be divide-by 1, 2, 4, or 8.

3.3.2 Input Select and Pin Control

Only one analog input may be used for conversion at any given time. The channel select bits in ADSCR are used to select the input signal for conversion.

3.3.3 Conversion Control

Conversions can be performed in either 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC10 module can be configured for low power operation, long sample time, and continuous conversion.

3.3.3.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADSCR (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADSCR is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

3.3.3.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADRH and ADRL. This is indicated by the setting of COCO. An interrupt request is generated if AIEN is set at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADRH and ADRL if the previous data is in the process of being read while in 10-bit mode (ADRH has been read but ADRL has not). In this case the data transfer is blocked, COCO is not set, and the new result is lost. When a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled). If single conversions are enabled, this could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

3.3.3.3 Aborting Conversions

Any conversion in progress will be aborted when:

- A write to ADSCR occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCLK occurs.
- The MCU is reset.
- The MCU enters stop mode with ACLK not enabled.

3.7 I/O Signals

The ADC10 module shares its pins with general-purpose input/output (I/O) port pins. See Figure 3-1 for port location of these shared pins. The ADC10 on this MCU uses V_{DD} and V_{SS} as its supply and reference pins. This MCU does not have an external trigger source.

3.7.1 ADC10 Analog Power Pin (V_{DDA})

The ADC10 analog portion uses V_{DDA} as its power pin. In some packages, V_{DDA} is connected internally to V_{DD} . If externally available, connect the V_{DDA} pin to the same voltage potential as V_{DD} . External filtering may be necessary to ensure clean V_{DDA} for good results.

NOTE

If externally available, route V_{DDA} carefully for maximum noise immunity and place bypass capacitors as near as possible to the package.

3.7.2 ADC10 Analog Ground Pin (V_{SSA})

The ADC10 analog portion uses V_{SSA} as its ground pin. In some packages, V_{SSA} is connected internally to V_{SS} . If externally available, connect the V_{SSA} pin to the same voltage potential as V_{SS} .

In cases where separate power supplies are used for analog and digital power, the ground connection between these supplies should be at the V_{SSA} pin. This should be the only ground connection between these supplies if possible. The V_{SSA} pin makes a good single point ground location.

3.7.3 ADC10 Voltage Reference High Pin (V_{REFH})

V_{REFH} is the power supply for setting the high-reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDA} . If externally available, V_{REFH} may be connected to the same potential as V_{DDA} , or may be driven by an external source that is between the minimum V_{DDA} spec and the V_{DDA} potential (V_{REFH} must never exceed V_{DDA}).

NOTE

Route V_{REFH} carefully for maximum noise immunity and place bypass capacitors as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as close as possible to the package pins. Resistance in the path is not recommended because the current will cause a voltage drop which could result in conversion errors. Inductance in this path must be minimum (parasitic only).

3.7.4 ADC10 Voltage Reference Low Pin (V_{REFL})

V_{REFL} is the power supply for setting the low-reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSA} . If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSA} . There will be a brief current associated with V_{REFL} when the sampling capacitor is charging. If externally available, connect the V_{REFL} pin to the same potential as V_{SSA} at the single point ground location.

Z — Zero Flag

The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of \$00.

1 = Zero result

0 = Non-zero result

C — Carry/Borrow Flag

The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag.

1 = Carry out of bit 7

0 = No carry out of bit 7

6.4 Arithmetic/Logic Unit (ALU)

The ALU performs the arithmetic and logic operations defined by the instruction set.

Refer to the *CPU08 Reference Manual* (document order number CPU08RM/AD) for a description of the instructions and addressing modes and more detail about the architecture of the CPU.

6.5 Low-Power Modes

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

6.5.1 Wait Mode

The WAIT instruction:

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

6.5.2 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.

6.6 CPU During Break Interrupts

If a break module is present on the MCU, the CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with \$FFFC:\$FFFD or with \$FEFC:\$FEFD in monitor mode

The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

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.

8.3.1 Keyboard Operation

Writing to the KBIE_x bits in the keyboard interrupt enable register (KBIER) independently enables or disables each KBI pin. The polarity of the keyboard interrupt is controlled using the KBIP_x bits in the keyboard interrupt polarity register (KBIPR). Edge-only or edge and level sensitivity is controlled using the MODEK bit in the keyboard status and control register (KBISCR).

