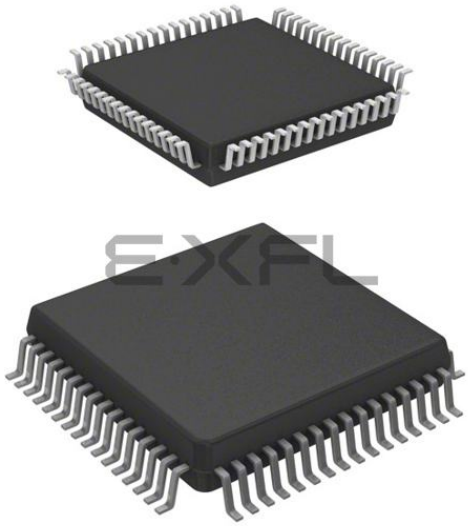


Welcome to E-XFL.COM

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	CANbus, SCI, SPI
Peripherals	LVD, POR, PWM
Number of I/O	40
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 15x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-QFP
Supplier Device Package	64-QFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908az32acfue

Features of the CPU08 include:

- 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
- C Language Support

1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908AZ32A.

2.4 Vector Addresses and Priority

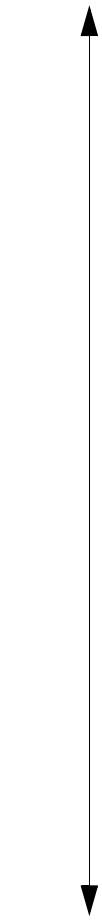
Addresses in the range $FFCC to $FFFF contain the user-specified vector locations. The vector addresses are shown in Table 2-1. It is recommended that all vector addresses are defined.

Table 2-1. Vector Addresses

	Address	Vector Description
Lowest Priority	$FFCC	TIMA Channel 5 Vector (High)
	$FFCD	TIMA Channel 5 Vector (Low)
	$FFCE	TIMA Channel 4 Vector (High)
	$FFCF	TIMA Channel 4 Vector (Low)
	$FFD0	ADC Vector (High)
	$FFD1	ADC Vector (Low)
	$FFD2	Keyboard Vector (High)
	$FFD3	Keyboard Vector (Low)
	$FFD4	SCI Transmit Vector (High)
	$FFD5	SCI Transmit Vector (Low)
	$FFD6	SCI Receive Vector (High)
	$FFD7	SCI Receive Vector (Low)
	$FFD8	SCI Error Vector (High)
	$FFD9	SCI Error Vector (Low)
	$FFDA	CAN Transmit Vector (High)
	$FFDB	CAN Transmit Vector (Low)
	$FFDC	CAN Receive Vector (High)
	$FFDD	CAN Receive Vector (Low)
	$FFDE	CAN Error Vector (High)
	$FFDF	CAN Error Vector (Low)
	$FFE0	CAN Wakeup Vector (High)
	$FFE1	CAN Wakeup Vector (Low)
	$FFE2	SPI Transmit Vector (High)
	$FFE3	SPI Transmit Vector (Low)
	$FFE4	SPI Receive Vector (High)
	$FFE5	SPI Receive Vector (Low)
	$FFE6	TIMB Overflow Vector (High)
	$FFE7	TIMB Overflow Vector (Low)
	$FFE8	TIMB CH1 Vector (High)
	$FFE9	TIMB CH1 Vector (Low)

— Continued on next page

Table 2-1. Vector Addresses (Continued)



