eZ80L92

Product Specification

PRELIMINARY

PS013011-0204

ZiLOG Worldwide Headquarters • 532 Race Street • San Jose, CA 95126
Telephone: 408.558.8500 • Fax: 408.558.8300 • www.ZiLOG.com

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

ZiLOG Worldwide Headquarters
532 Race Street
San Jose, CA 95126
Telephone: 408.558.8500
Fax: 408.558.8300
www.ZiLOG.com

Document Disclaimer

Table of Contents

List of Figures

List of Tables

Architectural Overview

The eZ80L92 microprocessor is a high-speed single-cycle instruction-fetch microprocessor with a maximum clock speed of 50 MHz. The eZ80L92 is a member of ZiLOG's new eZ80® product family. It can operate in Z80-compatible addressing mode (64 KB) or full 24-bit addressing mode (16 MB). The rich peripheral set of the eZ80L92 makes it suitable for a variety of applications including industrial control, embedded communication, and point-of-sale terminals.

Features

- 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 UARTs with independent baud rate generators
- SPI with independent clock rate generator
- 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-and Motorola-style buses
- Fixed-priority vectored interrupts (both internal and external) and interrupt controller
- Real-time clock with on-chip 32 KHz oscillator, selectable 50/60Hz input, and separate V_{DD} pin for battery backup
- Six 16-bit Counter/Timers with prescalers and direct input/output drive
- Watch-Dog Timer
- 24 bits of General-Purpose I/O
- JTAG and ZDI debug interfaces
- 100-pin LQFP package
- 3.0–3.6 V supply voltage with 5 V tolerant inputs

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

- Operating Temperature Range
  - Standard: 0ºC to +70ºC
  - Extended: –40ºC to +105ºC

Note: All signals with an overline are active Low. For example, B/$\overline{W}$, for which WORD is active Low, and $\overline{B}$/W, for which BYTE is active Low.

Power connections follow these conventional descriptions:

Connection	Circuit	Device
Power	V_{CC}	V_{DD}
Ground	GND	V_{SS}

Block Diagram

Figure 1 illustrates a block diagram of the eZ80L92 microprocessor.