Enabling a keyboard interrupt pin also enables its internal pullup or pulldown device based on the polarity enabled. On falling edge or low level detection, a pullup device is configured. On rising edge or high level detection, a pulldown device is configured.

The keyboard interrupt latch is set when one or more enabled keyboard interrupt inputs are asserted.

- If the keyboard interrupt sensitivity is edge-only, for KBIP_x = 0, a falling (for KBIP_x = 1, a rising) edge on a keyboard interrupt input does not latch an interrupt request if another enabled keyboard pin is already asserted. To prevent losing an interrupt request on one input because another input remains asserted, software can disable the latter input while it is asserted.
- If the keyboard interrupt is edge and level sensitive, an interrupt request is present as long as any enabled keyboard interrupt input is asserted.

8.3.1.1 MODEK = 1

If the MODEK bit is set, the keyboard interrupt inputs are both edge and level sensitive. The KBIP_x bit will determine whether a edge sensitive pin detects rising or falling edges and on level sensitive pins whether the pin detects low or high levels. With MODEK set, both of the following actions must occur to clear a keyboard interrupt request:

- Return of all enabled keyboard interrupt inputs to a deasserted level. As long as any enabled keyboard interrupt pin is asserted, the keyboard interrupt remains active.
- Vector fetch or software clear. A KBI vector fetch generates an interrupt acknowledge signal to clear the KBI latch. Software generates the interrupt acknowledge signal by writing a 1 to ACKK in KBSCR. The ACKK bit is useful in applications that poll the keyboard interrupt inputs and require software to clear the KBI latch. Writing to ACKK 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 inputs. An edge detect that occurs after writing to ACKK latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the KBI vector address.

The KBI vector fetch or software clear and the return of all enabled keyboard interrupt pins to a deasserted level may occur in any order.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt input stays asserted.

8.3.1.2 MODEK = 0

If the MODEK bit is clear, the keyboard interrupt inputs are edge sensitive. The KBIP_x bit will determine whether an edge sensitive pin detects rising or falling edges. A KBI vector fetch or software clear immediately clears the KBI latch.

Keyboard Interrupt Module (KBI)

	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 8-3. Keyboard Status and Control Register (KBSCR)

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

8.8.2 Keyboard Interrupt Enable Register (KBIER)

KBIER enables or disables each keyboard interrupt pin.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	R	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Write:								
Reset:	0	0	0	0	0	0	0	0

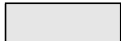
 = Unimplemented

Figure 8-4. Keyboard Interrupt Enable Register (KBIER)

KBIE5–KBIE0 — Keyboard Interrupt Enable Bits

Each of these read/write bits enables the corresponding keyboard interrupt pin to latch KBI interrupt requests.

- 1 = KBIx pin enabled as keyboard interrupt pin
- 0 = KBIx pin not enabled as keyboard interrupt pin

R — Reserved bit

This reserved bit should always be written to a 0 and will read 0.

Chapter 9

Low-Voltage Inhibit (LVI)

9.1 Introduction

The low-voltage inhibit (LVI) module is provided as a system protection mechanism to prevent the MCU from operating below a certain operating supply voltage level. The module has several configuration options to allow functionality to be tailored to different system level demands.

The configuration registers (see Chapter 4 Configuration Registers (CONFIG1 and CONFIG2)) contain control bits for this module.

9.2 Features

Features of the LVI module include:

- Programmable LVI reset
- Selectable LVI trip voltage
- Programmable stop mode operation

9.3 Functional Description

Figure 9-1 shows the structure of the LVI module. LVISTOP, LVIPWRD, LVITRIP, and LVIRSTD are user selectable options found in the configuration register.