Address	Vector Description
$FFEA	TIMB CH0 Vector (High)
$FFEB	TIMB CH0 Vector (Low)
$FFEC	TIMA Overflow Vector (High)
$FFED	TIMA Overflow Vector (Low)
$FFEE	TIMA CH3 Vector (High)
$FFEF	TIMA CH3 Vector (Low)
$FFF0	TIMA CH2 Vector (High)
$FFF1	TIMA CH2 Vector (Low)
$FFF2	TIMA CH1 Vector (High)
$FFF3	TIMA CH1 Vector (Low)
$FFF4	TIMA CH0 Vector (High)
$FFF5	TIMA CH0 Vector (Low)
$FFF6	PIT Vector (High)
$FFF7	PIT Vector (Low)
$FFF8	PLL Vector (High)
$FFF9	PLL Vector (Low)
$FFFA	IRQ1 Vector (High)
$FFFB	IRQ1 Vector (Low)
$FFFC	SWI Vector (High)
$FFFD	SWI Vector (Low)
$FFFE	Reset Vector (High)
$FFFF	Reset Vector (Low)

4.5 FLASH Mass Erase Operation

Use this step-by-step procedure to erase the entire FLASH memory to read as logic 1:

1. Set both the ERASE bit and the MASS bit in the FLASH Control Register (FLCR).
2. Read the FLASH Block Protect Register (FLBPR).
3. Write to any FLASH address within the FLASH array with any data.

NOTE

If the address written to in Step 3 is within address space protected by the FLASH Block Protect Register (FLBPR), no erase will occur.

4. Wait for a time, t_{NVS}.
5. Set the HVEN bit.
6. Wait for a time, t_{MERASE}.
7. Clear the ERASE bit.
8. Wait for a time, t_{NVHL}.
9. Clear the HVEN bit.
10. Wait for a time, t_{RCV}, after which the memory can be accessed in normal read mode.

NOTE

A. *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*
B. *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Care must be taken however to ensure that these operations do not access any address within the FLASH array memory space such as the COP Control Register (COPCTL) at $FFFF.*
C. *It is highly recommended that interrupts be disabled during program/erase operations.*

4.6 FLASH Page Erase Operation

Use this step-by-step procedure to erase a page (128 bytes) of FLASH memory to read as logic 1:

1. Set the ERASE bit and clear the MASS bit in the FLASH Control Register (FLCR).
2. Read the FLASH Block Protect Register (FLBPR).
3. Write any data to any FLASH address within the address range of the page (128 byte block) to be erased.
4. Wait for time, t_{NVS}.
5. Set the HVEN bit.
6. Wait for time, t_{ERASE}.
7. Clear the ERASE bit.
8. Wait for time, t_{NVH}.
9. Clear the HVEN bit.
10. Wait for a time, t_{RCV}, after which the memory can be accessed in normal read mode.

NOTE

A. *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

C. *It is highly recommended that interrupts be disabled during program/erase operations.*
D. *Do not exceed* t_{PROG} *maximum or* t_{HV} *maximum.* t_{HV} *is defined as the cumulative high voltage programming time to the same row before next erase.* t_{HV} *must satisfy this condition:* t_{NVS} *+* t_{NVH} *+* t_{PGS} *+ (*t_{PROG}*X 64) ð* t_{HV} *max. Please also see 25.1.14 FLASH Memory Characteristics.*
E. *The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH address programmed to clearing the PGM bit (step 7 to step 10) must not exceed the maximum programming time,* t_{PROG} *max.*
F. *Be cautious when programming the FLASH array to ensure that non-FLASH locations are not used as the address that is written to when selecting either the desired row address range in step 3 of the algorithm or the byte to be programmed in step 7 of the algorithm. This applies particularly to:*

- $FFCC-$FFFF: User Vector Area

4.8 Low-Power Modes

The WAIT and STOP instructions will place the MCU in low power consumption standby modes.

4.8.1 WAIT Mode

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. Wait mode will suspend any FLASH program/erase operations and leave the memory in a Standby Mode.

4.8.2 STOP Mode

Putting the MCU into stop mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. Stop mode will suspend any FLASH program/erase operations and leave the memory in a Standby Mode.

NOTE

Standby Mode is the power saving mode of the FLASH module, in which all internal control signals to the FLASH are inactive and the current consumption of the FLASH is minimum.