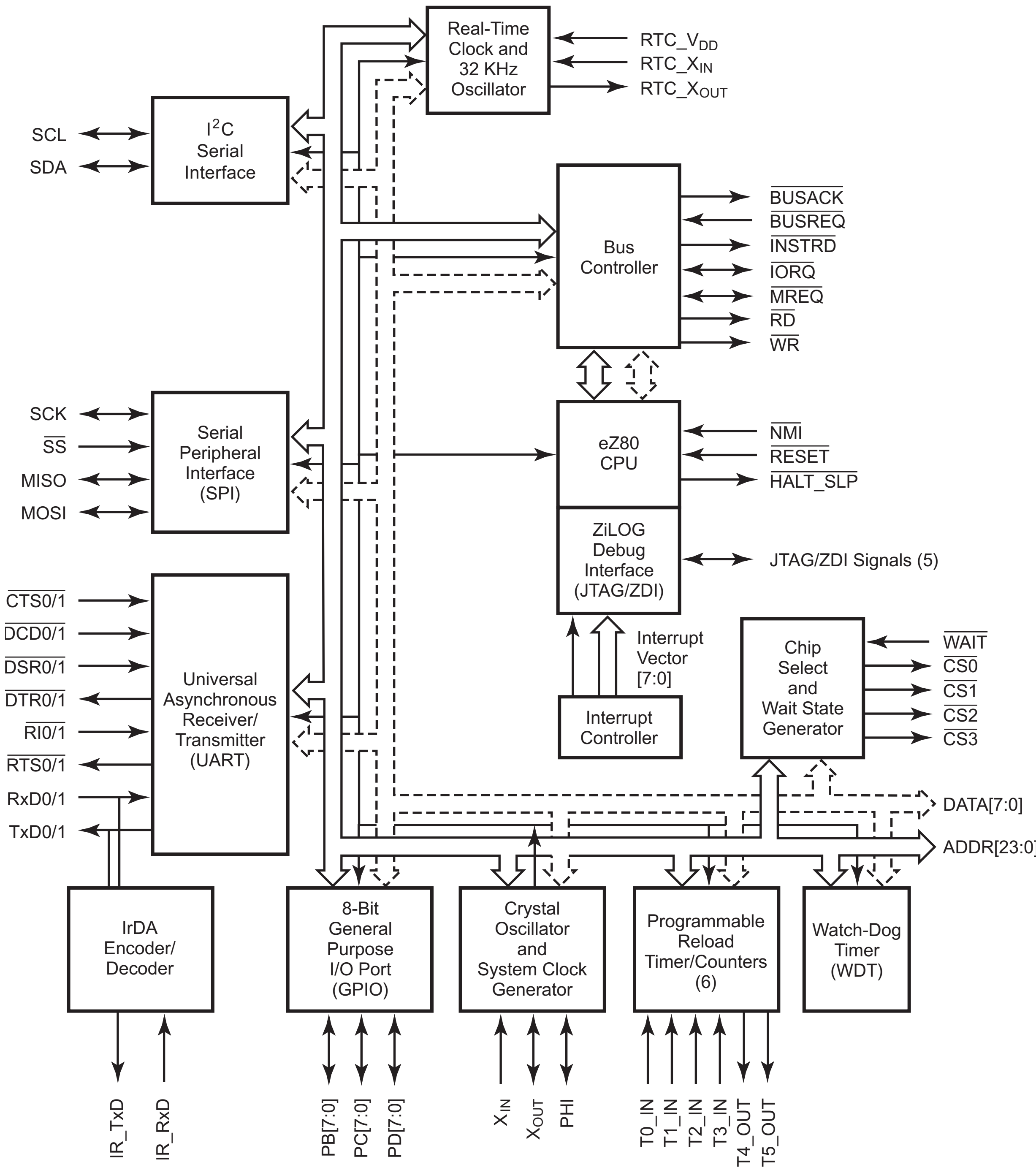


Figure 1. eZ80L92 **Block Diagram**

Pin Description

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

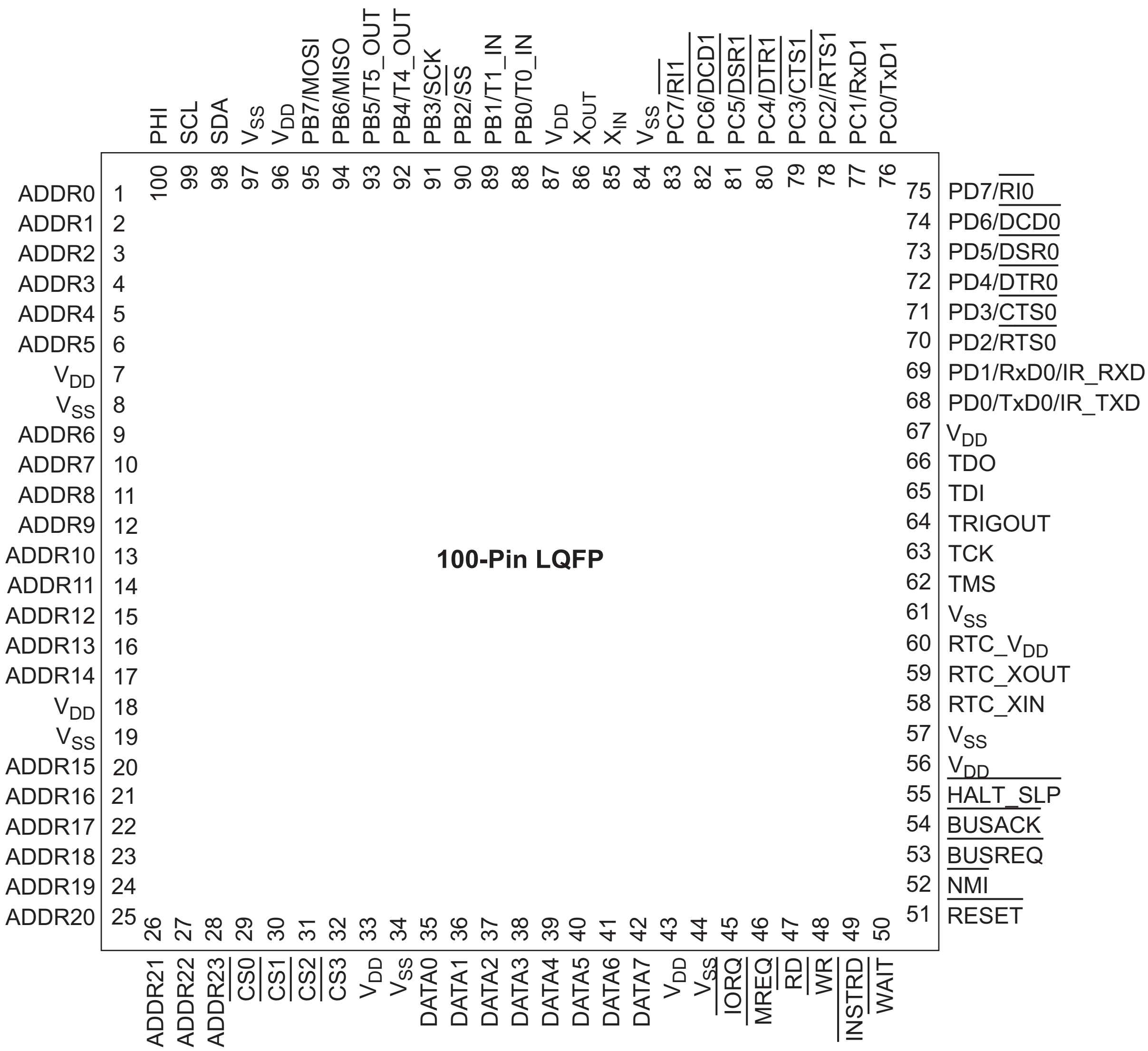


Figure 2. 100-Pin LQFP Configuration of the eZ80L92

Table 1. 100-Pin LQFP Pin Identification of the eZ80L92 Device

Pin #	Symbol	Function	Signal Direction	Description
1	ADDR0	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
2	ADDR1	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
3	ADDR2	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
4	ADDR3	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
5	ADDR4	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
6	ADDR5	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.

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

Pin #	Symbol	Function	Signal Direction	Description
7	V_{DD}	Power Supply		Power Supply.
8	V_{SS}	Ground		Ground.
9	ADDR6	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
10	ADDR7	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
11	ADDR8	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
12	ADDR9	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
13	ADDR10	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.

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

Pin #	Symbol	Function	Signal Direction	Description
14	ADDR11	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
15	ADDR12	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
16	ADDR13	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
17	ADDR14	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
18	V_{DD}	Power Supply		Power Supply.
19	V_{SS}	Ground		Ground.
20	ADDR15	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.

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

Pin #	Symbol	Function	Signal Direction	Description
21	ADDR16	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
22	ADDR17	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
23	ADDR18	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
24	ADDR19	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
25	ADDR20	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
26	ADDR21	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.

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

Pin #	Symbol	Function	Signal Direction	Description
27	ADDR22	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
28	ADDR23	Address Bus	Bidirectional	Configured as an output in normal operation. The address bus selects a location in memory or I/O space to be read or written. Configured as an input during bus acknowledge cycles. Drives the Chip Select/Wait State Generator block to generate Chip Selects.
29	$\overline{\text{CS0}}$	Chip Select 0	Output, Active Low	$\overline{\text{CS0}}$ Low indicates that an access is occurring in the defined CS0 memory or I/O address space.
30	$\overline{\text{CS1}}$	Chip Select 1	Output, Active Low	$\overline{\text{CS1}}$ Low indicates that an access is occurring in the defined CS1 memory or I/O address space.
31	$\overline{\text{CS2}}$	Chip Select 2	Output, Active Low	$\overline{\text{CS2}}$ Low indicates that an access is occurring in the defined CS2 memory or I/O address space.
32	$\overline{\text{CS3}}$	Chip Select 3	Output, Active Low	$\overline{\text{CS3}}$ Low indicates that an access is occurring in the defined CS3 memory or I/O address space.
33	V_{DD}	Power Supply		Power Supply.
34	V_{SS}	Ground		Ground.
35	DATA0	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80L92 drives these lines only during write cycles when the eZ80L92 is the bus master.
36	DATA1	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80L92 drives these lines only during write cycles when the eZ80L92 is the bus master.

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

Pin #	Symbol	Function	Signal Direction	Description
37	DATA2	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80L92 drives these lines only during write cycles when the eZ80L92 is the bus master.
38	DATA3	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80L92 drives these lines only during write cycles when the eZ80L92 is the bus master.
39	DATA4	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80L92 drives these lines only during write cycles when the eZ80L92 is the bus master.
40	DATA5	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80L92 drives these lines only during write cycles when the eZ80L92 is the bus master.
41	DATA6	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80L92 drives these lines only during write cycles when the eZ80L92 is the bus master.
42	DATA7	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80L92 drives these lines only during write cycles when the eZ80L92 is the bus master.
43	V_{DD}	Power Supply		Power Supply.
44	V_{SS}	Ground		Ground.
45	$\overline{IORQ}$	Input/Output Request	Bidirectional, Active Low	$\overline{IORQ}$ indicates that the CPU is accessing a location in I/O space. RD and WR indicate the type of access. The eZ80L92 does not drive this line during RESET. It is an input in bus acknowledge cycles.
46	$\overline{MREQ}$	Memory Request	Bidirectional, Active Low	$\overline{MREQ}$ Low indicates that the CPU is accessing a location in memory. The RD, WR, and INSTRD signals indicate the type of access. The eZ80L92 does not drive this line during RESET. It is an input in bus acknowledge cycles.

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

Pin #	Symbol	Function	Signal Direction	Description
47	$\overline{\text{RD}}$	Read	Output, Active Low	$\overline{\text{RD}}$ Low indicates that the eZ80L92 is reading from the current address location. This pin is tristated during bus acknowledge cycles.
48	$\overline{\text{WR}}$	Write	Output, Active Low	$\overline{\text{WR}}$ indicates that the CPU is writing to the current address location. This pin is tristated during bus acknowledge cycles.
49	$\overline{\text{INSTRD}}$	Instruction Read Indicator	Output, Active Low	$\overline{\text{INSTRD}}$ (with $\overline{\text{MREQ}}$ and $\overline{\text{RD}}$) indicates the eZ80L92 is fetching an instruction from memory. This pin is tristated during bus acknowledge cycles.
50	$\overline{\text{WAIT}}$	WAIT Request	Input, Active Low	Driving the $\overline{\text{WAIT}}$ pin Low forces the CPU to wait additional clock cycles for an external peripheral or external memory to complete its Read or Write operation.
51	$\overline{\text{RESET}}$	Reset	Schmitt Trigger Input, Active Low	This signal is used to initialize the eZ80L92. This input must be Low for a minimum of 3 system clock cycles, and must be held Low until the clock is stable. This input includes a Schmitt trigger to allow RC rise times.
52	$\overline{\text{NMI}}$	Nonmaskable Interrupt	Schmitt Trigger Input, Active Low	The $\overline{\text{NMI}}$ input is a higher priority input than the maskable interrupts. It is always recognized at the end of an instruction, regardless of the state of the interrupt enable control bits. This input includes a Schmitt trigger to allow RC rise times.
53	$\overline{\text{BUSREQ}}$	Bus Request	Input, Active Low	External devices can request the eZ80L92 to release the memory interface bus for their use, by driving this pin Low.
54	$\overline{\text{BUSACK}}$	Bus Acknowledge	Output, Active Low	The eZ80L92 responds to a Low on $\overline{\text{BUSREQ}}$, by tristating the address, data, and control signals, and by driving the $\overline{\text{BUSACK}}$ line Low. During bus acknowledge cycles ADDR[23:0], $\overline{\text{IORQ}}$, and $\overline{\text{MREQ}}$ are inputs.
55	$\overline{\text{HALT\_SLP}}$	HALT and SLEEP Indicator	Output, Active Low	A Low on this pin indicates that the CPU has entered either HALT or SLEEP mode because of execution of either a HALT or SLP instruction.