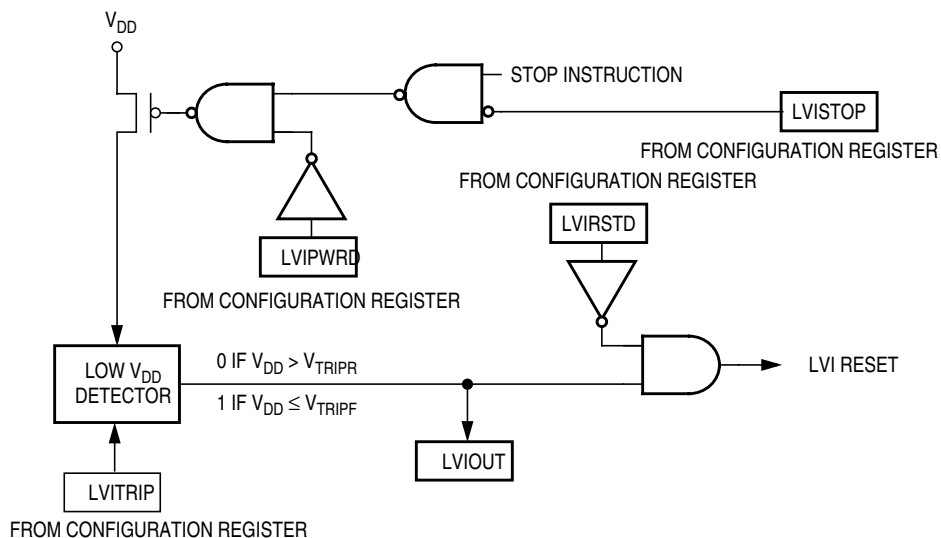


Figure 9-1. LVI Module Block Diagram

Chapter 12

Periodic Wakeup Module (PWU)

12.1 Introduction

This section describes the periodic wakeup (PWU) module. The PWU is available in all modes of operation (run, wait, and stop) and performs two main functions:

- Generate periodic wakeup requests to bring the microcontroller unit (MCU) out of stop mode.
- Generate periodic interrupt requests during run and wait modes.

12.2 Features

Features of the periodic wakeup module include:

- Interrupt with separate interrupt enable bit, interrupt vector and interrupt mask bit
- Exit from low-power stop mode without external signals
- Programmable clock input
- Selectable timeout periods (40 μ s to 3 minutes with an adjustment resolution of better than 1% for periods over 4 ms)
- Dedicated low-power 32 kHz internal oscillator separate from the main system clock sources
- Option to allow bus clock source to run the PWU
- Accessible in all modes of operation (run, wait, and stop)

12.3 Functional Description

Figure 12-1 is a block diagram of the PWU.

The PWU module consists of a PWU counter whose count is reset once it equals the value stored in the PWU modulo register (PWUMOD). The PWU counter clock, PWUCLOCK, frequency is selectable using the PWU prescaler register (PWUP). The prescaler clock source can be selected using the PWUCLKSEL bit in the PWU status and control register (PWUSC) and can either be the internal RC oscillator or the BUSCLKX4 clock. The PWUCLKSEL bit, PWUMOD and PWUP registers can only be written to when the PWUON bit is clear.

The PWUON bit in PWUSC is used to enable the PWU module. The SMODE bit in PWUSC is used to allow an enabled PWU module to continue running in stop mode. BUSCLKX4 must be enabled to run in stop mode if used as the clock source for the PWU when SMODE is set. See Chapter 4 Configuration Registers (CONFIG1 and CONFIG2) on enabling BUSCLKX4 to run in stop mode.

The PWU counter when enabled will count until it reaches the value stored in the PWU modulo register (PWUMOD), when they are equal the read-only PWU flag (PWUF) in PWUSC is latched. The PWU interrupt enable bit, PWUIE, in PWUSC enables a PWU interrupt request. The PWUF can be cleared by writing to the PWU acknowledge bit, PWUACK in PWUSC or by a PWU interrupt vector fetch.

error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming character, it resynchronizes the RT clock on any valid falling edge within the character. Resynchronization within characters corrects misalignments between transmitter bit times and receiver bit times.

