Zilog - EZ80L92AZ020EG Datasheet

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	eZ80
Core Size	8-Bit
Speed	20MHz
Connectivity	EBI/EMI, I²C, IrDA, SPI, UART/USART
Peripherals	DMA, WDT
Number of I/O	24
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	-
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/ez80l92az020eg

Document Disclaimer

Architectural Overview

ZiLOG's eZ80L92 MCU is a high-speed single-cycle instruction-fetch microcontroller with a maximum clock speed of 50 MHz. The eZ80L92 MCU is a member of eZ80Acclaim!® family of Flash microcontrollers. It operates in Z80® compatible addressing mode (64 KB) or full 24-bit addressing mode (16 MB). The rich peripheral set of the eZ80L92 MCU makes it suitable for various applications including industrial control, embedded communication, and point-of-sale terminals.

Features

The features of eZ80L92 MCU include:

- Single-cycle instruction fetch, high-performance, pipelined eZ80® CPU core[1]
- Low power features including SLEEP mode, HALT mode, and selective peripheral power-down control
- Two Universal Asynchronous Receiver/Transmitter (UARTs) with independent baud rate generators
- Serial Peripheral Interface (SPI) with independent clock rate generator
- Inter-Integrated Circuit (I^2C) with independent clock rate generator
- Infrared Data Association (IrDA)-compliant infrared encoder/decoder
- New DMA-like eZ80 instructions for efficient block data transfer
- Glueless external peripheral interface with 4 Chip Selects, individual Wait State generators, and an external $\overline{\text{WAIT}}$ input pin—supports Intel-style and Motorola-style buses Fixed-priority vectored interrupts (both internal and external) and interrupt controller
- Real-time clock with an on-chip 32 kHz oscillator, selectable 50/60 Hz input, and separate V_{DD} pin for battery backup
- Six 16-bit Counter/Timers with prescalers and direct input/output drive
- Watchdog Timer (WDT)
- 24 bits of General-Purpose Input/Output (GPIO)
- JTAG and ZDI debug interfaces
- 100-pin LQFP package
- 3.0 V to 3.6 V supply voltage with 5 V tolerant inputs

1. For simplicity, the term *eZ80 CPU* is referred as *CPU* for the rest of this document.

Block Diagram

Figure 1 illustrates the block diagram of the eZ80L92 MCU.

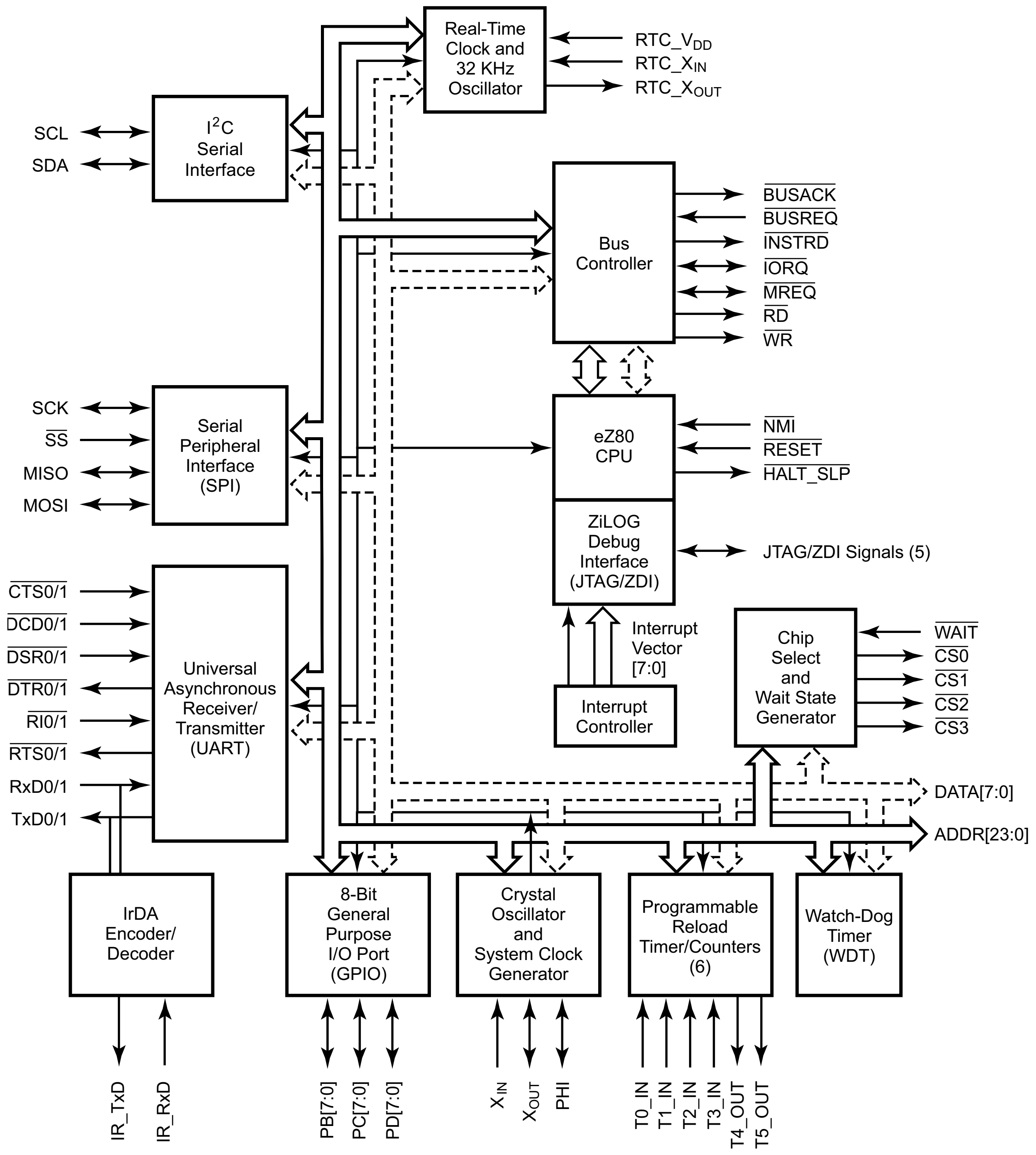

Figure 1. eZ80L92 Block Diagram

Pin Description

Figure 2 illustrates the pin layout of the eZ80L92 MCU in the 100-pin LQFP package. Table 1 lists the 100-Pin LQFP pins and their functions.