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

Pin #	Symbol	Function	Signal Direction	Description
56	V_{DD}	Power Supply		Power Supply.
57	V_{SS}	Ground		Ground.
58	RTC_XIN	Real-Time Clock Crystal Input	Input	This pin is the input to the low-power 32KHz crystal oscillator for the Real-Time Clock.
59	RTC_XOUT	Real-Time Clock Crystal Output	Bidirectional	This pin is the output from the low-power 32KHz crystal oscillator for the Real-Time Clock. This pin is an input when the RTC is configured to operate from 50/60 Hz input clock signals and the 32 KHz crystal oscillator is disabled.
60	RTC_V_{DD}	Real-Time Clock Power Supply		Power supply for the Real-Time Clock and associated 32KHz oscillator. Isolated from the power supply to the remainder of the chip. A battery can be connected to this pin to supply constant power to the Real-Time Clock and 32KHz oscillator.
61	V_{SS}	Ground		Ground.
62	TMS	JTAG Test Mode Select	Input	JTAG Mode Select Input.
63	TCK	JTAG Test Clock	Input	JTAG and ZDI clock input.
64	TRIGOUT	JTAG Test Trigger Output	Output	Active High trigger event indicator.
65	TDI	JTAG Test Data In	Bidirectional	JTAG data input pin. Functions as ZDI data I/O pin when JTAG is disabled.
66	TDO	JTAG Test Data Out	Output	JTAG data output pin.
67	V_{DD}	Power Supply		Power Supply.

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

Pin #	Symbol	Function	Signal Direction	Description
68	PD0	GPIO Port D	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port D pin, when programmed as output, can be selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
	TxD0	UART Transmit Data	Output	This pin is used by the UART to transmit asynchronous serial data. This signal is multiplexed with PD0.
	IR_TXD	IrDA Transmit Data	Output	This pin is used by the IrDA encoder/decoder to transmit serial data. This signal is multiplexed with PD0.
69	PD1	GPIO Port D	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port D pin, when programmed as output, can be selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
	RxD0	Receive Data	Input	This pin is used by the UART to receive asynchronous serial data. This signal is multiplexed with PD1.
	IR_RXD	IrDA Receive Data	Input	This pin is used by the IrDA encoder/decoder to receive serial data. This signal is multiplexed with PD1.
70	PD2	GPIO Port D	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port D pin, when programmed as output, can be selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
	$\overline{\text{RTS0}}$	Request to Send	Output, Active Low	Modem control signal from UART. This signal is multiplexed with PD2.

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

Pin #	Symbol	Function	Signal Direction	Description
71	PD3	GPIO Port D	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port D pin, when programmed as output, can be selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
	$\overline{\text{CTS0}}$	Clear to Send	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PD3.
72	PD4	GPIO Port D	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port D pin, when programmed as output, can be selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
	$\overline{\text{DTR0}}$	Data Terminal Ready	Output, Active Low	Modem control signal to the UART. This signal is multiplexed with PD4.
73	PD5	GPIO Port D	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port D pin, when programmed as output, can be selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
	$\overline{\text{DSR0}}$	Data Set Ready	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PD5.
74	PD6	GPIO Port D	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port D pin, when programmed as output, can be selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
	$\overline{\text{DCD0}}$	Data Carrier Detect	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PD6.

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

Pin #	Symbol	Function	Signal Direction	Description
75	PD7	GPIO Port D	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port D pin, when programmed as output, can be selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
	$\overline{RI0}$	Ring Indicator	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PD7.
76	PC0	GPIO Port C	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port C pin, when programmed as output, can be selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
	TxD1	Transmit Data	Output	This pin is used by the UART to transmit asynchronous serial data. This signal is multiplexed with PC0.
77	PC1	GPIO Port C	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port C pin, when programmed as output, can be selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
	RxD1	Receive Data	Input	This pin is used by the UART to receive asynchronous serial data. This signal is multiplexed with PC1.

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

Pin #	Symbol	Function	Signal Direction	Description
78	PC2	GPIO Port C	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port C pin, when programmed as output, can be selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
	$\overline{\text{RTS1}}$	Request to Send	Output, Active Low	Modem control signal from UART. This signal is multiplexed with PC2.
79	PC3	GPIO Port C	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port C pin, when programmed as output, can be selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
	$\overline{\text{CTS1}}$	Clear to Send	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PC3.
80	PC4	GPIO Port C	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port C pin, when programmed as output, can be selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
	$\overline{\text{DTR1}}$	Data Terminal Ready	Output, Active Low	Modem control signal to the UART. This signal is multiplexed with PC4.
81	PC5	GPIO Port C	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port C pin, when programmed as output, can be selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
	$\overline{\text{DSR1}}$	Data Set Ready	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PC5.

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

Pin #	Symbol	Function	Signal Direction	Description
82	PC6	GPIO Port C	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port C pin, when programmed as output, can be selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
	$\overline{\text{DCD1}}$	Data Carrier Detect	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PC6.
83	PC7	GPIO Port C	Bidirectional	This pin can be used for general-purpose I/O. It can be individually programmed as input or output and can also be used individually as an interrupt input. Each Port C pin, when programmed as output, can be selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
	$\overline{\text{RI1}}$	Ring Indicator	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PC7.
84	V_{SS}	Ground		Ground.
85	X_{IN}	System Clock Oscillator Input	Input	This pin is the input to the onboard crystal oscillator for the primary system clock. If an external oscillator is used, its clock output should be connected to this pin. When a crystal is used, it should be connected between X_{IN} and X_{OUT}.
86	X_{OUT}	System Clock Oscillator Output	Output	This pin is the output of the onboard crystal oscillator. When used, a crystal should be connected between X_{IN} and X_{OUT}.
87	V_{DD}	Power Supply		Power Supply.

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

Pin #	Symbol	Function	Signal Direction	Description
88	PB0	GPIO Port B	Bidirectional	This pin can be used for general-purpose I/O. 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.
	T0_IN	Timer 0 In	Input	Alternate clock source for Programmable Reload Timers 0 and 2. This signal is multiplexed with PB0.
89	PB1	GPIO Port B	Bidirectional	This pin can be used for general-purpose I/O. 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.
	T1_IN	Timer 1 In	Input	Alternate clock source for Programmable Reload Timers 1 and 3. This signal is multiplexed with PB1.
90	PB2	GPIO Port B	Bidirectional	This pin can be used for general-purpose I/O. 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.
	$\overline{SS}$	Slave Select	Input, Active Low	The slave select input line is used to select a slave device in SPI mode. This signal is multiplexed with PB2.
91	PB3	GPIO Port B	Bidirectional	This pin can be used for general-purpose I/O. 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.
	SCK	SPI Serial Clock	Bidirectional	SPI serial clock. This signal is multiplexed with PB3.

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

Pin #	Symbol	Function	Signal Direction	Description
92	PB4	GPIO Port B	Bidirectional	This pin can be used for general-purpose I/O. 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.
	T4_OUT	Timer 4 Out	Output	Programmable Reload Timer 4 timer-out signal. This signal is multiplexed with PB4.
93	PB5	GPIO Port B	Bidirectional	This pin can be used for general-purpose I/O. 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.
	T5_OUT	Timer 5 Out	Output	Programmable Reload Timer 5 timer-out signal. This signal is multiplexed with PB5.
94	PB6	GPIO Port B	Bidirectional	This pin can be used for general-purpose I/O. 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 is an SPI master device and as an output when eZ80L92 is an SPI slave device. This signal is multiplexed with PB6.

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

Pin #	Symbol	Function	Signal Direction	Description
95	PB7	GPIO Port B	Bidirectional	This pin can be used for general-purpose I/O. 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 is an SPI master device and as an input when the eZ80L92 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.

Pin Characteristics

Table 2 describes the characteristics of each pin in the eZ80L92's 100-pin LQFP package.