5.4.5.2 EEPROM Programming

The unprogrammed or erase state of an EEPROM bit is a logic 1. Programming changes the state to a logic 0. Only EEPROM bytes in the non-protected blocks and the EENVR register can be programmed.

Use the following procedure to program a byte of EEPROM:

1. Clear EERAS1 and EERAS0 and set EELAT in the EECR.(A)

NOTE

If using the AUTO mode, also set the AUTO bit during Step 1.

2. Write the desired data to the desired EEPROM address.(B)
3. Set the EEPGM bit.(C) Go to Step 7 if AUTO is set.
4. Wait for time, t_{EEPGM}, to program the byte.
5. Clear EEPGM bit.
6. Wait for time, t_{EEFPV}, for the programming voltage to fall. Go to Step 8.
7. Poll the EEPGM bit until it is cleared by the internal timer.(D)
8. Clear EELAT bits.(E)

NOTE

A. *EERAS1 and EERAS0 must be cleared for programming. Setting the EELAT bit configures the address and data buses to latch data for programming the array. Only data with a valid EEPROM address will be latched. If EELAT is set, other writes to the EECR will be allowed after a valid EEPROM write.*

B. *If more than one valid EEPROM write occurs, the last address and data will be latched overriding the previous address and data. Once data is written to the desired address, do not read EEPROM locations other than the written location. (Reading an EEPROM location returns the latched data and causes the read address to be latched).*

C. *The EEPGM bit cannot be set if the EELAT bit is cleared or a non-valid EEPROM address is latched. This is to ensure proper programming sequence. Once EEPGM is set, do not read any EEPROM locations; otherwise, the current program cycle will be unsuccessful. When EEPGM is set, the on-board programming sequence will be activated.*

D. *The delay time for the EEPGM bit to be cleared in AUTO mode is less than t_{EEPGM}. However, on other MCUs, this delay time may be different. For forward compatibility, software should not make any dependency on this delay time.*

E. *Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.*

5.5.4 EEPROM Timebase Divider Register

The 16-bit EEPROM timebase divider register consists of two 8-bit registers: EEDIVH and EEDIVL. The 11-bit value in this register is used to configure the timebase divider circuit to obtain the 35 μs timebase for EEPROM control.

These two read/write registers are respectively loaded with the contents of the EEPROM timebase divider onvolatile registers (EEDIVHNVR and EEDIVLNVR) after a reset.

Address: $FE1A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIVS-ECD	0	0	0	0	EEDIV10	EEDIV9	EEDIV8
Write:								
Reset:	Contents of EEDIVHNVR ($FE10), Bits [6:3] = 0							

= Unimplemented

Figure 5-5. EEDIV Divider High Register (EEDIVH)

Address: $FE1B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
Write:								
Reset:	Contents of EEDIVLNVR ($FE11)							

Figure 5-6. EEDIV Divider Low Register (EEDIVL)

EEDIVSECD — EEPROM Divider Security Disable

This bit enables/disables the security feature of the EEDIV registers. When EEDIV security feature is enabled, the state of the registers EEDIVH and EEDIVL are locked (including EEDIVSECD bit). The EEDIVHNVR and EEDIVLNVR nonvolatile memory registers are also protected from being erased/programmed.

1 = EEDIV security feature disabled
0 = EEDIV security feature enabled

EEDIV[10:0] — EEPROM timebase prescaler

These prescaler bits store the value of EEDIV which is used as the divisor to derive a timebase of 35μs from the selected reference clock source (CGMXCLK or bus block in the CONFIG-2 register) for the EEPROM related internal timer and circuits. EEDIV[10:0] bits are readable at any time. They are writable when EELAT = 0 and EEDIVSECD = 1.

The EEDIV value is calculated by the following formula:

EEDIV= INT[Reference Frequency(Hz) x 35 x10^{-6} +0.5]

Where the result inside the bracket is rounded down to the nearest integer value

For example, if the reference frequency is 4.9152MHz, the EEDIV value is 172

NOTE

Programming/erasing the EEPROM with an improper EEDIV value may result in data lost and reduce endurance of the EEPROM device.