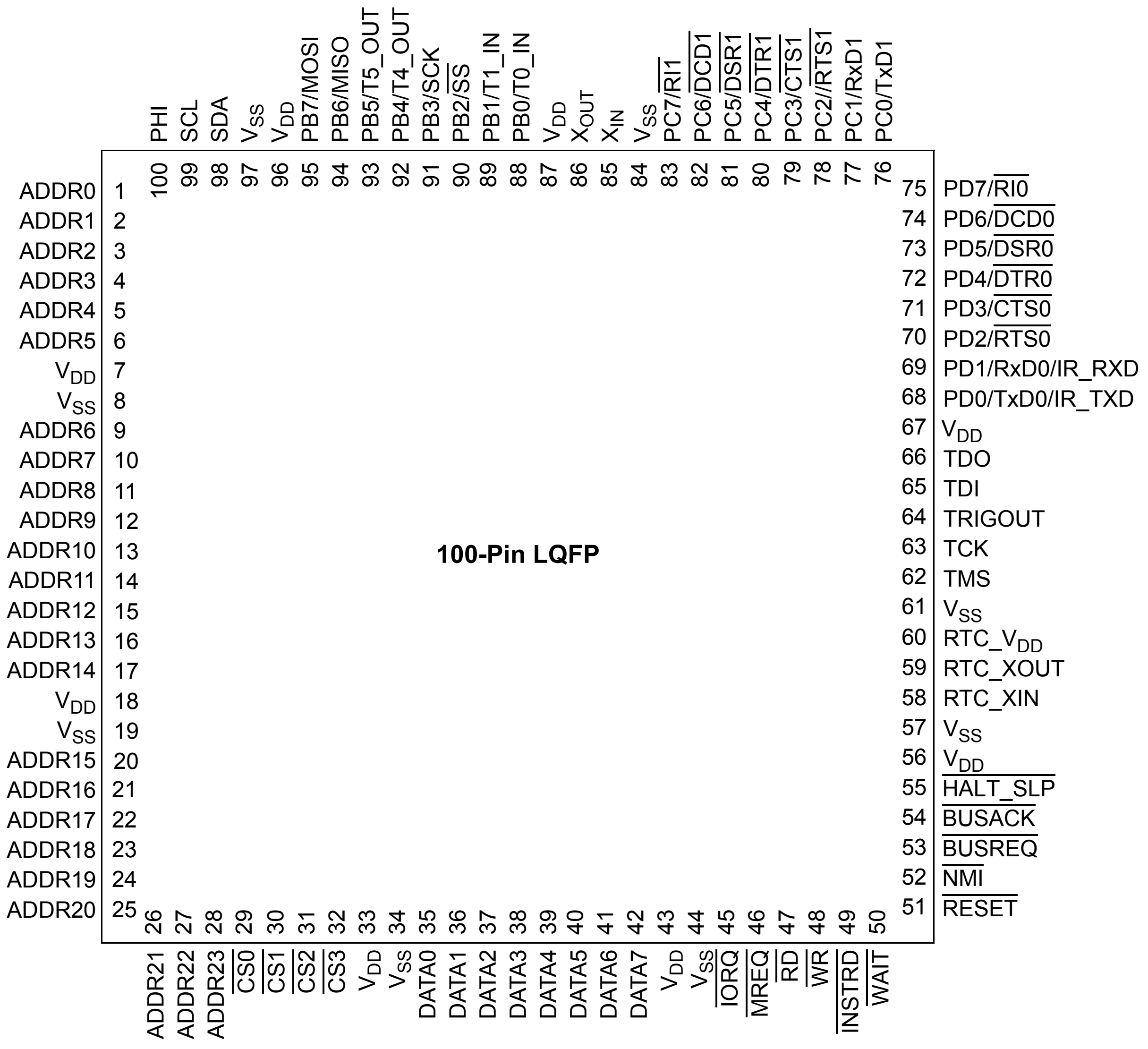

Figure 2. 100-Pin LQFP Configuration of the eZ80L92

Table 13. Register Values for Memory Chip Select Example in Figure 4

Chip Select	CS*x*_CTL[3] CS*x*_EN	CS*x*_CTL[4] CS*x*_IO	CS*x*_LBR	CS*x*_UBR	Description
CS0	1	0	00h	7Fh	CS0 is enabled as a Memory Chip Select. Valid addresses range from 000000h–7FFFFFh.
CS1	1	0	00h	9Fh	CS1 is enabled as a Memory Chip Select. Valid addresses range from 800000h–9FFFFFh.
CS2	1	0	A0h	CFh	CS2 is enabled as a Memory Chip Select. Valid addresses range from A00000h–CFFFFFh.
CS3	1	0	D0h	FFh	CS3 is enabled as a Memory Chip Select. Valid addresses range from D00000h–FFFFFFh.

