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	50MHz
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	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/ez80l92az050sg

Table 1. 100-Pin LQFP Pin Identification of eZ80L92 MCU (Continued)

Pin No	Symbol	Function	Signal Direction	Description
94	PB6	GPIO Port B	Bidirectional	This pin can be used for GPIO. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port B pin, when programmed as output, can be selected to be an open-drain or open-source output.
	MISO	Master In Slave Out	Bidirectional	The MISO line is configured as an input when the eZ80L92 MCU is an SPI master device and as an output when eZ80L92 MCU is an SPI slave device. This signal is multiplexed with PB6.
95	PB7	GPIO Port B	Bidirectional	This pin can be used for GPIO. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port B pin, when programmed as output, can be selected to be an open-drain or open-source output.
	MOSI	Master Out Slave In	Bidirectional	The MOSI line is configured as an output when the eZ80L92 MCU is an SPI master device and as an input when the eZ80L92 MCU is an SPI slave device. This signal is multiplexed with PB7.
96	V_{DD}	Power Supply		Power Supply.
97	V_{SS}	Ground		Ground.
98	SDA	I^2C Serial Data	Bidirectional	This pin carries the I^2C data signal.
99	SCL	I^2C Serial Clock	Bidirectional	This pin is used to receive and transmit the I^2C clock.
100	PHI	System Clock	Output	This pin is an output driven by the internal system clock.

Table 16. Intel® Bus Mode Read States (Separate Address and Data Buses) (Continued)

STATE T3	During State T3, no bus signals are altered. If the external ReadY ($\overline{\text{WAIT}}$) pin is driven Low at least one eZ80 system clock cycle prior to the beginning of State T3, additional WAIT states (T_{WAIT}) are asserted until the ReadY pin is driven High.
STATE T4	The CPU latches the Read data at the beginning of State T4. The CPU deasserts the $\overline{\text{RD}}$ signal and completes the Intel Bus Mode cycle.

During Write operations with separate address and data buses, the Intel Bus Mode employs 4 states (T1, T2, T3, and T4) as described in Table 17.

Table 17. Intel® Bus Mode Write States (Separate Address and Data Buses)

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, and the data is driven onto the data bus. The CPU drives the ALE signal High at the beginning of T1. During the middle of T1, the CPU drives ALE Low to facilitate the latching of the address.
STATE T2	During State T2, the CPU asserts the $\overline{\text{WR}}$ signal. Depending on the instruction, either the $\overline{\text{MREQ}}$ or $\overline{\text{IORQ}}$ signal is asserted.
STATE T3	During State T3, no bus signals are altered. If the external ReadY ($\overline{\text{WAIT}}$) pin is driven Low at least one eZ80 system clock cycle prior to the beginning of State T3, additional WAIT states (T_{WAIT}) are asserted until the ReadY pin is driven High.
STATE T4	The CPU deasserts the $\overline{\text{WR}}$ signal at the beginning of State T4. The CPU holds the data and address buses through the end of T4. The bus cycle is completed at the end of T4.

Intel Bus Mode timing is illustrated for a Read operation in Figure 10 and for a Write operation in Figure 11. If the ReadY signal (external $\overline{\text{WAIT}}$ pin) is driven Low prior to the beginning of State T3, additional WAIT states (T_{WAIT}) are asserted until the ReadY signal is driven High. The Intel Bus Mode states can be configured for 2 to 15 eZ80 system clock cycles. In the figures, each Intel® Bus Mode state is 2 eZ80 system clock cycles in duration. Figure 10 and Figure 11 also illustrate the assertion of one WAIT state (T_{WAIT}) by the selected peripheral.

Bit Position	Value	Description
[1:0] WDT_PERIOD	00	WDT time-out period is 2^{27} clock cycles.
	01	WDT time-out period is 2^{25} clock cycles.
	10	WDT time-out period is 2^{22} clock cycles.
	11	WDT time-out period is 2^{18} clock cycles.

Note: *RST_FLAG is only cleared by a non-WDT RESET.