Slow Data Tolerance

Figure 13-7 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

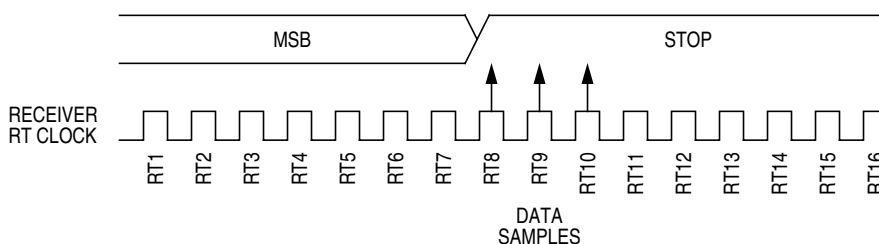


Figure 13-7. Slow Data

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

With the misaligned character shown in Figure 13-7, the receiver counts 154 RT cycles at the point when the count of the transmitting device is
 $9 \text{ bit times} \times 16 \text{ RT cycles} + 3 \text{ RT cycles} = 147 \text{ RT cycles}$.

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

$$\left| \frac{154 - 147}{154} \right| \times 100 = 4.54\%$$

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

With the misaligned character shown in Figure 13-7, the receiver counts 170 RT cycles at the point when the count of the transmitting device is
 $10 \text{ bit times} \times 16 \text{ RT cycles} + 3 \text{ RT cycles} = 163 \text{ RT cycles}$.

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

$$\left| \frac{170 - 163}{170} \right| \times 100 = 4.12\%$$

Chapter 14

System Integration Module (SIM)

14.1 Introduction

This section describes the system integration module (SIM), which supports up to 24 external and/or internal interrupts. Together with the central processor unit (CPU), the SIM controls all microcontroller unit (MCU) activities. A block diagram of the SIM is shown in Figure 14-1. The SIM is a system state controller that coordinates CPU and exception timing.

The SIM is responsible for:

- Bus clock generation and control for CPU and peripherals
 - Stop/wait/reset/break entry and recovery
 - Internal clock control
- Master reset control, including power-on reset (POR) and computer operating properly (COP) timeout
- Interrupt control:
 - Acknowledge timing
 - Arbitration control timing
 - Vector address generation
- CPU enable/disable timing

Table 14-1. Signal Name Conventions

Signal Name	Description
BUSCLKX4	Buffered clock from the internal, RC or XTAL oscillator circuit.
BUSCLKX2	The BUSCLKX4 frequency divided by two. This signal is again divided by two in the SIM to generate the internal bus clocks (bus clock = BUSCLKX4 / 4).
Address bus	Internal address bus
Data bus	Internal data bus
PORRST	Signal from the power-on reset module to the SIM
IRST	Internal reset signal
R/W	Read/write signal

14.2 \overline{RST} and \overline{IRQ} Pins Initialization

\overline{RST} and \overline{IRQ} pins come out of reset as PTA3 and PTA2 respectively. \overline{RST} and \overline{IRQ} functions can be activated by programming CONFIG2 accordingly. Refer to Chapter 4 Configuration Registers (CONFIG1 and CONFIG2).

14.3 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, BUSCLKX2, as shown in Figure 14-2.

System Integration Module (SIM)