I/O Chip Select Operation

I/O Chip Selects are active when the CPU is performing I/O instructions. As the I/O space is separate from the memory space in the eZ80L92 device, there can never be a conflict between I/O and memory addresses.

The eZ80L92 MCU supports a 16-bit I/O address. The I/O Chip Select logic decodes the High byte of the I/O address, ADDR[15:8]. Because the upper byte of the address bus, ADDR[23:16], is ignored, the I/O devices is always accessed from within any memory mode (ADL or Z80). The MBASE offset value used for setting the Z80 MEMORY mode page is also always ignored.

Four I/O Chip Selects are available with the eZ80L92. To generate a particular I/O Chip Select, the following conditions must be fulfilled:

- The Chip Select is enabled by setting CSX_EN to 1.
- The Chip Select is configured for I/O by setting CSx_IO to 1.
- An I/O Chip Select address match occurs—ADDR[15:8] = CS*x*_LBR[7:0].
- No higher-priority (lower-number) Chip Select meets the above conditions.
- The I/O address is not within the on-chip peripheral address range `0080h`–`00FFh`. On-chip peripheral registers assume priority for all addresses where:

  `0080h` ≤ ADDR[15:0] ≤ `00FFh`

- An I/O instruction must be executing.

During Write operations, Z80 Bus Mode employs 3 states (T1, T2, and T3) as described in Table 15.

Table 15. Z80® Bus Mode Write States

STATE T1	The Write cycle begins in State T1. The CPU drives the address onto the address bus, the associated Chip Select signal is asserted.
STATE T2	During State T2, the $\overline{\text{WR}}$ signal is asserted. Depending upon the instruction, either the $\overline{\text{MREQ}}$ or $\overline{\text{IORQ}}$ signal is asserted. If the external $\overline{\text{WAIT}}$ pin is driven Low at least one eZ80 system clock cycle prior to the end of State T2, additional WAIT states (T_{WAIT}) are asserted until the $\overline{\text{WAIT}}$ pin is driven High.
STATE T3	During State T3, no bus signals are altered.

Z80 Bus Mode Read and Write timing is illustrated in Figure 7 and Figure 8. The Z80 Bus Mode states can be configured for 1 to 15 eZ80 system clock cycles. In these figures, each Z80 Bus Mode state is two eZ80 system clock cycles in duration. Figure 7 and Figure 8 also illustrate the assertion of 1 WAIT state (T_{WAIT}) by the external peripheral during each Z80 Bus Mode cycle.

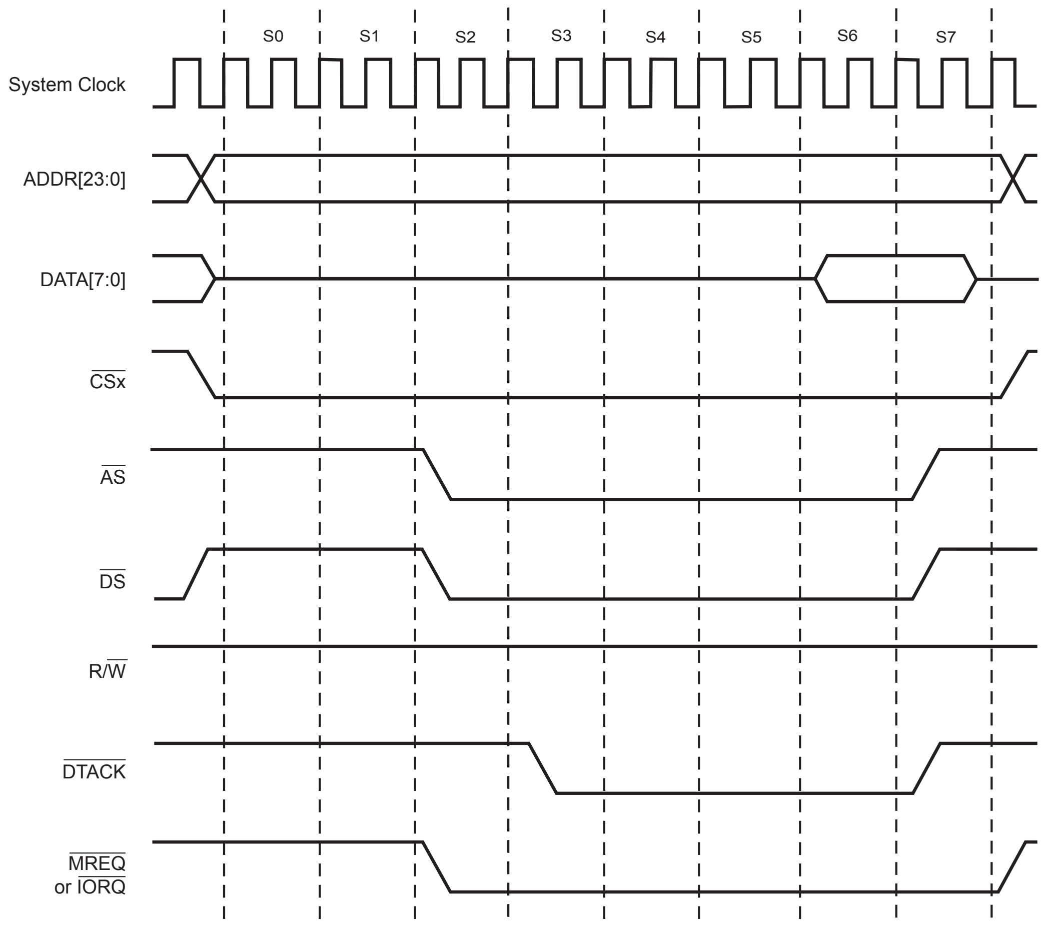

Figure 15. Motorola Bus Mode Read Timing Example

Watchdog Timer

The Watchdog Timer (WDT) helps protect against corrupt or unreliable software, power faults, and other system-level problems which may place the eZ80 CPU into unsuitable operating states. The eZ80L92 MCU WDT features:

- Four programmable time-out periods: 2^{18}, 2^{22}, 2^{25}, and 2^{27} clock cycles
- Two selectable WDT clock sources: the system clock or the real time clock source (on-chip 32 kHz crystal oscillator or 50/60 Hz signal)
- A selectable time-out response: a time-out can be configured to generate either a RESET or a non-maskable interrupt (NMI)
- A WDT time-out RESET indicator flag

Figure 17 illustrates the block diagram for the Watchdog Timer.

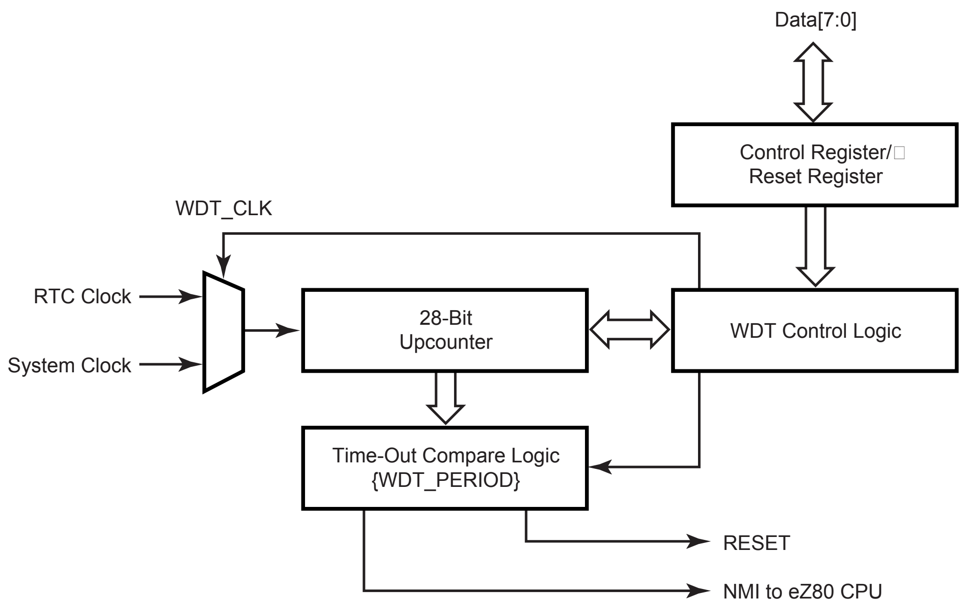

Figure 17. Watchdog Timer Block Diagram

Table 29. PRT SINGLE PASS Mode Operation Example

Parameter	Control Register(s)	Value
PRT Enabled	TMRx_CTL[0]	1
Reload and Restart Enabled	TMRx_CTL[1]	1
PRT Clock Divider = 4	TMRx_CTL[3:2]	00b
SINGLE PASS Mode	TMRx_CTL[4]	0
PRT Interrupt Enabled	TMRx_CTL[6]	1
PRT Reload Value	{TMRx_RR_H, TMRx_RR_L}	0004h

CONTINUOUS Mode

In CONTINUOUS mode, when the end-of-count value, `0000h`, is reached, the timer automatically reloads the 16-bit start value from the Timer Reload registers, TMRx_RR_H and TMRx_RR_L. Downcounting continues on the next clock edge. In CONTINUOUS mode, the PRT continues to count until disabled. An example of a PRT operating in CONTINUOUS mode is illustrated in Figure 20. Timer register information is indicated in Table 30.

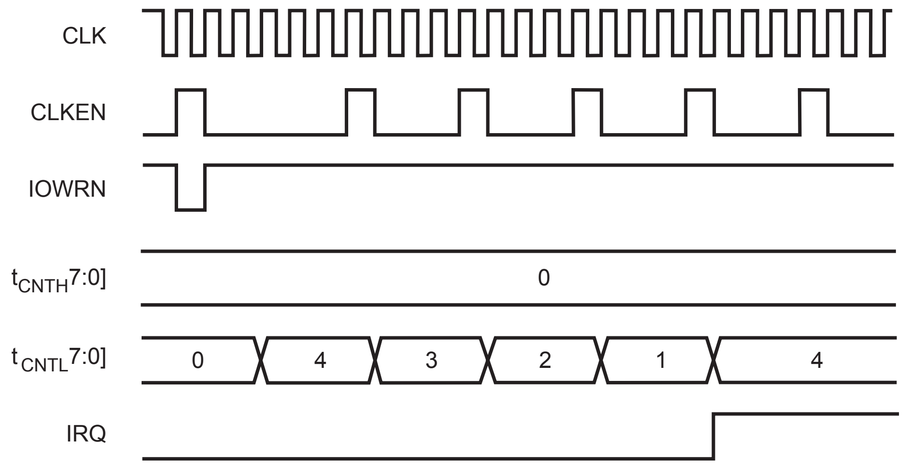

Figure 20. PRT CONTINUOUS Mode Operation Example

Table 30. PRT CONTINUOUS Mode Operation Example

Parameter	Control Register(s)	Value
PRT Enabled	TMRx_CTL[0]	1
Reload and Restart Enabled	TMRx_CTL[1]	1
PRT Clock Divider = 4	TMRx_CTL[3:2]	00b
CONTINUOUS Mode	TMRx_CTL[4]	1
PRT Interrupt Enabled	TMRx_CTL[6]	1
PRT Reload Value	{TMRx_RR_H, TMRx_RR_L}	0004h

Reading the Current Count Value

The CPU is capable of reading the current count value while the timer is running. This Read event does not affect timer operation. The High byte of the current count value is latched during a Read of the Low byte.