5.6.2 Stop Mode

The STOP instruction reduces the EEPROM power consumption to a minimum. The STOP instruction should not be executed while a programming or erasing sequence is in progress.

If stop mode is entered while EELAT and EEPGM are set, the programming sequence will be stopped and the programming voltage to the EEPROM array removed. The programming sequence will be restarted after leaving stop mode; access to the EEPROM is only possible after the programming sequence has completed.

If stop mode is entered while EELAT and EEPGM is cleared, the programming sequence will be terminated abruptly.

In either case, the data integrity of the EEPROM is not guaranteed.

8.9.4 Reaction Time Calculation

The actual acquisition and lock times can be calculated using the equations below. These equations yield nominal values under the following conditions:

- Correct selection of filter capacitor, C_F (see 8.9.3 Choosing a Filter Capacitor).
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

The K factor in the equations is derived from internal PLL parameters. K_{acq} is the K factor when the PLL is configured in acquisition mode, and K_{trk} is the K factor when the PLL is configured in tracking mode. (See 8.3.2.2 Acquisition and Tracking Modes).

$$t_{acq} = \left(\frac{V_{DDA}}{f_{CGMRDV}}\right)\left(\frac{8}{K_{ACQ}}\right)$$

$$t_{al} = \left(\frac{V_{DDA}}{f_{CGMRDV}}\right)\left(\frac{4}{K_{TRK}}\right)$$

$$t_{Lock} = t_{ACQ} + t_{AL}$$

Note the inverse proportionality between the lock time and the reference frequency.

In automatic bandwidth control mode, the acquisition and lock times are quantized into units based on the reference frequency. (See 8.3.2.3 Manual and Automatic PLL Bandwidth Modes). A certain number of clock cycles, n_{ACQ}, is required to ascertain that the PLL is within the tracking mode entry tolerance, Δ_{TRK}, before exiting acquisition mode. A certain number of clock cycles, n_{TRK}, is required to ascertain that the PLL is within the lock mode entry tolerance, Δ_{Lock}. Therefore, the acquisition time, t_{ACQ}, is an integer multiple of n_{ACQ}/f_{CGMRDV}, and the acquisition to lock time, t_{AL}, is an integer multiple of n_{TRK}/f_{CGMRDV}. Also, since the average frequency over the entire measurement period must be within the specified tolerance, the total time usually is longer than t_{Lock} as calculated above.

In manual mode, it is usually necessary to wait considerably longer than t_{Lock} before selecting the PLL clock (see 8.3.3 Base Clock Selector Circuit), because the factors described in 8.9.2 Parametric Influences on Reaction Time, may slow the lock time considerably.

When defining a limit in software for the maximum lock time, the value must allow for variation due to all of the factors mentioned in this section, especially due to the C_F capacitor and application specific influences.

The calculated lock time is only an indication and it is the customer's responsibility to allow enough of a guard band for their application. Prior to finalizing any software and while determining the maximum lock time, take into account all device to device differences. Typically, applications set the maximum lock time as an order of magnitude higher than the measured value. This is considered sufficient for all such device to device variation.

Freescale recommends measuring the lock time of the application system by utilizing dedicated software, running in FLASH, EEPROM or RAM. This should toggle a port pin when the PLL is first configured and switched on, then again when it goes from acquisition to lock mode and finally again when the PLL lock

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
SCI Control Register 1 (SCC1)	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Control Register 2 (SCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Control Register 3 (SCC3)	Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
	Write:								
	Reset:	U	U	0	0	0	0	0	0
SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
	Write:								
	Reset:	1	1	0	0	0	0	0	0
SCI Status Register 2 (SCS2)	Read:	0	0	0	0	0	0	BKF	RPF
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Data Register (SCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Unaffected by Reset							
SCI Baud Rate Register (SCBR)	Read:	0	0	SCP1	SCP0	R	SCR2	SCR1	SCR0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented U = Unaffected R = Reserved