Table 2. Pin Characteristics of the eZ80™ Webserver-i

Pin #	Symbol	Direction	Reset Direction	Active Low/High	Tristate Output	Pull Up/Down	Schmitt Trigger Input	Open Drain/Source
1	ADDR0	I/O	O	N/A	Yes	No	No	No
2	ADDR1	I/O	O	N/A	Yes	No	No	No
3	ADDR2	I/O	O	N/A	Yes	No	No	No
4	ADDR3	I/O	O	N/A	Yes	No	No	No
5	ADDR4	I/O	O	N/A	Yes	No	No	No
6	ADDR5	I/O	O	N/A	Yes	No	No	No
7	V_{DD}							

Table 2. Pin Characteristics of the eZ80™ Webserver-i (Continued)

Pin #	Symbol	Direction	Reset Direction	Active Low/High	Tristate Output	Pull Up/Down	Schmitt Trigger Input	Open Drain/Source
8	V_{SS}							
9	ADDR6	I/O	O	N/A	Yes	No	No	No
10	ADDR7	I/O	O	N/A	Yes	No	No	No
11	ADDR8	I/O	O	N/A	Yes	No	No	No
12	ADDR9	I/O	O	N/A	Yes	No	No	No
13	ADDR10	I/O	O	N/A	Yes	No	No	No
14	ADDR11	I/O	O	N/A	Yes	No	No	No
15	ADDR12	I/O	O	N/A	Yes	No	No	No
16	ADDR13	I/O	O	N/A	Yes	No	No	No
17	ADDR14	I/O	O	N/A	Yes	No	No	No
18	V_{DD}							
19	V_{SS}							
20	ADDR15	I/O	O	N/A	Yes	No	No	No
21	ADDR16	I/O	O	N/A	Yes	No	No	No
22	ADDR17	I/O	O	N/A	Yes	No	No	No
23	ADDR18	I/O	O	N/A	Yes	No	No	No
24	ADDR19	I/O	O	N/A	Yes	No	No	No
25	ADDR20	I/O	O	N/A	Yes	No	No	No
26	ADDR21	I/O	O	N/A	Yes	No	No	No
27	ADDR22	I/O	O	N/A	Yes	No	No	No
28	ADDR23	I/O	O	N/A	Yes	No	No	No
29	$\overline{\text{CS0}}$	O	O	Low	No	No	No	No
30	$\overline{\text{CS1}}$	O	O	Low	No	No	No	No
31	$\overline{\text{CS2}}$	O	O	Low	No	No	No	No
32	$\overline{\text{CS3}}$	O	O	Low	No	No	No	No
33	V_{DD}							
34	V_{SS}							
35	DATA0	I/O	I	N/A	Yes	No	No	No

Table 2. Pin Characteristics of the eZ80™ Webserver-i (Continued)

Pin #	Symbol	Direction	Reset Direction	Active Low/High	Tristate Output	Pull Up/Down	Schmitt Trigger Input	Open Drain/Source
36	DATA1	I/O	I	N/A	Yes	No	No	No
37	DATA2	I/O	I	N/A	Yes	No	No	No
38	DATA3	I/O	I	N/A	Yes	No	No	No
39	DATA4	I/O	I	N/A	Yes	No	No	No
40	DATA5	I/O	I	N/A	Yes	No	No	No
41	DATA6	I/O	I	N/A	Yes	No	No	No
42	DATA7	I/O	I	N/A	Yes	No	No	No
43	V_{DD}							
44	V_{SS}							
45	$\overline{IORQ}$	I/O	O	Low	Yes	No	No	No
46	$\overline{MREQ}$	I/O	O	Low	Yes	No	No	No
47	$\overline{RD}$	O	O	Low	No	No	No	No
48	$\overline{WR}$	O	O	Low	No	No	No	No
49	$\overline{INSTRD}$	O	O	Low	No	No	No	No
50	$\overline{WAIT}$	I	I	Low	N/A	No	No	N/A
51	$\overline{RESET}$	I	I	Low	N/A	Up	Yes	N/A
52	$\overline{NMI}$	I	I	Low	N/A	No	Yes	N/A
53	$\overline{BUSREQ}$	I	I	Low	N/A	No	No	N/A
54	$\overline{BUSACK}$	O	O	Low	No	No	No	No
55	$\overline{HALT\_SLP}$	O	O	Low	No	No	No	No
56	V_{DD}							
57	V_{SS}							
58	RTC_X_{IN}	I	I	N/A	N/A	No	No	N/A
59	RTC_X_{OUT}	I/O	U	N/A	N/A	No	No	No
60	RTC_V_{DD}							
61	V_{SS}							
62	TMS	I	I	N/A	N/A	Up	No	N/A

Table 2. Pin Characteristics of the eZ80™ Webserver-i (Continued)

Pin #	Symbol	Direction	Reset Direction	Active Low/High	Tristate Output	Pull Up/Down	Schmitt Trigger Input	Open Drain/Source
63	TCK	I	I	Rising (In) Falling (Out)	N/A	Up	No	N/A
64	TRIGOUT	I/O	O	High	Yes	No	No	No
65	TDI	I/O	I	N/A	Yes	Up	No	No
66	TDO	O	O	N/A	Yes	No	No	No
67	V_{DD}							
68	PD0	I/O	I	N/A	Yes	No	No	OD & OS
69	PD1	I/O	I	N/A	Yes	No	No	OD & OS
70	PD2	I/O	I	N/A	Yes	No	No	OD & OS
71	PD3	I/O	I	N/A	Yes	No	No	OD & OS
72	PD4	I/O	I	N/A	Yes	No	No	OD & OS
73	PD5	I/O	I	N/A	Yes	No	No	OD & OS
74	PD6	I/O	I	N/A	Yes	No	No	OD & OS
75	PD7	I/O	I	N/A	Yes	No	No	OD & OS
76	PC0	I/O	I	N/A	Yes	No	No	OD & OS
77	PC1	I/O	I	N/A	Yes	No	No	OD & OS
78	PC2	I/O	I	N/A	Yes	No	No	OD & OS
79	PC3	I/O	I	N/A	Yes	No	No	OD & OS
80	PC4	I/O	I	N/A	Yes	No	No	OD & OS
81	PC5	I/O	I	N/A	Yes	No	No	OD & OS
82	PC6	I/O	I	N/A	Yes	No	No	OD & OS
83	PC7	I/O	I	N/A	Yes	No	No	OD & OS
84	V_{SS}							
85	X_{IN}	I	I	N/A	N/A	No	No	N/A
86	X_{OUT}	O	O	N/A	No	No	No	No
87	V_{DD}							
88	PB0	I/O	I	N/A	Yes	No	No	OD & OS
89	PB1	I/O	I	N/A	Yes	No	No	OD & OS

Table 2. Pin Characteristics of the eZ80™ Webserver-i (Continued)

Pin #	Symbol	Direction	Reset Direction	Active Low/High	Tristate Output	Pull Up/Down	Schmitt Trigger Input	Open Drain/Source
90	PB2	I/O	I	N/A	Yes	No	No	OD & OS
91	PB3	I/O	I	N/A	Yes	No	No	OD & OS
92	PB4	I/O	I	N/A	Yes	No	No	OD & OS
93	PB5	I/O	I	N/A	Yes	No	No	OD & OS
94	PB6	I/O	I	N/A	Yes	No	No	OD & OS
95	PB7	I/O	I	N/A	Yes	No	No	OD & OS
96	V_{DD}							
97	V_{SS}							
98	SDA	I/O	I	N/A	Yes	Up	No	OD
99	SCL	I/O	I	N/A	Yes	Up	No	OD
100	PHI	O	O	N/A	Yes	No	No	No

Register Map

All on-chip peripheral registers are accessed in the I/O address space. All I/O operations employ 16-bit addresses. The upper byte of the 24-bit address bus is undefined during all I/O operations (ADDR[23:16] = UU). All I/O operations using 16-bit addresses within the range 0080h–00FFh are routed to the on-chip peripherals. External I/O Chip Selects are not generated if the address space programmed for the I/O Chip Selects overlaps the 0080h–00FFh address range.

Registers at unused addresses within the 0080h–00FFh range assigned to on-chip peripherals are not implemented. READ access to such addresses returns unpredictable values and WRITE access produces no effect. Table 3 diagrams the register map for the eZ80L92.

Table 3. Register Map

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page #
Programmable Reload Counter/Timers					
0080	TMR0_CTL	Timer 0 Control Register	00	R/W	81
0081	TMR0_DR_L	Timer 0 Data Register—Low Byte	00	R	82
	TMR0_RR_L	Timer 0 Reload Register—Low Byte	00	W	83
0082	TMR0_DR_H	Timer 0 Data Register—High Byte	00	R	83
	TMR0_RR_H	Timer 0 Reload Register—High Byte	00	W	84
0083	TMR1_CTL	Timer 1 Control Register	00	R/W	81
0084	TMR1_DR_L	Timer 1 Data Register—Low Byte	00	R	82
	TMR1_RR_L	Timer 1 Reload Register—Low Byte	00	W	83
0085	TMR1_DR_H	Timer 1 Data Register—High Byte	00	R	83
	TMR1_RR_H	Timer 1 Reload Register—High Byte	00	W	84
0086	TMR2_CTL	Timer 2 Control Register	00	R/W	81

Notes:

1. After an external pin reset, the Watch-Dog Timer Control register is reset to 00h. After a Watch-Dog Timer time-out reset, the Watch-Dog Timer Control register is reset to 20h.
2. When the CPU reads this register, the current sampled value of the port is read.
3. Read-only if RTC is locked; Read/Write if RTC is unlocked.
4. After an external pin reset or a WDT reset, the RTC Control register is reset to x0xxxx00b. After an RTC Alarm Sleep-Mode Recovery reset, the RTC Control register is reset to x0xxxx10b.