Timer Interrupts

The timer interrupt flag, PRT_IRQ, is set to 1 whenever the timer reaches its end-of-count value, `0000h`, in SINGLE PASS mode, or when the timer reloads the start value in CONTINUOUS mode. The interrupt flag is only set when the timer reaches `0000h` (or reloads) from `0001h`. The timer interrupt flag is not set to 1 when the timer is loaded with the value `0000h`, which selects the maximum time-out period.

The CPU can be programmed to poll the PRT_IRQ bit for the time-out event. Alternatively, an interrupt service request signal can be sent to the CPU by setting IRQ_EN to 1. Then, when the end-of-count value, `0000h`, is reached and PRT_IRQ is set to 1, an interrupt service request signal is passed to the CPU. PRT_IRQ is cleared to 0 and the interrupt service request signal is inactivated whenever the CPU reads from the timer control registers, TMRx_CTL.

Timer Input Source Selection

Timers 0–3 feature programmable input source selection. By default, the input is taken from the eZ80L92's system clock. Alternatively, Timers 0–3 can take their input from port input pins PB0 (Timers 0 and 2) or PB1 (Timers 1 and 3). Timers 0–3 can also use the Real-Time Clock clock source (50, 60, or 32768 Hz) as their clock sources. When the timer clock source is the Real-Time Clock signal, the timer decrements on the second rising edge of the system clock following the falling edge of the RTC_X_{OUT} pin. The input source for these timers is set using the Timer Input Source Select register.

Timer Reload Registers—High Byte

The Timer Reload Register—High Byte, detailed in Table 36, stores the most significant byte (MSB) of the 2-byte timer reload value. In CONTINUOUS mode, the timer reload value is reloaded into the timer upon end-of-count. When RST_EN (TMRx_CTL[1]) is set to 1 to enable the automatic reload and restart function, the timer reload value is written to the timer on the next rising edge of the clock.

Note: The Timer Data registers and Timer Reload registers share the same address space.

Table 36. Timer Reload Registers—High Byte (TMR0_RR_H = 0082h, TMR1_RR_H = 0085h, TMR2_RR_H = 0088h, TMR3_RR_H = 008Bh, TMR4_RR_H = 008Eh, or TMR5_RR_H = 0091h)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
CPU Access	W	W	W	W	W	W	W	W
Note: W = Write only.								

Bit Position	Value	Description
[7:0] TMRx_RR_H	00h–FFh	These bits represent the High byte of the 2-byte timer reload value, {TMRx_RR_H[7:0], TMRx_RR_L[7:0]}. Bit 7 is bit 15 (msb) of the 16-bit timer reload value. Bit 0 is bit 8 of the 16-bit timer reload value.

Timer Input Source Select Register

The Timer Input Source Select register, detailed in Table 37, sets the input source for Programmable Reload Timer 0–3 (TMR0, TMR1, TMR2, TMR3). Event frequency must be less than one-half of the system clock frequency. When configured for event inputs through the port pins, the Timers decrement on the fifth system clock rising edge following the rising edge of the port pin.

Real Time Clock Alarm Control Register

This register contains alarm enable bits for the real-time clock. The RTC_ACTRL register is cleared by a RESET. See Table 50.

Table 50. Real Time Clock Alarm Control Register (RTC_ACTRL = 00ECh)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	0	0
CPU Access	R	R	R	R	R/W	R/W	R/W	R/W
Note: X = Unchanged by RESET; R = Read-only; R/W = Read/Write.								

Bit Position	**Value**	**Description**
[7:4]	0000	Reserved.
3 ADOW_EN	0	The day-of-the-week alarm is disabled.
	1	The day-of-the-week alarm is enabled.
2 AHRS_EN	0	The hours alarm is disabled.
	1	The hours alarm is enabled.
1 AMIN_EN	0	The minutes alarm is disabled.
	1	The minutes alarm is enabled.
0 ASEC_EN	0	The seconds alarm is disabled.
	1	The seconds alarm is enabled.

Real-Time Clock Control Register

This register contains control and status bits for the real-time clock. Some bits in the RTC_CTRL register are cleared by a RESET. The ALARM flag and associated interrupt (if INT_EN is enabled) are cleared by reading this register. The ALARM flag is updated by clearing (locking) RTC_UNLOCK or by an increment of the RTC count. Writing to the RTC_CTRL register also resets the RTC count prescaler allowing the RTC to be synchronized to another time source.

SLP_WAKE indicates if an RTC alarm condition initiated the CPU recovery from SLEEP mode. This bit can be checked after RESET to determine if a sleep-mode recovery is caused by the RTC. SLP_WAKE is cleared by a Read of the RTC_CTRL register.

Setting BCD_EN causes the RTC to use BCD counting in all registers including the alarm set points.

Table 81. I2C Slave Address Register (I2C_SAR = 00C8h)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
CPU Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Note: R/W = Read/Write.								

Bit Position	Value	Description
[7:1] SLA	00h–7Fh	7-bit slave address or upper 2 bits,I2C_SAR[2:1], of address when operating in 10-bit mode.
0 GCE	0	I^{2}C not enabled to recognize the General Call Address.
	1	I^{2}C enabled to recognize the General Call Address.