Figure 16-7. SCI I/O Receiver Register Summary

Table 16-4. SCI Receiver I/O Address Summary

Register	SCC1	SCC2	SCC3	SCS1	SCS2	SCDR	SCBR
Address	$0013	$0014	$0015	$0016	$0017	$0018	$0019

FE — Receiver Framing Error Bit

This clearable, read-only bit is set when a logic 0 is accepted as the stop bit. FE generates an SCI error CPU interrupt request if the FEIE bit in SCC3 also is set. Clear the FE bit by reading SCS1 with FE set and then reading the SCDR. Reset clears the FE bit.

1 = Framing error detected
0 = No framing error detected

PE — Receiver Parity Error Bit

This clearable, read-only bit is set when the SCI detects a parity error in incoming data. PE generates a PE CPU interrupt request if the PEIE bit in SCC3 is also set. Clear the PE bit by reading SCS1 with PE set and then reading the SCDR. Reset clears the PE bit.

1 = Parity error detected
0 = No parity error detected

16.8.5 SCI Status Register 2

SCI status register 2 contains flags to signal the following conditions:

- Break character detected
- Incoming data

Address: $0017

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

= Unimplemented

Figure 16-16. SCI Status Register 2 (SCS2)

BKF — Break Flag Bit

This clearable, read-only bit is set when the SCI detects a break character on the RxD pin. In SCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in SCC3 is cleared. BKF does not generate a CPU interrupt request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after logic 1s again appear on the RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

RPF — Reception in Progress Flag Bit

This read-only bit is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RPF does not generate an interrupt request. RPF is reset after the receiver detects false start bits (usually from noise or a baud rate mismatch), or when the receiver detects an idle character. Polling RPF before disabling the SCI module or entering stop mode can show whether a reception is in progress.

1 = Reception in progress
0 = No reception in progress

CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See Figure 17-4 and Figure 17-5.) To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be set to logic 1 between bytes. (See Figure 17-11). Reset sets the CPHA bit.

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. Once the transmission begins, no new data is allowed into the shift register from the data register. Therefore, the slave data register must be loaded with the desired transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the data register and transferred to the shift register at the current transmission.

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. The same applies when $\overline{SS}$ is high for a slave. The MISO pin is held in a high-impedance state, and the incoming SPSCK is ignored. In certain cases, it may also cause the MODF flag to be set. (See 17.6.2 Mode Fault Error). A logic 1 on the $\overline{SS}$ pin does not in any way affect the state of the SPI state machine.

SPWOM — SPI Wired-OR Mode Bit

This read/write bit disables the pullup devices on pins SPSCK, MOSI, and MISO so that those pins become open-drain outputs.

1 = Wired-OR SPSCK, MOSI, and MISO pins
0 = Normal push-pull SPSCK, MOSI, and MISO pins

SPE — SPI Enable Bit

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI (see 17.9 Resetting the SPI). Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE — SPI Transmit Interrupt Enable Bit

This read/write bit enables CPU interrupt requests generated by the SPTE bit. SPTE is set when a byte transfers from the transmit data register to the shift register. Reset clears the SPTIE bit.

1 = SPTE CPU interrupt requests enabled
0 = SPTE CPU interrupt requests disabled

17.13.2 SPI Status and Control Register

The SPI status and control register contains flags to signal the following conditions:

- Receive data register full
- Failure to clear SPRF bit before next byte is received (overflow error)
- Inconsistent logic level on $\overline{SS}$ pin (mode fault error)
- Transmit data register empty

The SPI status and control register also contains bits that perform these functions:

- Enable error interrupts
- Enable mode fault error detection
- Select master SPI baud rate

18.3.1 TIMA Counter Prescaler

The TIMA clock source can be one of the seven prescaler outputs or the TIMA clock pin, PTD6/ATD14/TCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMA status and control register select the TIMA clock source.