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page #
Programmable Reload Counter/Timers					
0087	TMR2_DR_L	Timer 2 Data Register—Low Byte	00	R	82
	TMR2_RR_L	Timer 2 Reload Register—Low Byte	00	W	83
0088	TMR2_DR_H	Timer 2 Data Register—High Byte	00	R	83
	TMR2_RR_H	Timer 2 Reload Register—High Byte	00	W	84
0089	TMR3_CTL	Timer 3 Control Register	00	R/W	81
008A	TMR3_DR_L	Timer 3 Data Register—Low Byte	00	R	82
	TMR3_RR_L	Timer 3 Reload Register—Low Byte	00	W	83
008B	TMR3_DR_H	Timer 3 Data Register—High Byte	00	R	83
	TMR3_RR_H	Timer 3 Reload Register—High Byte	00	W	84
008C	TMR4_CTL	Timer 4 Control Register	00	R/W	81
008D	TMR4_DR_L	Timer 4 Data Register—Low Byte	00	R	82
	TMR4_RR_L	Timer 4 Reload Register—Low Byte	00	W	83
008E	TMR4_DR_H	Timer 4 Data Register—High Byte	00	R	83
	TMR4_RR_H	Timer 4 Reload Register—High Byte	00	W	84
008F	TMR5_CTL	Timer 5 Control Register	00	R/W	81
0090	TMR5_DR_L	Timer 5 Data Register—Low Byte	00	R	82
	TMR5_RR_L	Timer 5 Reload Register—Low Byte	00	W	83
0091	TMR5_DR_H	Timer 5 Data Register—High Byte	00	R	83
	TMR5_RR_H	Timer 5 Reload Register—High Byte	00	W	84
0092	TMR_ISS	Timer Input Source Select Register	00	R/W	84
Watch-Dog Timer					
0093	WDT_CTL	Watch-Dog Timer Control Register[1]	00/20	R/W	73
0094	WDT_RR	Watch-Dog Timer Reset Register	XX	W	74

Notes:
1. After an external pin reset, the Watch-Dog Timer Control register is reset to 00h. After a Watch-Dog Timer time-out reset, the Watch-Dog Timer Control register is reset to 20h.
2. When the CPU reads this register, the current sampled value of the port is read.
3. Read-only if RTC is locked; Read/Write if RTC is unlocked.
4. After an external pin reset or a WDT reset, the RTC Control register is reset to x0xxxx00b. After an RTC Alarm Sleep-Mode Recovery reset, the RTC Control register is reset to x0xxxx10b.

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page #
General-Purpose Input/Output Ports					
009A	PB_DR	Port B Data Register[2]	XX	R/W	43
009B	PB_DDR	Port B Data Direction Register	FF	R/W	44
009C	PB_ALT1	Port B Alternate Register 1	00	R/W	44
009D	PB_ALT2	Port B Alternate Register 2	00	R/W	44
009E	PC_DR	Port C Data Register	XX	R/W[2]	43
009F	PC_DDR	Port C Data Direction Register	FF	R/W	44
00A0	PC_ALT1	Port C Alternate Register 1	00	R/W	44
00A1	PC_ALT2	Port C Alternate Register 2	00	R/W	44
00A2	PD_DR	Port D Data Register	XX	R/W[2]	43
00A3	PD_DDR	Port D Data Direction Register	FF	R/W	44
00A4	PD_ALT1	Port D Alternate Register 1	00	R/W	44
00A5	PD_ALT2	Port D Alternate Register 2	00	R/W	44
Chip Select/Wait State Generator					
00A8	CS0_LBR	Chip Select 0 Lower Bound Register	00	R/W	67
00A9	CS0_UBR	Chip Select 0 Upper Bound Register	FF	R/W	67
00AA	CS0_CTL	Chip Select 0 Control Register	E8	R/W	68
00AB	CS1_LBR	Chip Select 1 Lower Bound Register	00	R/W	67
00AC	CS1_UBR	Chip Select 1 Upper Bound Register	00	R/W	67
00AD	CS1_CTL	Chip Select 1 Control Register	00	R/W	68
00AE	CS2_LBR	Chip Select 2 Lower Bound Register	00	R/W	67
00AF	CS2_UBR	Chip Select 2 Upper Bound Register	00	R/W	67
00B0	CS2_CTL	Chip Select 2 Control Register	00	R/W	68
00B1	CS3_LBR	Chip Select 3 Lower Bound Register	00	R/W	67

Notes:
1. After an external pin reset, the Watch-Dog Timer Control register is reset to 00h. After a Watch-Dog Timer time-out reset, the Watch-Dog Timer Control register is reset to 20h.
2. When the CPU reads this register, the current sampled value of the port is read.
3. Read-only if RTC is locked; Read/Write if RTC is unlocked.
4. After an external pin reset or a WDT reset, the RTC Control register is reset to x0xxxx00b. After an RTC Alarm Sleep-Mode Recovery reset, the RTC Control register is reset to x0xxxx10b.

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page #
Chip Select/Wait State Generator					
00B2	CS3_UBR	Chip Select 3 Upper Bound Register	00	R/W	67
00B3	CS3_CTL	Chip Select 3 Control Register	00	R/W	68
Serial Peripheral Interface (SPI) Block					
00B8	SPI_BRG_L	SPI Baud Rate Generator Register—Low Byte	02	R/W	131
00B9	SPI_BRG_H	SPI Baud Rate Generator Register—High Byte	00	R/W	131
00BA	SPI_CTL	SPI Control Register	04	R/W	132
00BB	SPI_SR	SPI Status Register	00	R	133
00BC	SPI_TSR	SPI Transmit Shift Register	XX	W	134
	SPI_RBR	SPI Receive Buffer Register	XX	R	134
Infrared Encoder/Decoder Block					
00BF	IR_CTL	Infrared Encoder/Decoder Control	00	R/W	124
Universal Asynchronous Receiver/Transmitter 0 (UART0) Block					
00C0	UART0_RBR	UART 0 Receive Buffer Register	XX	R	110
	UART0_THR	UART 0 Transmit Holding Register	XX	W	109
	UART0_BRG_L	UART 0 Baud Rate Generator Register—Low Byte	02	R/W	108
00C1	UART0_IER	UART 0 Interrupt Enable Register	00	R/W	110
	UART0_BRG_H	UART 0 Baud Rate Generator Register—High Byte	00	R/W	108
00C2	UART0_IIR	UART 0 Interrupt Identification Register	01	R	111
	UART0_FCTL	UART 0 FIFO Control Register	00	W	112
00C3	UART0_LCTL	UART 0 Line Control Register	00	R/W	113

Notes:
1. After an external pin reset, the Watch-Dog Timer Control register is reset to 00h. After a Watch-Dog Timer time-out reset, the Watch-Dog Timer Control register is reset to 20h.
2. When the CPU reads this register, the current sampled value of the port is read.
3. Read-only if RTC is locked; Read/Write if RTC is unlocked.
4. After an external pin reset or a WDT reset, the RTC Control register is reset to x0xxxx00b. After an RTC Alarm Sleep-Mode Recovery reset, the RTC Control register is reset to x0xxxx10b.