I^{2}C Extended Slave Address Register

The I2C_XSAR register is used in conjunction with the I2C_SAR register to provide 10-bit addressing of the I^{2}C when in SLAVE mode. The I2C_SAR value forms the lower 8 bits of the 10-bit slave address. The full 10-bit address is supplied by {I2C_SAR[2:1], I2C_XSAR[7:0]}.

When the register receives an address starting with `F7h` to `F0h` (I2C_SAR[7:3] = 11110b), the I^{2}C recognizes that a 10-bit slave addressing mode is being selected. The I^{2}C sends an ACK after receiving the I2C_XSAR byte (the device does not generate an interrupt at this point). After the next byte of the address (I2C_XSAR) is received, the I^{2}C generates an interrupt and goes into SLAVE mode. Then I2C_SAR[2:1] are used as the upper 2 bits for the 10-bit extended address. The full 10-bit address is supplied by {I2C_SAR[2:1], I2C_XSAR[7:0]}. See Table 82.

Table 82. I^{2}C Extended Slave Address Register (I2C_XSAR = 00C9h)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
CPU Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Note: R/W = Read/Write.								

Bit Position	Value	Description
[6:3] M	0000–1111	I^2C clock divider scalar value.
[2:0] N	000–111	I^2C clock divider exponent.

The I^2C clocks are derived from the eZ80L92's system clock. The frequency of the eZ80L92 system clock is f_{SCK}. The I^2C bus is sampled by the I^2C block at the frequency f_{SAMP} supplied by:

$$f_{SAMP} = \frac{f_{SCLK}}{2^N}$$

In MASTER mode, the I^2C clock output frequency on SCL (f_{SCL}) is supplied by:

$$f_{SCL} = \frac{f_{SCLK}}{10 \bullet (M + 1)(2)^N}$$

The use of two separately-programmable dividers allows the MASTER mode output frequency to be set independently of the frequency at which the I^2C bus is sampled. This feature is particularly useful in multimaster systems because the frequency at which the I^2C bus is sampled must be at least 10 times the frequency of the fastest master on the bus to ensure that START and STOP conditions are always detected. By using two programmable clock divider stages, a high sampling frequency can be ensured while allowing the MASTER mode output to be set to a lower frequency.

Bus Clock Speed

The I^2C bus is defined for bus clock speeds up to 100 Kbps (400 Kbps in FAST mode).

To ensure correct detection of START and STOP conditions on the bus, the I^2C must sample the I^2C bus at least ten times faster than the bus clock speed of the fastest master on the bus. The sampling frequency should therefore be at least 1 MHz (4 MHz in FAST mode) to guarantee correct operation with other bus masters.

The I^2C sampling frequency is determined by the frequency of the eZ80L92 system clock and the value in the I2C_CCR bits 2 to 0. The bus clock speed generated by the I^2C in MASTER mode is determined by the frequency of the input clock and the values in I2C_CCR[2:0] and I2C_CCR[6:3].

ZDI Block Write

The Block Write operation is initiated in the same manner as the single-byte Write operation, but instead of terminating the Write operation after the first data byte is transferred, the ZDI master can continue to transmit additional bytes of data to the ZDI slave on the eZ80L92. After the receipt of each byte of data the ZDI register address increments by 1. If the ZDI register address reaches the end of the Write-Only ZDI register address space (`30h`), the address stops incrementing. Figure 41 illustrates the timing for ZDI Block Write operations.

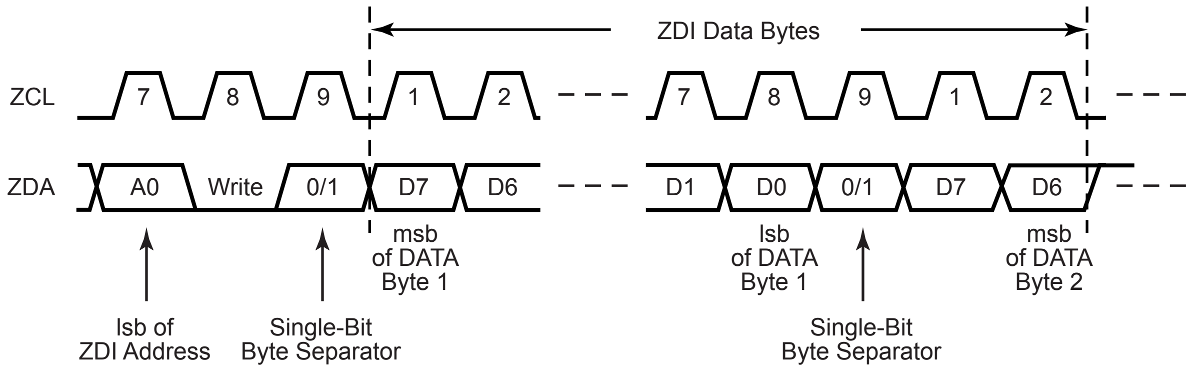

Figure 41. ZDI Block Data Write Timing

ZDI Read Operations

ZDI Single-Byte Read

Single-byte Read operations are initiated in the same manner as single-byte Write operations, with the exception that the R/$\overline{W}$ bit of the ZDI register address is set to 1. Upon receipt of a slave address with the R/$\overline{W}$ bit set to 1, the eZ80L92's ZDI block loads the selected data into the shifter at the beginning of the first cycle following the single-bit data separator. The most significant bit (msb) is shifted out first. Figure 42 illustrates the timing for ZDI single-byte Read operations.