Watchdog Timer Reset Register

The Watchdog Timer Reset register, described in Table 28, is an 8-bit Write-Only register. The Watchdog Timer is reset when an `A5h` value followed by `5Ah` is written to this register. Any amount of time can occur between the writing of the `A5h` value and the `5Ah` value, so long as the WDT time-out does not occur prior to completion.

Table 28. Watchdog Timer Reset Register (WDT_RR = 0094h)

Bit	7	6	5	4	3	2	1	0
Reset	X	X	X	X	X	X	X	X
CPU Access	W	W	W	W	W	W	W	W

Note: X = Undefined; W = Write only.

Bit Position	Value	Description
[7:0] WDT_RR	A5h	The first Write value required to reset the WDT prior to a time-out.
	5Ah	The second Write value required to reset the WDT prior to a time-out. If an A5h, 5Ah sequence is written to WDT_RR, the WDT timer is reset to its initial count value, and counting resumes.

Programmable Reload Timers

The eZ80L92 MCU features six Programmable Reload Timers (PRT). Each PRT contains a 16-bit downcounter and a 16-bit reload register. In addition, each PRT features a clock prescaler with four selectable taps for CLK ÷ 4, CLK ÷ 16, CLK ÷ 64, and CLK ÷ 256. Each timer can be individually enabled to operate in either SINGLE PASS or CONTINUOUS mode. The timer can be programmed to start, stop, restart from the current value, or restart from the initial value, and generate interrupts to the CPU.

Four of the Programmable Reload Timers (timers 0–3) feature a selectable clock source input. The input for these timers can be either the system clock or the Real-Time Clock (RTC) source. Timers 0–3 can also be used for event counting, with their inputs received from a GPIO port pin. Output from timers 4 and 5 can be directed to a GPIO port pin.

Each of the six PRTs available on the eZ80L92 can be controlled individually. They do not share the same counters, reload registers, control registers, or interrupt signals.
A simplified block diagram of a programmable reload timer is illustrated in Figure 18.

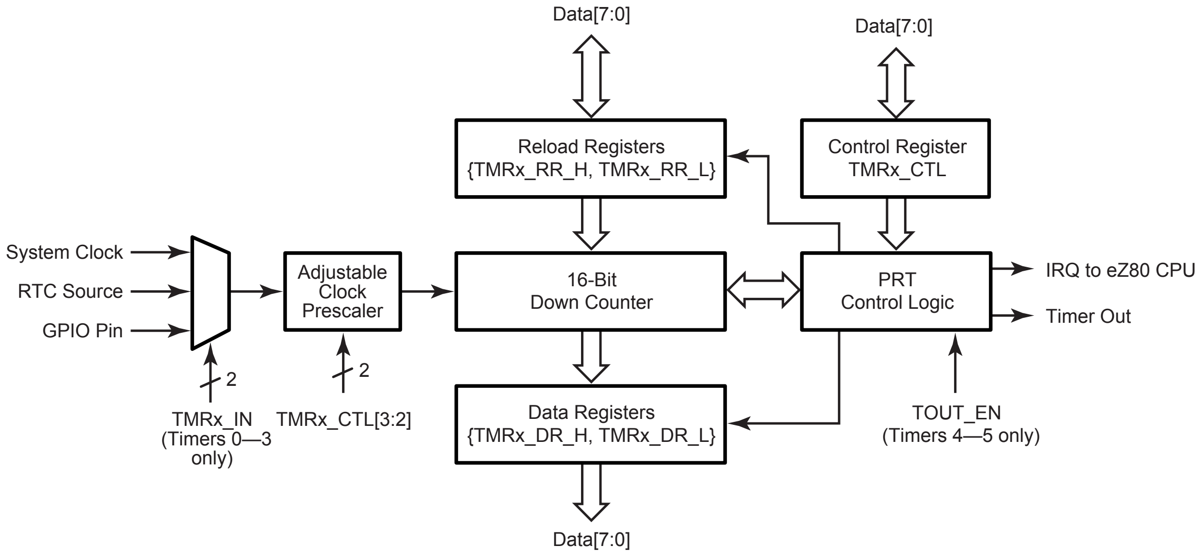

Figure 18. Programmable Reload Timer Block Diagram

Programmable Reload Timer Operation

Setting Timer Duration

There are three factors to consider when determining Programmable Reload Timer duration—clock frequency, clock divider ratio, and initial count value. Minimum duration

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.