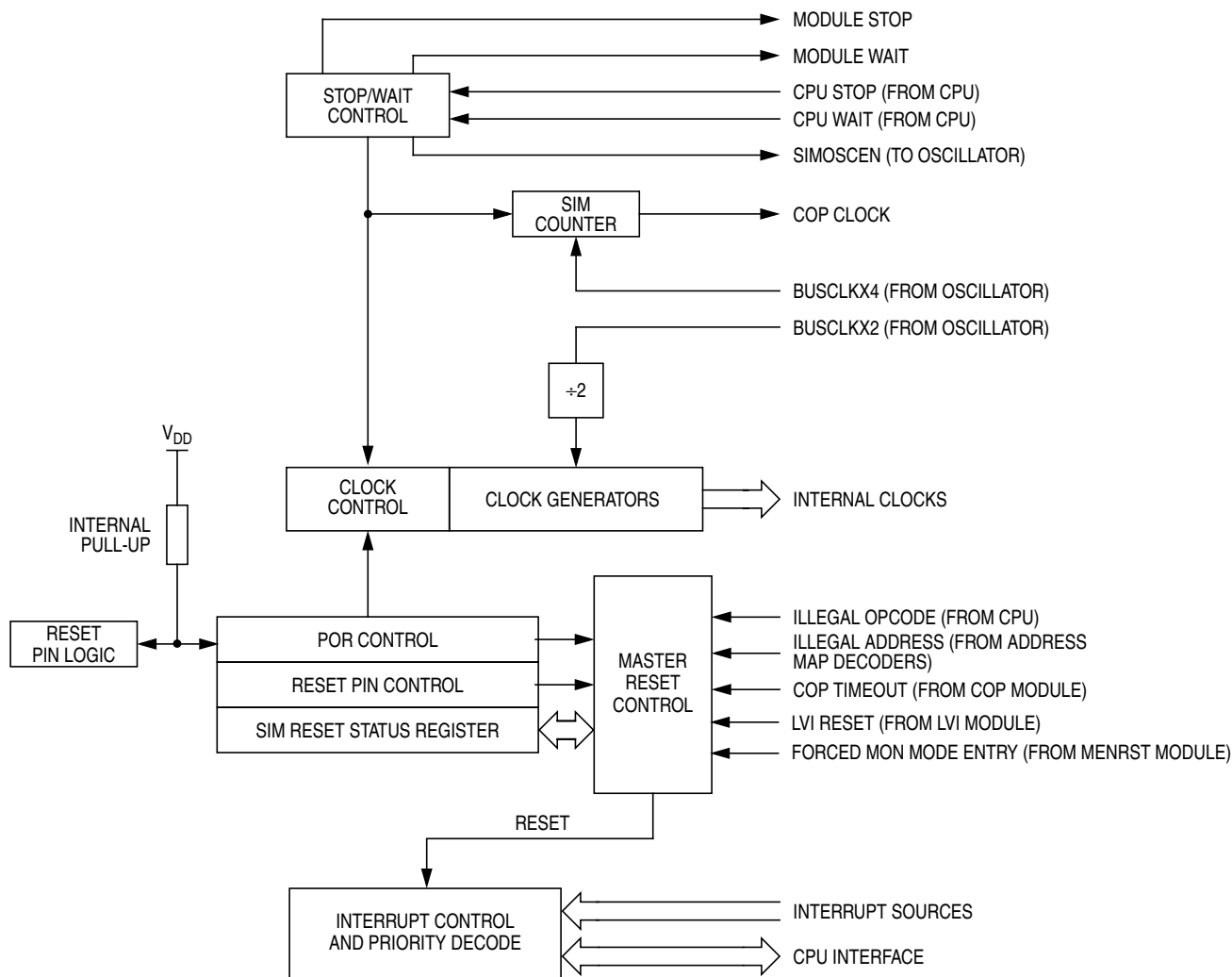


Figure 14-1. SIM Block Diagram

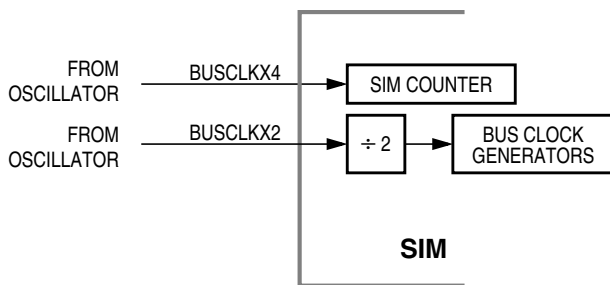


Figure 14-2. SIM Clock Signals

Chapter 15

Serial Peripheral Interface (SPI) Module

15.1 Introduction

This section describes the serial peripheral interface (SPI) module, which allows full-duplex, synchronous, serial communications with peripheral devices.

The SPI shares its pins with general-purpose input/output (I/O) port pins. See Figure 15-1 for port location of these shared pins.

15.2 Features

Features of the SPI module include:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four master mode frequencies (maximum = bus frequency \div 2)
- Maximum slave mode frequency = bus frequency
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts:
 - SPRF (SPI receiver full)
 - SPTE (SPI transmitter empty)
- Mode fault error flag with interrupt capability
- Overflow error flag with interrupt capability
- Programmable wired-OR mode

15.3 Functional Description

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs. Software can poll the SPI status flags or SPI operation can be interrupt driven.

The following paragraphs describe the operation of the SPI module.

Serial Peripheral Interface (SPI) Module

input (\overline{SS}) is low, so that only the selected slave drives to the master. The \overline{SS} pin of the master is not shown but is assumed to be inactive. The \overline{SS} pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See 15.3.6.2 Mode Fault Error.) When $CPHA = 0$, the first SPSCCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCCK edge, and a falling edge on the \overline{SS} pin is used to start the slave data transmission. The slave's \overline{SS} pin must be toggled back to high and then low again between each byte transmitted as shown in Figure 15-5.