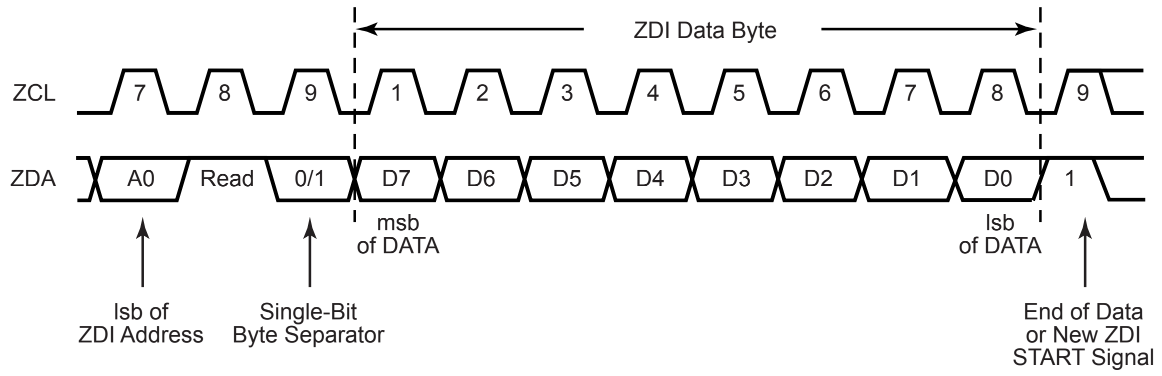

Figure 42. ZDI Single-Byte Data Read Timing

ZDI Block Read

A Block Read operation is initiated the same as a single-byte Read; however, the ZDI master continues to clock in the next byte from the ZDI slave as the ZDI slave continues to output data. The ZDI register address counter increments with each Read. If the ZDI register address reaches the end of the Read-Only ZDI register address space (`20h`), the address stops incrementing. Figure 43 illustrates the ZDI's Block Read timing.

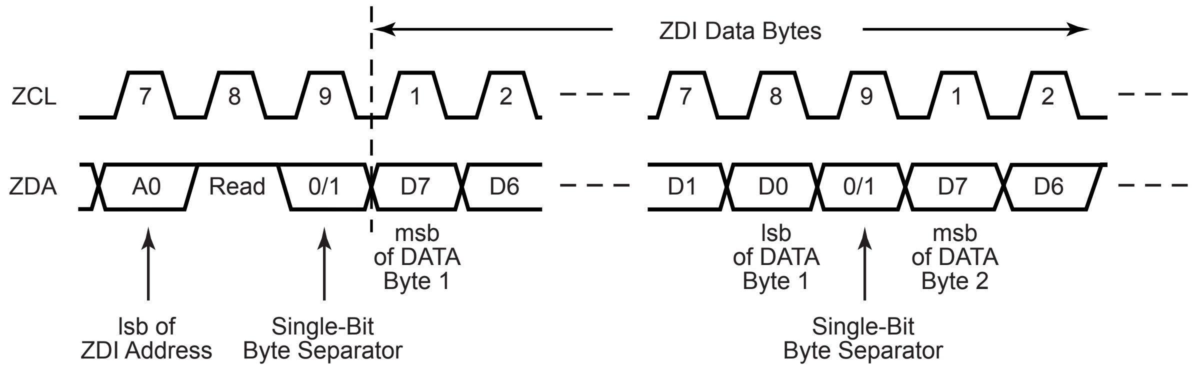

Figure 43. ZDI Block Data Read Timing

Operation of the eZ80L92 During ZDI Break Points

If the ZDI forces the CPU to break, only the CPU suspends operation. The system clock continues to operate and drive other peripherals. Those peripherals that can operate autonomously from the CPU may continue to operate, if so enabled. For example, the Watchdog Timer and Programmable Reload Timers continue to count during a ZDI break point.

When using the ZDI interface, any Write or Read operations of peripheral registers in the I/O address space produces the same effect as Read or Write operations using the CPU. Because many register Read/Write operations exhibit secondary effects, such as clearing flags or causing operations to commence, the effects of the Read/Write operations during a ZDI Break must be taken into consideration.

Bus Requests During ZDI Debug Mode

The ZDI block on the eZ80L92 allows an external device to take control of the address and data bus while the eZ80L92 is in DEBUG mode. ZDI_BUSACK_EN causes ZDI to allow or prevent acknowledgement of bus requests by external peripherals. The bus acknowledge only occurs at the end of the current ZDI operation (indicated by a High during the single-bit byte separator). The default reset condition is for bus acknowledgement to be disabled. To allow bus acknowledgement, the ZDI_BUSACK_EN must be written.

When an external bus request ($\overline{\text{BUSREQ}}$ pin asserted) is detected, ZDI waits until completion of the current operation before responding. ZDI acknowledges the bus request by asserting the bus acknowledge ($\overline{\text{BUSACK}}$) signal. If the ZDI block is not currently shifting data, it acknowledges the bus request immediately. ZDI uses the single-bit byte separator of each data word to determine if it is at the end of a ZDI operation. If the bit is a logical 0, ZDI does not assert $\overline{\text{BUSACK}}$ to allow additional data Read or Write operations. If the bit is a logical 1, indicating completion of the ZDI commands, $\overline{\text{BUSACK}}$ is asserted.

Potential Hazards of Enabling Bus Requests During Debug Mode

There are some potential hazards that the user must be aware of when enabling external bus requests during ZDI Debug mode. First, when the address and data bus are being used by an external source, ZDI must only access ZDI registers and internal CPU registers to prevent possible Bus contention. The bus acknowledge status is reported in the ZDI_BUS_STAT register. The $\overline{\text{BUSACK}}$ output pin also indicates the bus acknowledge state.