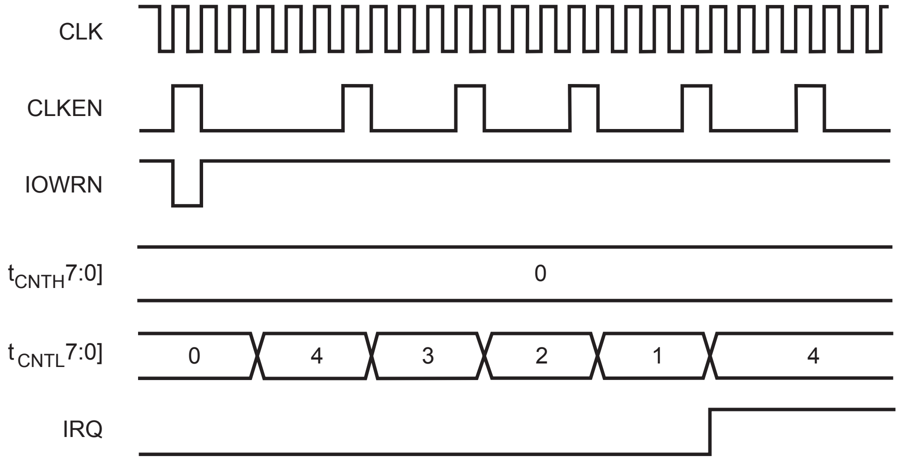

Figure 20. PRT CONTINUOUS Mode Operation Example

The third source of a receiver interrupt is a line status error, indicating an error in byte reception. This error may result from:

- Incorrect received parity.
- Incorrect framing; that is, the stop bit is not detected by receiver at the end of the byte.
- Receiver over run condition.
- A BREAK condition being detected on the receive data input.

An interrupt due to one of the above conditions is cleared when the UARTx_LSR register is read. In FIFO mode, a line status interrupt is generated only after the received byte with an error reaches the top of the FIFO and is ready to be read.

A line status interrupt is activated (provided this interrupt is enabled) as long as the Read pointer of the receiver FIFO points to the location of the FIFO that contains a byte with the error. The interrupt is immediately cleared when the UARTx_LSR register is read. The ERR bit of the UARTx_LSR register is active as long as an erroneous byte is present in the receiver FIFO.

UART Modem Status Interrupt

The modem status interrupt is generated if there is any change in state of the modem status inputs to the UART. This interrupt is cleared when the processor reads the UARTx_MSR register.

UART Recommended Usage

The following is the standard sequence of events that occur in the eZ80L92 MCU using the UART. A description of each follows.

1. Module reset.
2. Control transfers to configure UART operation.
3. Data transfers.

Module Reset. Upon reset, all internal registers are set to their default values. All command status registers are programmed with their default values, and the FIFOs are flushed.

Control Transfers. Based on the requirements of the application, the data transfer baud rate is determined and the BRG is configured to generate a 16X clock frequency. Interrupts are disabled and the communication control parameters are programmed in the UARTx_LCTL register. The FIFO configuration is determined and the receive trigger levels are set in the UARTx_FCTL register. The status registers, UARTx_LSR and UARTx_MSR, are read, and ensure that none of the interrupt sources are active. The inter-

Note: The UARTx_BRG_L registers share the same address space with the UARTx_RBR and UARTx_THR registers. The UARTx_BRG_H registers share the same address space with the UARTx_IER registers. Bit 7 of the associated UART Line Control register (UARTx_LCTL) must be set to 1 to enable access to the BRG registers.

Table 52. UART Baud Rate Generator Registers—Low Byte (UART0_BRG_L = 00C0h, UART1_BRG_L = 00D0h)

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

Bit Position	Value	Description
[7:0] UARTx_BRG_L	00h–FFh	These bits represent the Low byte of the 16-bit Baud Rate Generator divider value. The complete BRG divisor value is returned by {UARTx_BRG_H, UARTx_BRG_L}.