When $CPHA = 0$ for a slave, the falling edge of \overline{SS} indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. After the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the falling edge of \overline{SS} . Any data written after the falling edge is stored in the transmit data register and transferred to the shift register after the current transmission.

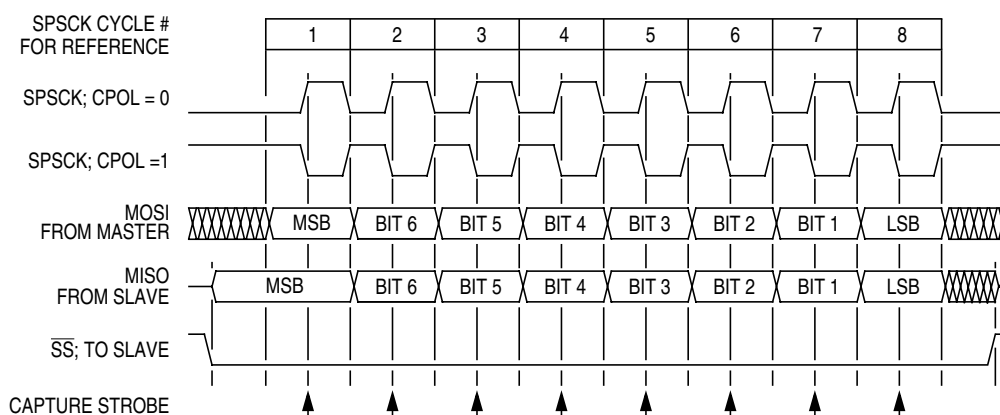


Figure 15-4. Transmission Format (CPHA = 0)

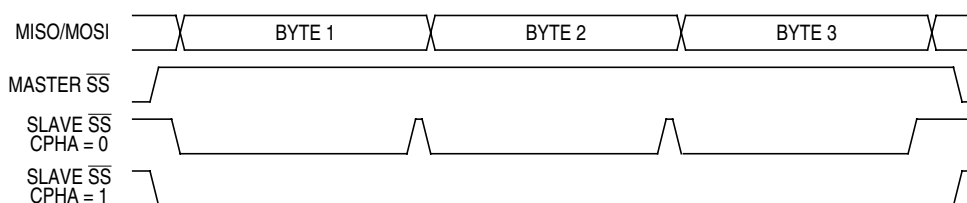


Figure 15-5. CPHA/ \overline{SS} Timing

15.3.3.3 Transmission Format When $CPHA = 1$

Figure 15-6 shows an SPI transmission in which $CPHA = 1$. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SPSCCK: one for $CPOL = 0$ and another for $CPOL = 1$. The diagram may be interpreted as a master or slave timing diagram because the serial clock (SPSCCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The \overline{SS} line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input (\overline{SS}) is low, so that only the selected slave drives to the master. The \overline{SS} pin of the master is not shown but is assumed to be inactive. The \overline{SS}

\$00FF (255) to the TIM2 counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is 000. See 17.8.1 TIM2 Status and Control Register.

The value in the TIM2 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 TIM2 channel registers produces a duty cycle of 128/256 or 50%.

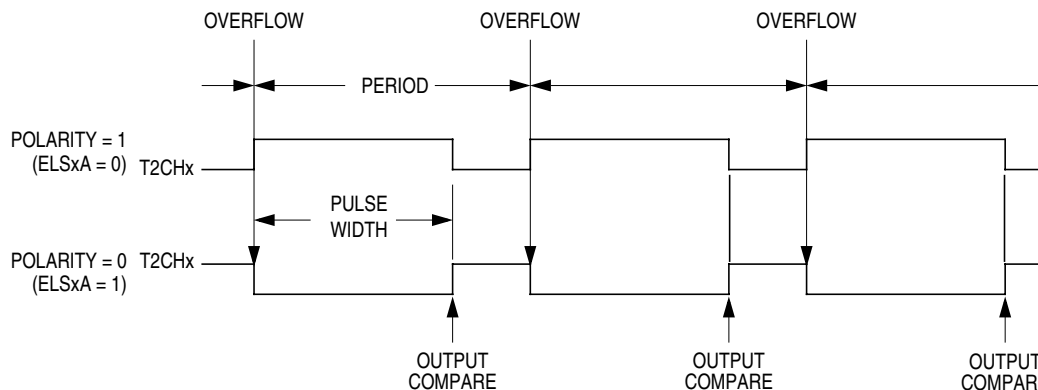


Figure 17-3. PWM Period and Pulse Width

17.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 17.3.4 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 TIM2 channel registers.