18.3.2 Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TASC0 through TASC5 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMA latches the contents of the TIMA counter into the TIMA channel registers, TACHxH–TACHxL. Input captures can generate TIMA CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The result obtained by an input capture will be two more than the value of the free-running counter on the rising edge of the internal bus clock preceding the external transition. This delay is required for internal synchronization.

The free-running counter contents are transferred to the TIMA channel register (TACHxH–TACHxL see 18.8.5 TIMA Channel Registers) on each proper signal transition regardless of whether the TIMA channel flag (CH0F–CH5F in TASC0–TASC5 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see 18.8.5 TIMA Channel Registers). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the TIMA channel register (TACHxH–TACHxL).

18.3.3 Output Compare

With the output compare function, the TIMA can generate a periodic pulse with a programmable polarity, duration and frequency. When the counter reaches the value in the registers of an output compare channel, the TIMA can set, clear or toggle the channel pin. Output compares can generate TIMA CPU interrupt requests.

20.7.2 TIM Counter Registers

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

NOTE

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

Address: $004C

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

Address: $004D

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

= Unimplemented

Figure 20-4. PIT Counter Registers (PCNTH–PCNTL)

20.7.3 TIM Counter Modulo Registers

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

Address: $004E:$004F

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

Address: $004E:$004F

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

Figure 20-5. PIT Counter Modulo Registers (PMODH–PMODL)

NOTE

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

one conversion will take between 16 and 17 μs and there will be between 128 bus cycles between each conversion. Sample rate is approximately 60 kHz.

Refer to 25.1.6 ADC Characteristics.

$$\text{Conversion Time} = \frac{\text{16 to 17 ADC Clock Cycles}}{\text{ADC Clock Frequency}}$$

$$\text{Number of Bus Cycles} = \text{Conversion Time} \times \text{Bus Frequency}$$

21.3.4 Continuous Conversion

In the continuous conversion mode, the ADC data register will be filled with new data after each conversion. Data from the previous conversion will be overwritten whether that data has been read or not. Conversions will continue until the ADCO bit (ADC status control register, $0038) is cleared. The COCO bit is set after the first conversion and will stay set for the next several conversions until the next write of the ADC status and control register or the next read of the ADC data register.

21.3.5 Accuracy and Precision

The conversion process is monotonic and has no missing codes. See 25.1.6 ADC Characteristics for accuracy information.

21.4 Interrupts

When the AIEN bit is set, the ADC module is capable of generating a CPU interrupt after each ADC conversion. A CPU interrupt is generated if the COCO bit (ADC status control register, $0038) is at logic 0. If the COCO bit is set, an interrupt is generated. The COCO bit is not used as a conversion complete flag when interrupts are enabled.

21.5 Low-Power Modes

The following subsections describe the low-power modes.

21.5.1 Wait Mode

The ADC continues normal operation during wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting the ADCH[4:0] bits in the ADC status and control register before executing the WAIT instruction.

21.5.2 Stop Mode

The ADC module is inactive after the execution of a STOP instruction. Any pending conversion is aborted. ADC conversions resume when the MCU exits stop mode. Allow one conversion cycle to stabilize the analog circuitry before attempting a new ADC conversion after exiting stop mode.

Figure 23-16 shows the port E I/O logic.