Table 53. UART Baud Rate Generator Registers—High Byte (UART0_BRG_H = 00C1h, UART1_BRG_H = 00D1h)

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 = Read only; R/W = Read/Write.								

Bit Position	Value	Description
[7:0] UARTx_BRG_H	00h–FFh	These bits represent the High byte of the 16-bit Baud Rate Generator divider value. The complete BRG divisor value is returned by {UARTx_BRG_H, UARTx_BRG_L}.

UART Registers

After a RESET, all UART registers are set to their default values. Any Writes to unused registers or register bits are ignored and Reads return a value of 0. For compatibility with future revisions, unused bits within a register must be written with a value of 0. Read/

Table 55. UART Receive Buffer Registers (UART0_RBR = 00C0h, UART1_RBR = 00D0h)

Bit	7	6	5	4	3	2	1	0
Reset	X	X	X	X	X	X	X	X
CPU Access	R	R	R	R	R	R	R	R
Note: R = Read only.								

Bit Position	Value	Description
[7:0] RxD	00h–FFh	Receive data byte.

UART Interrupt Enable Registers

The UARTx_IER register is used to enable and disable the UART interrupts. The UARTx_IER registers share the same I/O addresses as the UARTx_BRG_H registers. See Table 56.

Table 56. UART Interrupt Enable Registers (UART0_IER = 00C1h, UART1_IER = 00D1h)

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: R = Read only.; R/W = Read/Write.								

Bit Position	Value	Description
[7:4]	0000	Reserved
3 MIIE	0	Modem interrupt on edge detect of status inputs is disabled.
	1	Modem interrupt on edge detect of status inputs is enabled.
2 LSIE	0	Line status interrupt is disabled.
	1	Line status interrupt is enabled for receive data errors: incorrect parity bit received, framing error, overrun error, or break detection.

6. When the SPI data transfer is complete, de-assert the $\overline{\text{ENABLE}}$ pin of the slave device.

Data Transfer Procedure with SPI Configured as a Slave

Follow the steps below for data transfer with SPI configured as the slave:

1. Load the SPI Baud Rate Generator Registers, SPI_BRG_H and SPI_BRG_L.
2. Load the SPI Transmit Shift Register, SPI_TSR. This load cannot occur while the SPI slave is currently receiving data.
3. Wait for the external SPI Master device to initiate the data transfer by asserting $\overline{\text{SS}}$.

SPI Registers

There are six registers in the Serial Peripheral Interface which provide control, status, and data storage functions. The SPI registers are described in the following section.

SPI Baud Rate Generator Registers—Low Byte and High Byte

These registers hold the Low and High bytes of the 16-bit divisor count loaded by the processor for baud rate generation. The 16-bit clock divisor value is returned by {SPI_BRG_H, SPI_BRG_L}. Upon RESET, the 16-bit BRG divisor value resets to `0002h`. When configured as a Master, the 16-bit divisor value must be between `0003h` and `FFFFh`, inclusive. When configured as a Slave, the 16-bit divisor value must be between `0004h` and `FFFFh`, inclusive.

A Write to either the Low or High byte registers for the BRG Divisor Latch causes both bytes to be loaded into the BRG counter and the count restarted. See Table 69 and Table 70.

Table 69. SPI Baud Rate Generator Register—Low Byte (SPI_BRG_L = 00B8h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	1	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**
[7:0] SPI_BRG_L	00h–FFh	These bits represent the Low byte of the 16-bit Baud Rate Generator divider value. The complete BRG divisor value is returned by {SPI_BRG_H, SPI_BRG_L}.

is transmitted, the IFLG is set and the I2C_SR register contains C8h and the I^2C returns to the idle state. The AAK bit must be set to 1 before reentering SLAVE mode.

If no acknowledge is received after transmitting a byte, the IFLG is set and the I2C_SR register contains C0h. The I^2C then returns to the idle state.

If a STOP condition is detected after an acknowledge bit, the I^2C returns to the idle state.

Slave Receive

In SLAVE RECEIVE mode, a number of data bytes are received from a master transmitter.