An unsynchronized write to the TIM2 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 TIM2 overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM2 may pass the new value before it is written to the timer channel (T2CHxH:T2CHxL).

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 TIM2 overflow interrupts and write the new value in the TIM2 overflow interrupt routine. The TIM2 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.

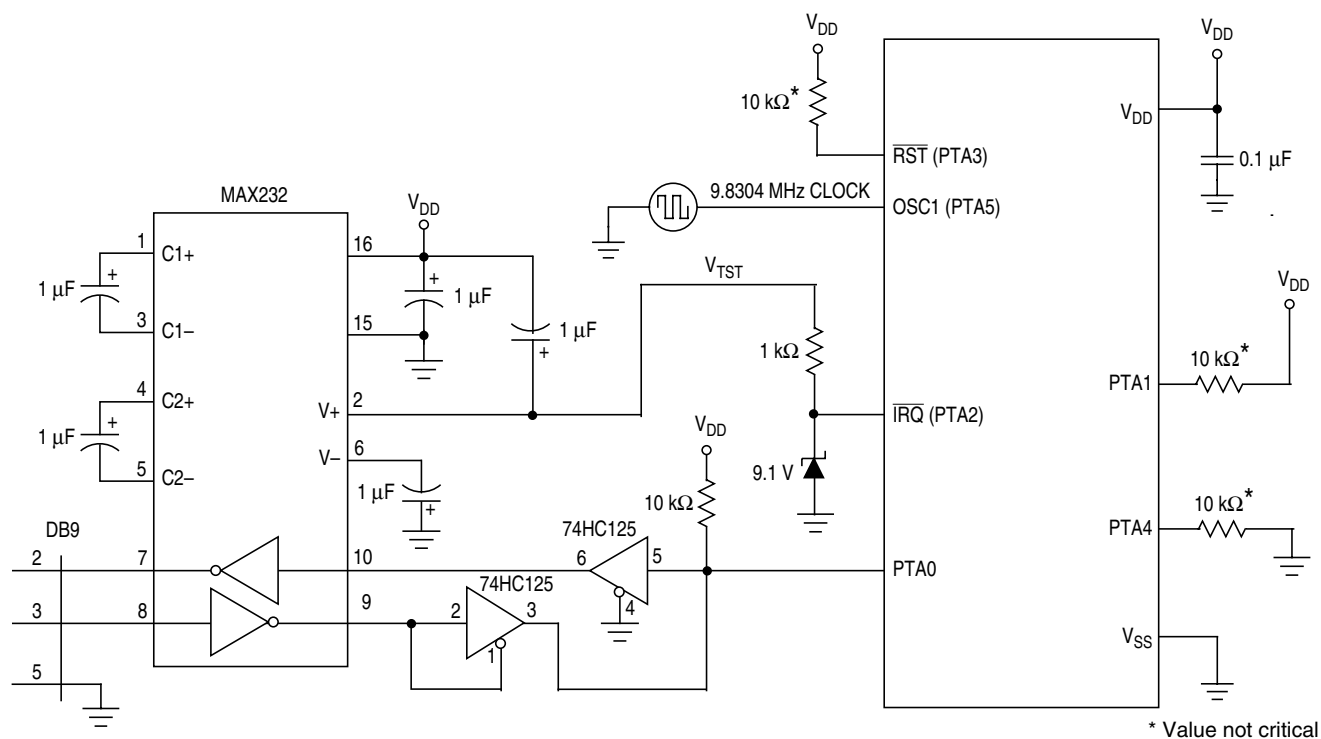


Figure 18-10. Monitor Mode Circuit (External Clock, with High Voltage)