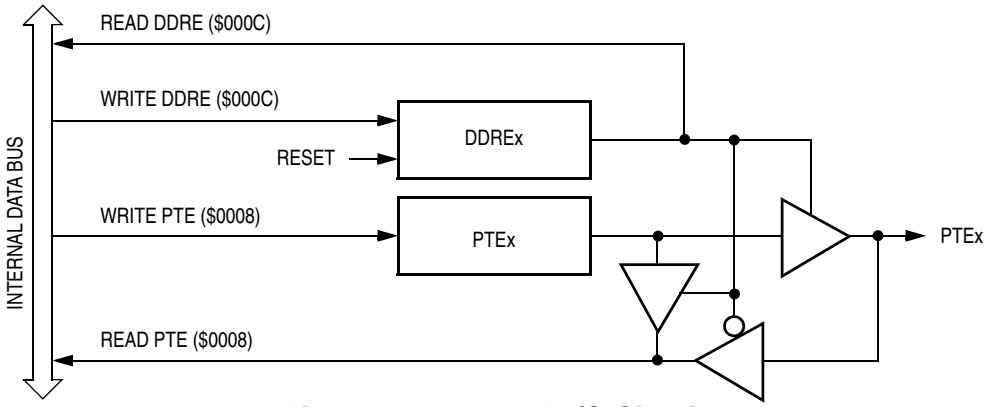


Figure 23-16. Port E I/O Circuit

When bit DDREx is a logic 1, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic 0, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 23-5 summarizes the operation of the port E pins.

Table 23-5. Port E Pin Functions

DDRE Bit	PTE Bit	I/O Pin Mode	Accesses to DDRE	Accesses to PTE	
			Read/Write	Read	Write
0	X	Input, Hi-Z	DDRE[7:0]	Pin	PTE[7:0](1)
1	X	Output	DDRE[7:0]	PTE[7:0]	PTE[7:0]

X = don't care
Hi-Z = high impedance
1. Writing affects data register, but does not affect input.

23.9.2 Data Direction Register H

Data direction register H determines whether each port H pin is an input or an output. Writing a logic 1 to a DDRH bit enables the output buffer for the corresponding port H pin; a logic 0 disables the output buffer.

Address: $000F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	DDRH1	DDRH0
Write:	R	R	R	R	R	R		
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 23-24. Data Direction Register H (DDRH)

DDRH[1:0] — Data Direction Register H Bits

These read/write bits control port H data direction. Reset clears DDRG[1:0], configuring all port H pins as inputs.

1 = Corresponding port H pin configured as output
0 = Corresponding port H pin configured as input

NOTE

Avoid glitches on port H pins by writing to the port H data register before changing data direction register H bits from 0 to 1.

Figure 23-25 shows the port H I/O logic.