The I^2C enters SLAVE RECEIVE mode when it receives its own slave address and a Write bit (lsb = 0) after a START condition. The I^2C transmits an acknowledge bit and sets the IFLG bit in the I2C_CTL register and the I2C_SR register contains the status code 60h. The I^2C also enters SLAVE RECEIVE mode when it receives the general call address 00h (if the GCE bit in the I2C_SAR register is set). The status code is then 70h.

Note: When the I^2C contains a 10-bit slave address (signified by F0h–F7h in the I2C_SAR register), it transmits an acknowledge after the first address byte is received but no interrupt is generated. IFLG is not set and the status does not change. The I^2C generates an interrupt only after the second address byte is received. The I^2C sets the IFLG bit and loads the status code as described above.

I^2C goes from MASTER mode to SLAVE RECEIVE mode when arbitration is lost during the transmission of an address, and the slave address and Write bit (or the general call address if the CGE bit in the I2C_SAR register is set to 1) are received. The status code in the I2C_SR register is 68h if the slave address is received or 78h if the general call address is received. The IFLG bit must be cleared to 0 to allow data transfer to continue.

If the AAK bit in the I2C_CTL register is set to 1 then an acknowledge bit (Low level on SDA) is transmitted and the IFLG bit is set after each byte is received. The I2C_SR register contains the status code 80h or 90h if SLAVE RECEIVE mode is entered with the general call address. The received data byte can be read from the I2C_DR register and the IFLG bit must be cleared to allow the transfer to continue. If a STOP condition or a repeated START condition is detected after the acknowledge bit, the IFLG bit is set and the I2C_SR register contains status code A0h.

If the AAK bit is cleared to 0 during a transfer, the I^2C transmits a not-acknowledge bit (High level on SDA) after the next byte is received, and set the IFLG bit. The I2C_SR register contains the status code 88h or 98h if SLAVE RECEIVE mode is entered with the general call address. The I^2C returns to the idle state when the IFLG bit is cleared to 0.

Table 86. I²C Status Codes (Continued)

Code	Status
B0h	Arbitration lost in address as master, slave address and Read bit received, ACK transmitted
B8h	Data byte transmitted in SLAVE mode, ACK received
C0h	Data byte transmitted in SLAVE mode, ACK not received
C8h	Last byte transmitted in SLAVE mode, ACK received
D0h	Second Address byte and Write bit transmitted, ACK received
D8h	Second Address byte and Write bit transmitted, ACK not received
F8h	No relevant status information, IFLG = 0

If an illegal condition occurs on the I²C bus, the bus error state is entered (status code 00h). To recover from this state, the STP bit in the I2C_CTL register must be set and the IFLG bit cleared. The I²C then returns to the idle state. No STOP condition is transmitted on the I²C bus.

Note: The STP and STA bits may be set to 1 at the same time to recover from the bus error. The I²C then sends a START.

I²C Clock Control Register

The I2C_CCR register is a Write-Only register. The seven LSBs control the frequency at which the I²C bus is sampled and the frequency of the I²C clock line (SCL) when the I²C is in MASTER mode. The Write-Only I2C_CCR registers share the same I/O addresses as the Read-Only I2C_SR registers. See Table 87.

Table 87. I²C Clock Control Registers (I2C_CCR = 00CCh)

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 = Read only.								

Bit Position	Value	Description
7	0	Reserved.