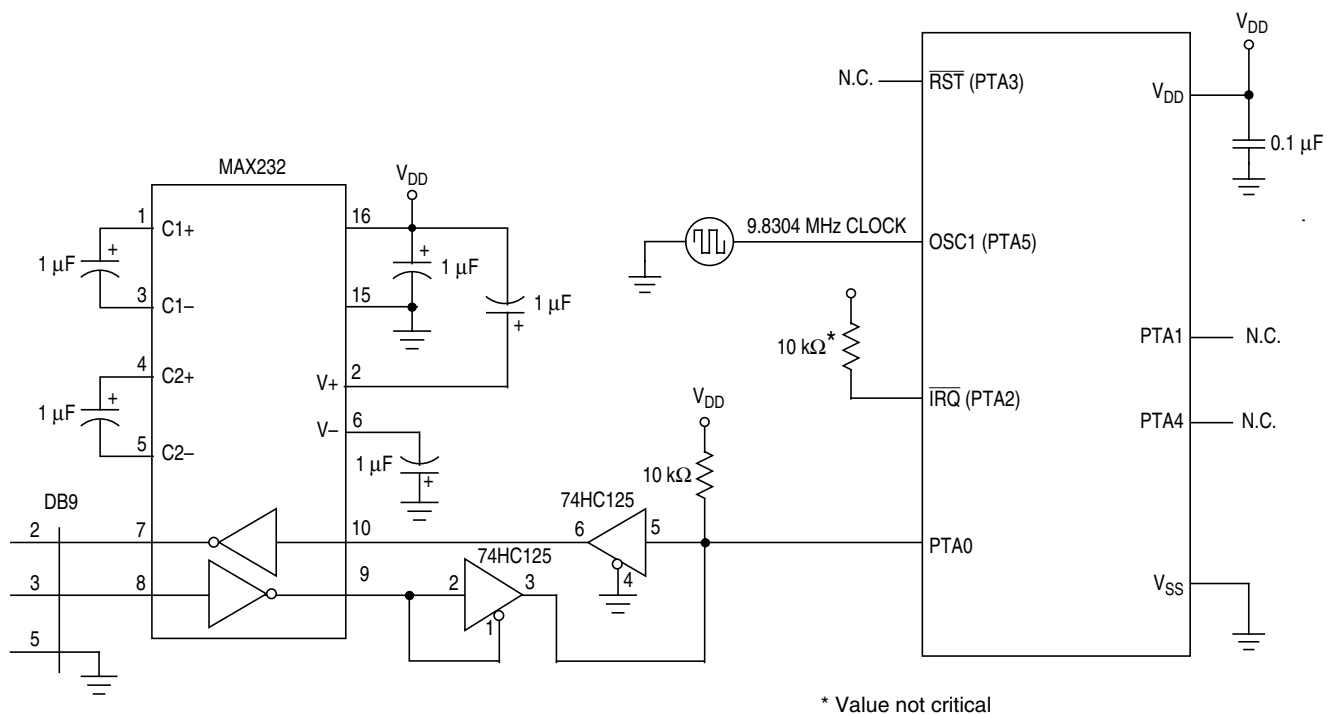


Figure 18-11. Monitor Mode Circuit (External Clock, No High Voltage)



NOTES:

1. CONTROLLING DIMENSION: MILLIMETER

2. DIMENSIONS AND TOLERANCES PER ANSI Y14.5M-1982.

3. DIMENSION DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. MOLD FLASH OR GATE BURRS SHALL NOT EXCEED 0.15 PER SIDE.

4. DIMENSION DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION. INTERLEAD FLASH OR PROTRUSION SHALL NOT EXCEED 0.25 PER SIDE.

5. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 TOTAL IN EXCESS OF THE DIMENSION AT MAXIMUM MATERIAL CONDITION.

6. TERMINAL NUMBERS ARE SHOWN FOR REFERENCE ONLY.

7. DIMENSIONS ARE TO BE DETERMINED AT DATUM PLANE -W-.

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: 20 LD TSSOP, PITCH 0.65MM	DOCUMENT NO: 98ASH70169A	REV: C	
	CASE NUMBER: 948E-02	25 MAY 2005	
	STANDARD: JEDEC		