A second hazard is that when a bus acknowledge is granted, the ZDI is subject to any WAIT states that are assigned to the device currently being accessed by the external peripheral. To prevent data errors, ZDI should avoid data transmission while another device is controlling the bus.

Finally, exiting ZDI Debug mode while an external peripheral controls the address and data buses, as indicated by $\overline{\text{BUSACK}}$ assertion, may produce unpredictable results.

Table 122. External Write Timing

		20 MHz (ns)		50 MHz (ns)	
Parameter	**Description**	**Min**	**Max**	**Min**	**Max**
T_1	Clock Rise to ADDR Valid Delay	—	10.2	—	10.2
T_2	Clock Rise to ADDR Hold Time	2.4	—	2.4	—
T_3	Clock Fall to Output DATA Valid Delay	—	6	—	6
T_4	DATA Hold Time from Clock Rise	2.4	—	2.4	—
T_5	Clock Rise to $\overline{\text{CSx}}$ Assertion Delay	3.2	10.3	3.2	10.3
T_6	Clock Rise to $\overline{\text{CSx}}$ Deassertion Delay	2.9	9.7	2.9	9.7
T_7	Clock Rise to $\overline{\text{MREQ}}$ Assertion Delay	2.8	9.6	2.8	9.6
T_8	Clock Rise to $\overline{\text{MREQ}}$ Deassertion Delay	2.6	6.9	2.6	6.9
T_9	Clock Fall to $\overline{\text{WR}}$ Assertion Delay	1.5	5.0	1.5	5.0
T_{10}	Clock Rise to $\overline{\text{WR}}$ Deassertion Delay*	1.4	3.6	1.4	3.6

Note: *At the conclusion of a Write cycle, de-assertion of $\overline{\text{WR}}$ always occurs before any change to ADDR, DATA, $\overline{\text{CSx}}$, or $\overline{\text{MREQ}}$. In certain applications, the de-assertion of $\overline{\text{WR}}$ can be concurrent with ADDR, DATA, $\overline{\text{CSx}}$, or $\overline{\text{MREQ}}$ when buffering is used off-chip.

Wait State Timing for Write Operations

Figure 53 illustrates the extension of the memory access signals using a single WAIT state for a Write operation. This WAIT state is generated by setting CS_WAIT to 001 in the Chip Select Control Register.

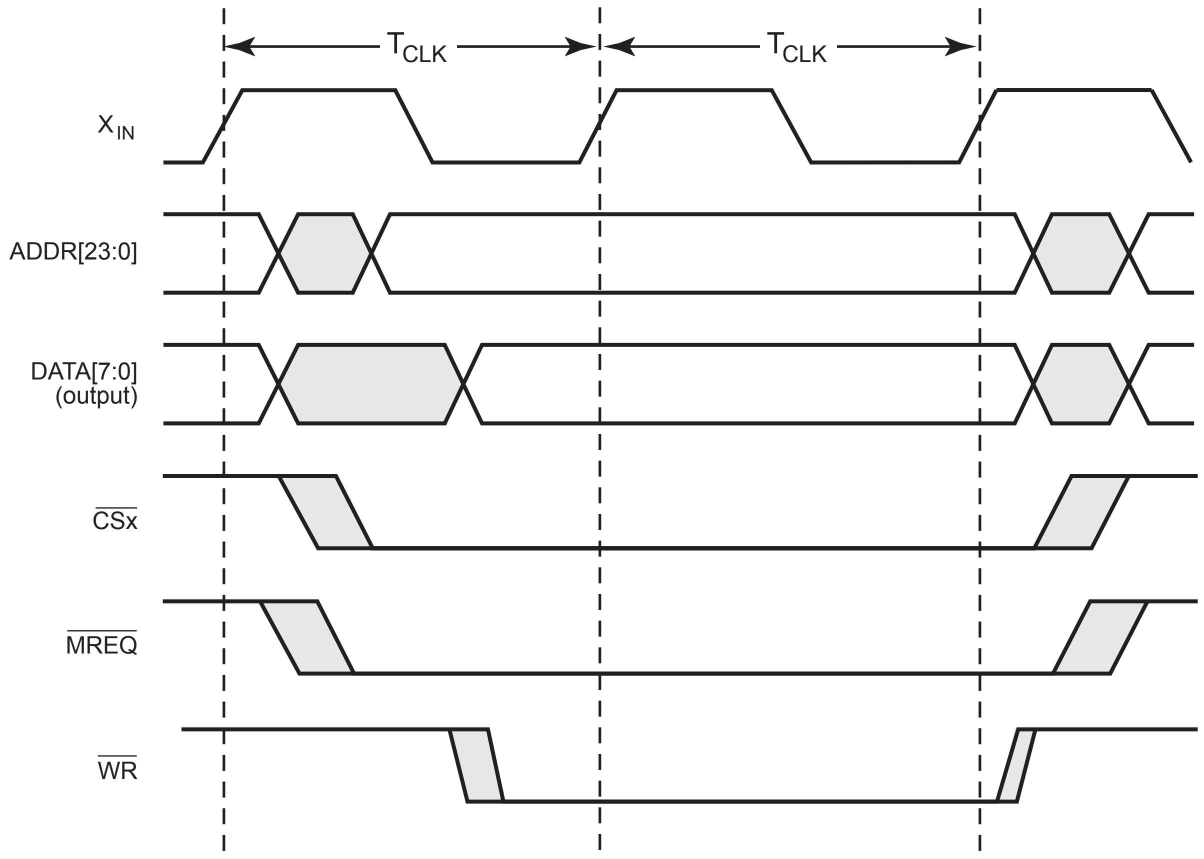

Figure 53. Wait State Timing for Write Operations