Bit Position	Value	Description
7 BRK_NEXT	0	The ZDI BREAK on the next CPU instruction is disabled. Clearing this bit releases the CPU from its current BREAK condition.
	1	The ZDI BREAK on the next CPU instruction is enabled. The CPU can use multibyte Op Codes and multibyte operands. break points only occur on the first Op Code in a multibyte Op Code instruction. If the ZCL pin is High and the ZDA pin is Low at the end of RESET, this bit is set to 1 and a BREAK occurs on the first instruction following the RESET. This bit is set automatically during ZDI BREAK on address match. A BREAK can also be forced by writing a 1 to this bit.
6 brk_addr3	0	The ZDI BREAK, upon matching break address 3, is disabled.
	1	The ZDI BREAK, upon matching break address 3, is enabled.
5 brk_addr2	0	The ZDI BREAK, upon matching break address 2, is disabled.
	1	The ZDI BREAK, upon matching break address 2, is enabled.
4 brk_addr1	0	The ZDI BREAK, upon matching break address 1, is disabled.
	1	The ZDI BREAK, upon matching break address 1, is enabled.
3 brk_addr0	0	The ZDI BREAK, upon matching break address 0, is disabled.
	1	The ZDI BREAK, upon matching break address 0, is enabled.
2 ign_low_1	0	The *Ignore the Low Byte* function of the ZDI Address Match 1 registers is disabled. If BRK_ADDR1 is set to 1, ZDI initiates a BREAK when the entire 24-bit address, ADDR[23:0], matches the 3-byte value {ZDI_ADDR1_U, ZDI_ADDR1_H, ZDI_ADDR1_L}.
	1	The *Ignore the Low Byte* function of the ZDI Address Match 1 registers is enabled. If BRK_ADDR1 is set to 1, ZDI initiates a BREAK when only the upper 2 bytes of the 24-bit address, ADDR[23:8], match the 2-byte value {ZDI_ADDR1_U, ZDI_ADDR1_H}. As a result, a BREAK can occur anywhere within a 256-byte page.

ZDI Write Data Registers

These three registers are used in the ZDI Write-Only register address space to store the data that is written when a Write instruction is sent to the ZDI Read/Write Control register (ZDI_RW_CTL). The ZDI Read/Write Control register is located at ZDI address `16h` immediately following the ZDI Write Data registers. As a result, the ZDI Master is allowed to write the data to {ZDI_WR_U, ZDI_WR_H, ZDI_WR_L} and the Write command in one data transfer operation. See Table 95.

Table 95. ZDI Write Data Registers (ZDI_WR_U = 13h, ZDI_WR_H = 14h, and ZDI_WR_L = 15h in the ZDI Register Write-Only Address Space)

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

Bit Position	Value	Description
[7:0] ZDI_WR_L, ZDI_WR_H, or ZDI_WR_L	00h–FFh	These registers contain the data that is written during execution of a Write operation defined by the ZDI_RW_CTL register. The 24-bit data value is stored as {ZDI_WR_U, ZDI_WR_H, ZDI_WR_L}. If less than 24 bits of data are required to complete the required operation, the data is taken from the least significant byte(s).

ZDI Read/Write Control Register

The ZDI Read/Write Control register is used in the ZDI Write-Only Register address to read data from, write data to, and manipulate the CPU's registers or memory locations. When this register is written, the eZ80L92 immediately performs the operation corresponding to the data value written as described in Table 96.

When a Read operation is executed via this register, the requested data values are placed in the ZDI Read Data registers {ZDI_RD_U, ZDI_RD_H, ZDI_RD_L}. When a Write operation is executed via this register, the Write data is taken from the ZDI Write Data registers {ZDI_WR_U, ZDI_WR_H, ZDI_WR_L}. See Table 96. For more information on the CPU registers, refer to the *eZ80® CPU User Manual (UM0077)*.

Bit Position	Value	Description
[7:0] zdi_wr_mem	00h–FFh	The 8-bit data that is transferred to the ZDI slave following a Write to this address is written to the address indicated by the current program counter. The program counter is incremented following each 8 bits of data. In Z80 MEMORY mode, ({MBASE, PC[15:0]}) ← 8 bits of transferred data. In ADL MEMORY mode, (PC[23:0]) ← 8 bits of transferred data.

eZ80® Product ID Low and High Byte Registers

The eZ80 Product ID Low and High Byte registers combine to provide a means for an external device to determine the particular eZ80 product being addressed. For the eZ80L92, these two bytes, {ZDI_ID_H, ZDI_ID_L} return the value {00h, 06h}. See Tables 100 and 101.

Table 100. eZ80® Product ID Low Byte Register (ZDI_ID_L = 00h in the ZDI Register Read-Only Address Space)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	1	1	0
CPU Access	R	R	R	R	R	R	R	R
Note: R = Read-only.								

Bit Position	Value	Description
[7:0] zdi_id_l	06h	{ZDI_ID_H, ZDI_ID_L} = {00h, 06h} indicates the eZ80L92 product.