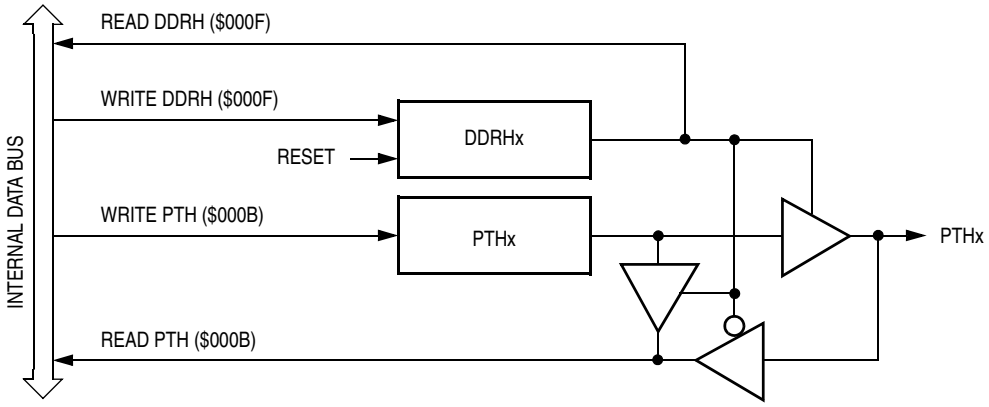
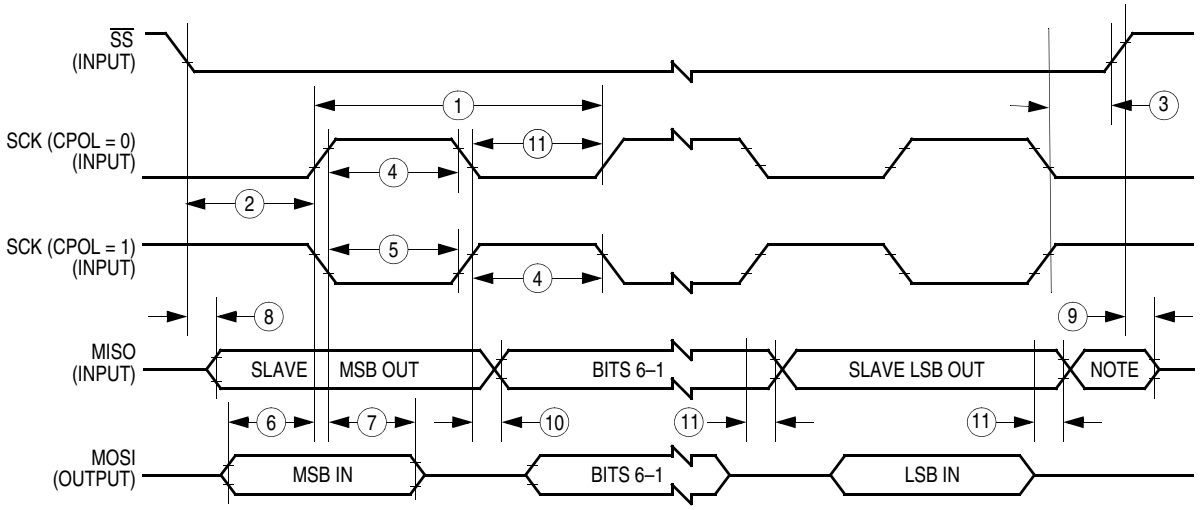
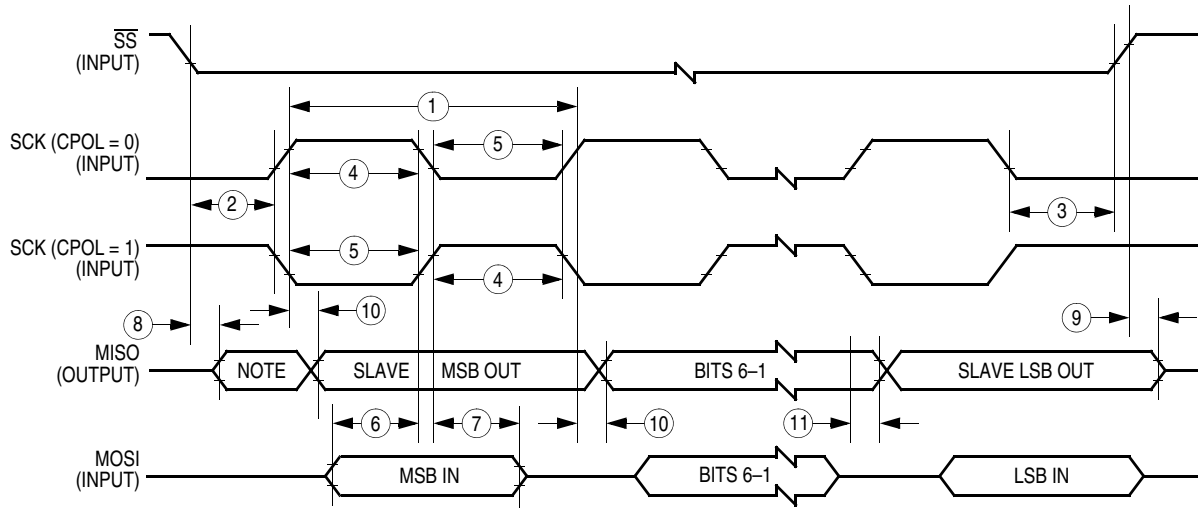


Figure 23-25. Port H I/O Circuit



NOTE: Not defined but normally MSB of character just received

a) SPI Slave Timing (CPHA = 0)



NOTE: Not defined but normally LSB of character previously transmitted

b) SPI Slave Timing (CPHA = 1)

Figure 25-2. SPI Slave Timing Diagram