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page #
Universal Asynchronous Receiver/Transmitter 0 (UART0) Block					
00C4	UART0_MCTL	UART 0 Modem Control Register	00	R/W	115
00C5	UART0_LSR	UART 0 Line Status Register	60	R	116
00C6	UART0_MSR	UART 0 Modem Status Register	XX	R	118
00C7	UART0_SPR	UART 0 Scratch Pad Register	00	R/W	119
I^2C Block					
00C8	I2C_SAR	I^2C Slave Address Register	00	R/W	148
00C9	I2C_XSAR	I^2C Extended Slave Address Register	00	R/W	149
00CA	I2C_DR	I^2C Data Register	00	R/W	149
00CB	I2C_CTL	I^2C Control Register	00	R/W	151
00CC	I2C_SR	I^2C Status Register	F8	R	152
	I2C_CCR	I^2C Clock Control Register	00	W	154
00CD	I2C_SRR	I^2C Software Reset Register	XX	W	155
Universal Asynchronous Receiver/Transmitter 1 (UART1) Block					
00D0	UART1_RBR	UART 1 Receive Buffer Register	XX	R	110
	UART1_THR	UART 1 Transmit Holding Register	XX	W	109
	UART1_BRG_L	UART 1 Baud Rate Generator Register—Low Byte	02	R/W	108
00D1	UART1_IER	UART 1 Interrupt Enable Register	00	R/W	110
	UART1_BRG_H	UART 1 Baud Rate Generator Register—High Byte	00	R/W	108
00D2	UART1_IIR	UART 1 Interrupt Identification Register	01	R	111
	UART1_FCTL	UART 1 FIFO Control Register	00	W	112
00D3	UART1_LCTL	UART 1 Line Control Register	00	R/W	113
00D4	UART1_MCTL	UART 1 Modem Control Register	00	R/W	115

Notes:
1. After an external pin reset, the Watch-Dog Timer Control register is reset to 00h. After a Watch-Dog Timer time-out reset, the Watch-Dog Timer Control register is reset to 20h.
2. When the CPU reads this register, the current sampled value of the port is read.
3. Read-only if RTC is locked; Read/Write if RTC is unlocked.
4. After an external pin reset or a WDT reset, the RTC Control register is reset to x0xxxx00b. After an RTC Alarm Sleep-Mode Recovery reset, the RTC Control register is reset to x0xxxx10b.

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page #
Universal Asynchronous Receiver/Transmitter 1 (UART1) Block					
00D5	UART1_LSR	UART 1 Line Status Register	60	R/W	116
00D6	UART1_MSR	UART 1 Modem Status Register	XX	R/W	118
00D7	UART1_SPR	UART 1 Scratch Pad Register	00	R/W	119
Low-Power Control					
00DB	CLK_PPD1	Clock Peripheral Power-Down Register 1	00	R/W	37
00DC	CLK_PPD2	Clock Peripheral Power-Down Register 2	00	R/W	38
Real-Time Clock					
00E0	RTC_SEC	RTC Seconds Register[3]	XX	R/W	88
00E1	RTC_MIN	RTC Minutes Register	XX	R/W[3]	89
00E2	RTC_HRS	RTC Hours Register	XX	R/W[3]	90
00E3	RTC_DOW	RTC Day-of-the-Week Register	XX	R/W[3]	91
00E4	RTC_DOM	RTC Day-of-the-Month Register	XX	R/W[3]	92
00E5	RTC_MON	RTC Month Register	XX	R/W[3]	93
00E6	RTC_YR	RTC Year Register	XX	R/W[3]	94
00E7	RTC_CEN	RTC Century Register	XX	R/W[3]	95
00E8	RTC_ASEC	RTC Alarm Seconds Register	XX	R/W	96
00E9	RTC_AMIN	RTC Alarm Minutes Register	XX	R/W	97
00EA	RTC_AHRS	RTC Alarm Hours Register	XX	R/W	98
00EB	RTC_ADOW	RTC Alarm Day-of-the-Week Register	0X	R/W	99
00EC	RTC_ACTRL	RTC Alarm Control Register	00	R/W	100
00ED	RTC_CTRL	RTC Control Register[4]	x0xxxx00b/ x0xxxx10b	R/W	101

Notes:
1. After an external pin reset, the Watch-Dog Timer Control register is reset to 00h. After a Watch-Dog Timer time-out reset, the Watch-Dog Timer Control register is reset to 20h.
2. When the CPU reads this register, the current sampled value of the port is read.
3. Read-only if RTC is locked; Read/Write if RTC is unlocked.
4. After an external pin reset or a WDT reset, the RTC Control register is reset to x0xxxx00b. After an RTC Alarm Sleep-Mode Recovery reset, the RTC Control register is reset to x0xxxx10b.

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page #
Chip Select Bus Mode Control					
00F0	CS0_BMC	Chip Select 0 Bus Mode Control Register	02h	R/W	69
00F1	CS1_BMC	Chip Select 1 Bus Mode Control Register	02h	R/W	69
00F2	CS2_BMC	Chip Select 2 Bus Mode Control Register	02h	R/W	69
00F3	CS3_BMC	Chip Select 3 Bus Mode Control Register	02h	R/W	69

Notes:

1. After an external pin reset, the Watch-Dog Timer Control register is reset to 00h. After a Watch-Dog Timer time-out reset, the Watch-Dog Timer Control register is reset to 20h.
2. When the CPU reads this register, the current sampled value of the port is read.
3. Read-only if RTC is locked; Read/Write if RTC is unlocked.
4. After an external pin reset or a WDT reset, the RTC Control register is reset to x0xxxx00b. After an RTC Alarm Sleep-Mode Recovery reset, the RTC Control register is reset to x0xxxx10b.

eZ80® CPU Core

The eZ80® CPU is the first 8-bit microprocessor to support 16 MB linear addressing. Each software module or task under a real-time executive or operating system can operate in Z80-compatible (64 KB) mode or full 24-bit (16 MB) address mode.

The eZ80® CPU instruction set is a superset of the instruction sets for the Z80 and Z180 CPUs. Z80 and Z180 programs can be executed on an eZ80® CPU with little or no modification.

Features

- Code-compatible with Z80 and Z180 products
- 24-bit linear address space
- Single-cycle instruction fetch
- Pipelined fetch, decode, and execute
- Dual Stack Pointers for ADL (24-bit) and Z80 (16-bit) memory modes
- 24-bit CPU registers and ALU (Arithmetic Logic Unit)
- Debug support
- Nonmaskable Interrupt (NMI), plus support for 128 maskable vectored interrupts

New and Improved Instructions

- Four new block transfer instructions provide DMA-like operations for memory to I/O and I/O to memory transfers. These new instructions are:
  - INDRX (input from I/O, decrement the memory address, leave the I/O address unchanged, and repeat)
  - INIRX (input from I/O, increment the memory address, leave the I/O address unchanged, and repeat)
  - OTDRX (output to I/O, decrement the memory address, leave the I/O address unchanged, and repeat)
  - OTIRX (output to I/O, increment the memory address, leave the I/O address unchanged, and repeat)

- Four other block transfer instructions are modified to improve performance relative to the eZ80190 device. These modified instructions are:
  - IND2R (input from I/O, decrement the memory address, decrement the I/O address, and repeat)
  - INI2R (input from I/O, increment the memory address, increment the I/O address, and repeat)
  - OTD2R (output to I/O, decrement the memory address, decrement the I/O address, and repeat)
  - OTI2R (output to I/O, increment the memory address, increment the I/O address, and repeat)

For more information on the eZ80® CPU, its instruction set, and eZ80® programming, please refer to the eZ80 CPU User Manual. For more information on the eZ80190, please refer to the eZ80190 Product Specification.

Reset

RESET Operation

The RESET controller within the eZ80L92 provides a consistent system reset (RESET) function for all type of resets that may affect the system. There are 4 events which can cause a RESET:

- External $\overline{\text{RESET}}$ pin assertion
- Watch-Dog Timer (WDT) time-out when configured to generate a RESET
- Real-Time Clock alarm with the eZ80® CPU in low-power SLEEP mode
- Execution of a Debug RESET command

During RESET, an internal RESET mode timer holds the system in RESET for 257 system clock (SCLK) cycles. The RESET mode timer begins incrementing on the next rising edge of SCLK following deactivation of all RESET events (RESET pin, Watch-Dog Timer, Real-Time Clock, Debugger)

Note: User must determine is 257 SCLK cycles provides sufficient time for the primary crystal oscillator to stabilize.

RESET, via the external $\overline{\text{RESET}}$ pin, must always be executed following application of power (V_{DD} ramp). Without RESET following power-up, proper operation of the eZ80L92 cannot be guaranteed.

Low-Power Modes

Overview

The eZ80L92 provides a range of power-saving features. The highest level of power reduction is provided by SLEEP mode. The next level of power reduction is provided by the HALT instruction. The lowest level of power reduction is provided by the clock peripheral power-down registers.