Table 101. eZ80® Product ID High Byte Register (ZDI_ID_H = 01h in the ZDI Register Read-Only Address Space)

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	R	R	R
Note: R = Read-only.								

Table 108. Input/Output Instructions

Mnemonic	Instruction
OUT0	Output to I/O on Page 0
OUTD (OTDR)	Output to I/O and Decrement (with Repeat)
OUTD2 (OTD2R)	Output to I/O and Decrement (with Repeat)
OUTI (OTIR)	Output to I/O and Increment (with Repeat)
OUTI2 (OTI2R)	Output to I/O and Increment (with Repeat)
TSTIO	Test I/O

Table 108. Load Instructions

Mnemonic	Instruction
LD	Load
LEA	Load Effective Address
PEA	Push Effective Address
POP	Pop
PUSH	Push

Table 108. Logical Instructions

Mnemonic	Instruction
AND	Logical AND
CPL	Complement Accumulator
OR	Logical OR
TST	Test Accumulator
XOR	Logical Exclusive OR

Table 108. Processor Control Instructions

Mnemonic	Instruction
CCF	Complement Carry Flag
DI	Disable Interrupts

External Memory Read Timing

Figure 48 and Table 121 illustrate the timing for external Reads.

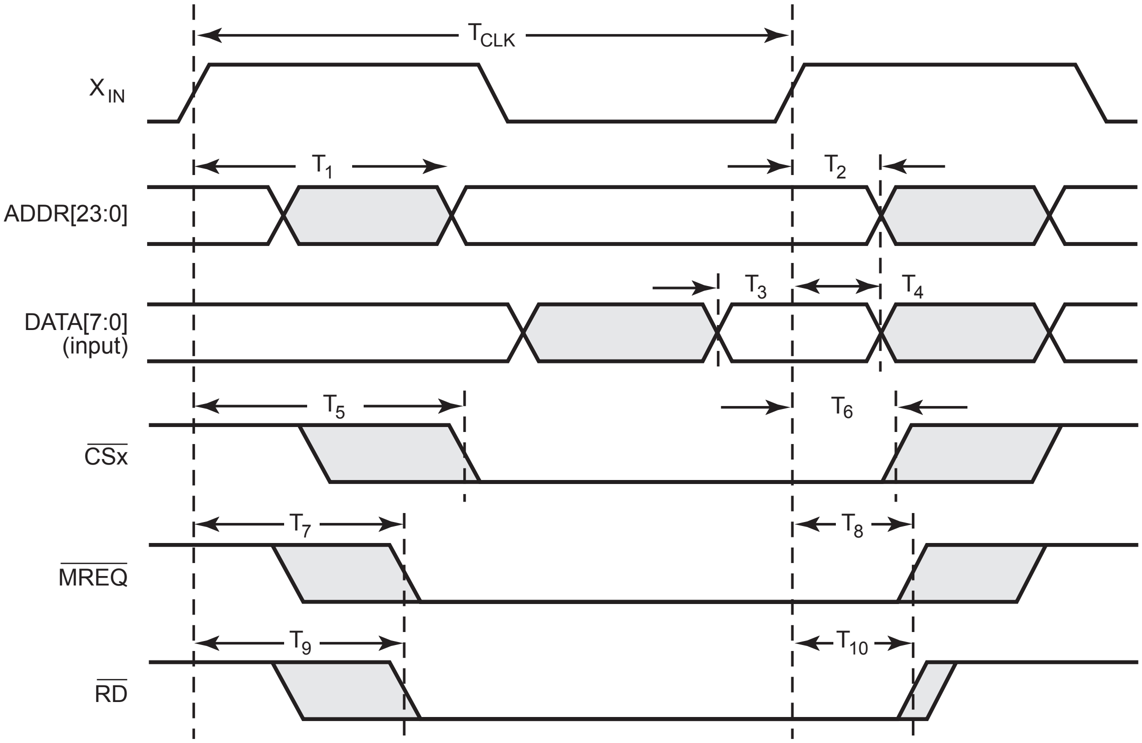

Figure 48. External Memory Read Timing

Table 121. External Read 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	Input DATA Valid to Clock Rise Setup Time	1.0	—	1.0	—
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

Index