SLEEP Mode

Execution of the eZ80® CPU's SLP instruction places the eZ80L92 into SLEEP mode. In SLEEP mode, the operating characteristics are:

- Primary crystal oscillator is disabled
- System clock is disabled
- eZ80® CPU is idle
- Program counter (PC) stops incrementing
- 32KHz crystal oscillator continues to operate and drive the Real-Time Clock and the Watch-Dog Timer (if WDT is configured to operate from the 32KHz oscillator)

The eZ80® CPU can be brought out of SLEEP mode by any of the following operations:

- RESET via the external $\overline{\text{RESET}}$ pin driven Low
- RESET via a Real-Time Clock alarm
- RESET via a Watch-Dog Timer time-out (if running off of the 32KHz oscillator and configured to generate a RESET upon time-out)
- RESET via execution of a Debug RESET command

After exiting SLEEP mode, the standard RESET delay occurs to allow the primary crystal oscillator to stabilize. Refer to the Reset section on page 34 for more information.

HALT Mode

Execution of the eZ80® CPU's HALT instruction places the eZ80L92 into HALT mode. In HALT mode, the operating characteristics are:

- Primary crystal oscillator is enabled and continues to operate
- System clock is enabled and continues to operate
- eZ80® CPU is idle
- Program counter (PC) stops incrementing

The eZ80® CPU can be brought out of HALT mode by any of the following operations:

- Nonmaskable interrupt (NMI)
- Maskable interrupt
- RESET via the external $\overline{\text{RESET}}$ pin driven Low
- Watch-Dog Timer time-out (if configured to generate either an NMI or RESET upon time-out)
- RESET via execution of a Debug RESET command

To minimize current in HALT mode, the system clock should be disabled for all unused on-chip peripherals via the Clock Peripheral Power-Down Registers.

Clock Peripheral Power-Down Registers

To reduce power, the Clock Peripheral Power-Down Registers allow the system clock to be disabled to unused on-chip peripherals. Upon RESET, all peripherals are enabled. The clock to unused peripherals can be disabled by setting the appropriate bit in the Clock Peripheral Power-Down Registers to 1. When powered down, the peripherals are completely disabled. To reenable, the bit in the Clock Peripheral Power-Down Registers must be cleared to 0.

Many peripherals feature separate enable/disable control bits that must be appropriately set for operation. These peripheral specific enable/disable bits do not provide the same level of power reduction as the Clock Peripheral Power-Down Registers. When powered down, the standard peripheral control registers are not accessible for read or write access. See Tables 4 and 5.

Table 4. Clock Peripheral Power-Down Register 1 (CLK_PPD1 = 00DBh)

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	R/W	R/W	R/W	R/W

Note: R/W = Read/Write; R = Read Only.

Bit Position	Value	Description
7 GPIO_D_OFF	1	System clock to GPIO Port D is powered down. Port D alternate functions do not operate correctly.
	0	System clock to GPIO Port D is powered up.
6 GPIO_C_OFF	1	System clock to GPIO Port C is powered down. Port C alternate functions do not operate correctly.
	0	System clock to GPIO Port C is powered up.
5 GPIO_B_OFF	1	System clock to GPIO Port B is powered down. Port B alternate functions do not operate correctly.
	0	System clock to GPIO Port B is powered up.
4		Reserved.
3 SPI_OFF	1	System clock to SPI is powered down.
	0	System clock to SPI is powered up.
2 I2C_OFF	1	System clock to I^2C is powered down.
	0	System clock to I^2C is powered up.
1 UART1_OFF	1	System clock to UART1 is powered down.
	0	System clock to UART1 is powered up.
0 UART0_OFF	1	System clock to UART0 and IrDA endec is powered down.
	0	System clock to UART0 and IrDA endec is powered up.

Table 5. Clock Peripheral Power-Down Register 2 (CLK_PPD2 = 00DCh)

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

Note: R/W = Read/Write; R = Read Only.

Bit Position	Value	Description
7 PHI_OFF	1	PHI Clock output is disabled (output is high-impedance).
	0	PHI Clock output is enabled.
6	0	Reserved.
5 PRT5_OFF	1	System clock to PRT5 is powered down.
	0	System clock to PRT5 is powered up.
4 PRT4_OFF	1	System clock to PRT4 is powered down.
	0	System clock to PRT4 is powered up.
3 PRT3_OFF	1	System clock to PRT3 is powered down.
	0	System clock to PRT3 is powered up.
2 PRT2_OFF	1	System clock to PRT2 is powered down.
	0	System clock to PRT2 is powered up.
1 PRT1_OFF	1	System clock to PRT1 is powered down.
	0	System clock to PRT1 is powered up.
0 PRT0_OFF	1	System clock to PRT0 is powered down.
	0	System clock to PRT0 is powered up.

General-Purpose Input/Output

GPIO Overview

The eZ80L92 features 24 General-Purpose Input/Output (GPIO) pins. The GPIO pins are assembled as three 8-bit ports— Port B, Port C, and Port D. All port signals can be configured for use as either inputs or outputs. In addition, all of the port pins can be used as vectored interrupt sources for the eZ80® CPU.

GPIO Operation

The GPIO operation is the same for all 3 GPIO ports (Ports B, C, and D). Each port features eight GPIO port pins. The operating mode for each pin is controlled by four bits that are divided between four 8-bit registers. These GPIO mode control registers are:

- Port *x* Data Register (Px_DR)
- Port *x* Data Direction Register (Px_DDR)
- Port *x* Alternate Register 1 (Px_ALT1)
- Port *x* Alternate Register 2 (Px_ALT2)

where *x* can be *B*, *C*, or *D* representing any of the three GPIO ports B, C, or D. The mode for each pin is controlled by setting each register bit pertinent to the pin to be configured. For example, the operating mode for Port B Pin 7 (PB7), is set by the values contained in PB_DR[7], PB_DDR[7], PB_ALT1[7], and PB_ALT2[7].

The combination of the GPIO control register bits allows individual configuration of each port pin for nine modes. In all modes, reading of the Port *x* Data register returns the sampled state, or level, of the signal on the corresponding pin. Table 6 indicates the function of each port signal based upon these four register bits. After a RESET event, all GPIO port pins are configured as standard digital inputs, with interrupts disabled.

Table 6. GPIO Mode Selection

GPIO Mode	Px_ALT2 Bits7:0	Px_ALT1 Bits7:0	Px_DDR Bits7:0	Px_DR Bits7:0	Port Mode	Output
1	0	0	0	0	Output	0
	0	0	0	1	Output	1
2	0	0	1	0	Input from pin	High impedance
	0	0	1	1	Input from pin	High impedance
3	0	1	0	0	Open-Drain output	0
	0	1	0	1	Open-Drain I/O	High impedance
4	0	1	1	0	Open source I/O	High impedance
	0	1	1	1	Open source output	1
5	1	0	0	0	Reserved	High impedance
6	1	0	0	1	Interrupt—dual edge triggered	High impedance
7	1	0	1	0	Port B, C, or D—alternate function controls port I/O.	
	1	0	1	1	Port B, C, or D—alternate function controls port I/O.	
8	1	1	0	0	Interrupt—active Low	High impedance
	1	1	0	1	Interrupt—active High	High impedance
9	1	1	1	0	Interrupt—falling edge triggered	High impedance
	1	1	1	1	Interrupt—rising edge triggered	High impedance

GPIO Mode 1. The port pin is configured as a standard digital output pin. The value written to the Port *x* Data register (P*x*_DR) is presented on the pin.

GPIO Mode 2. The port pin is configured as a standard digital input pin. The output is tristated (high impedance). The value stored in the Port *x* Data register produces no effect. As in all modes, a Read from the Port *x* Data register returns the pin's value. GPIO Mode 2 is the default operating mode following a RESET.

GPIO Mode 3. The port pin is configured as open-drain I/O. The GPIO pins do not feature an internal pull-up to the supply voltage. To employ the GPIO pin in OPEN-DRAIN mode, an external pull-up resistor must connect the pin to the supply voltage. Writing a 0 to the Port *x* Data register outputs a Low at the pin. Writing a 1 to the Port *x* Data register results in high-impedance output.

GPIO Mode 4. The port pin is configured as open-source I/O. The GPIO pins do not feature an internal pull-down to the supply ground. To employ the GPIO pin in OPEN-SOURCE mode, an external pull-down resistor must connect the pin to the supply ground. Writing a 1 to the Port *x* Data register outputs a High at the pin. Writing a 0 to the Port *x* Data register results in a high-impedance output.

GPIO Mode 5. Reserved. This pin produces high-impedance output.

GPIO Mode 6. This bit enables a dual edge-triggered interrupt mode. Both a rising and a falling edge on the pin cause an interrupt request to be sent to the eZ80® CPU. Writing a 1 to the Port *x* Data register bit position resets the corresponding interrupt request. Writing a 0 produces no effect. The programmer must set the Port *x* Data register before entering the edge-triggered interrupt mode.

GPIO Mode 7. For Ports B, C, and D, the port pin is configured to pass control over to the alternate (secondary) functions assigned to the pin. For example, the alternate mode function for PC7 is RI1 and the alternate mode function for PB4 is the Timer 4 Out. When GPIO Mode 7 is enabled, the pin output data and pin tristated control come from the alternate function's data output and tristate control, respectively. The value in the Port *x* Data register produces no effect on operation.

Note: Input signals are sampled by the system clock before being passed to the alternate function input.

GPIO Mode 8. The port pin is configured for level-sensitive interrupt modes. An interrupt request is generated when the level at the pin is the same as the level stored in the Port *x* Data register. The port pin value is sampled by the system clock. The input pin must be held at the selected interrupt level for a minimum of 2 clock periods to initiate an interrupt. The interrupt request remains active as long as this condition is maintained at the external source.

GPIO Mode 9. The port pin is configured for single edge-triggered interrupt mode. The value in the Port *x* Data register determines if a positive or negative edge causes an interrupt request. A 0 in the Port *x* Data register bit sets the selected pin to generate an interrupt request for falling edges. A 1 in the Port *x* Data register bit sets the selected pin to generate an interrupt request for rising edges. The interrupt request remains active until a 1 is written to the corresponding interrupt request of the Port *x* Data register bit. Writing a 0 produces no effect on operation. The programmer must set the Port *x* Data register before entering the edge-triggered interrupt mode.

A simplified block diagram of a GPIO port pin is illustrated in Figure 3.

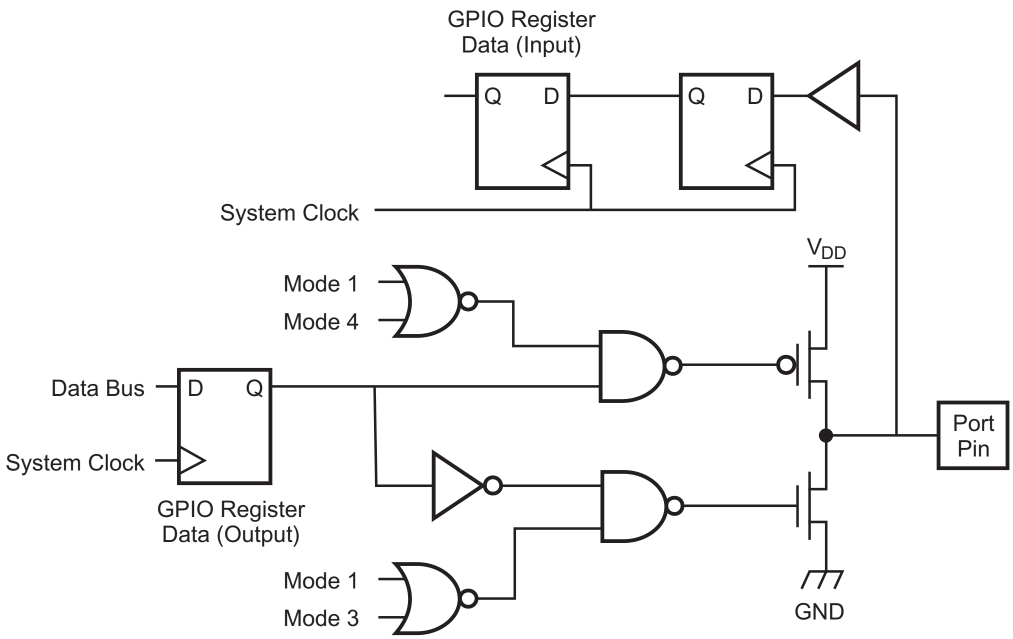


Figure 3. GPIO Port Pin Block Diagram

GPIO Interrupts

Each port pin can be used as an interrupt source. Interrupts can be either level- or edge-triggered.

Level-Triggered Interrupts

When the port is configured for level-triggered interrupts, the corresponding port pin is tristated. An interrupt request is generated when the level at the pin is the same as the level stored in the Port *x* Data register. The port pin value is sampled by the system clock. The input pin must be held at the selected interrupt level for a minimum of 2 consecutive clock cycles to initiate an interrupt. The interrupt request remains active as long as this condition is maintained at the external source.

For example, if PD3 is programmed for low-level interrupt and the pin is forced Low for 2 consecutive clock cycles, an interrupt request signal is generated from that port pin and sent to the eZ80® CPU. The interrupt request signal remains active until the external device driving PD3 forces the pin High.

Edge-Triggered Interrupts

When the port is configured for edge-triggered interrupts, the corresponding port pin is tristated. If the pin receives the correct edge from an external device, the port pin generates an interrupt request signal to the eZ80® CPU. Any time a port pin is configured for edge-triggered interrupt, writing a 1 to that pin's Port *x* Data register causes a reset of the edge-detected interrupt. The programmer must set the bit in the Port *x* Data register to 1 before entering either single or dual edge-triggered interrupt mode for that port pin.

When configured for dual edge-triggered interrupt mode (GPIO Mode 6), both a rising and a falling edge on the pin cause an interrupt request to be sent to the eZ80® CPU.

When configured for single edge-triggered interrupt mode (GPIO Mode 9), the value in the Port *x* Data register determines if a positive or negative edge causes an interrupt request. A 0 in the Port *x* Data register bit sets the selected pin to generate an interrupt request for falling edges. A 1 in the Port *x* Data register bit sets the selected pin to generate an interrupt request for rising edges.

GPIO Control Registers

The 12 GPIO Control Registers operate in groups of four with a set for each Port (B, C, and D). Each GPIO port features a Port Data register, Port Data Direction register, Port Alternate register 1, and Port Alternate register 2.

Port *x* Data Registers

When the port pins are configured for one of the output modes, the data written to the Port *x* Data registers, detailed in Table 7, are driven on the corresponding pins. In all modes, reading from the Port *x* Data registers always returns the current sampled value of the corresponding pins. When the port pins are configured as edge-triggered interrupt sources, writing a 1 to the corresponding bit in the Port *x* Data register clears the interrupt signal that is sent to the eZ80® CPU. When the port pins are configured for edge-selectable interrupts or level-sensitive interrupts, the value written to the Port *x* Data register bit selects the interrupt edge or interrupt level. See Table 6 for more information.

Table 7. Port *x* Data Registers (PB_DR = 009Ah, PC_DR = 009Eh, PD_DR = 00A2h)

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

Note: X = Undefined; R/W = Read/Write.

Port x Data Direction Registers

In conjunction with the other GPIO Control Registers, the Port *x* Data Direction registers, detailed in Table 8, control the operating modes of the GPIO port pins. See Table 6 for more information.

Table 8. Port *x* Data Direction Registers (PB_DDR = 009Bh, PC_DDR = 009Fh, PD_DDR = 00A3h)

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

Note: R/W = Read/Write.

Port x Alternate Register 1

In conjunction with the other GPIO Control Registers, the Port *x* Alternate Register 1, detailed in Table 9, control the operating modes of the GPIO port pins. See Table 6 for more information.

Table 9. Port *x* Alternate Registers 1 (PB_ALT1 = 009Ch, PC_ALT1 = 00A0h, PD_ALT1 = 00A4h)

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.

Port x Alternate Register 2

In conjunction with the other GPIO Control Registers, the Port *x* Alternate Register 2, detailed in Table 10, control the operating modes of the GPIO port pins. See Table 6 for more information.

Table 10. Port x Alternate Registers 2 (PB_ALT2 = 009Dh, PC_ALT2 = 00A1h, PD_ALT2 = 00A5h)

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.

Interrupt Controller

The interrupt controller on the eZ80L92 routes the interrupt request signals from the internal peripherals and external devices (via the GPIO pins) to the eZ80® CPU.

Maskable Interrupts

On the eZ80L92, all maskable interrupts use the eZ80® CPU's vectored interrupt function. Table 11 lists the low-byte vector for each of the maskable interrupt sources. The maskable interrupt sources are listed in order of their priority, with vector `00h` being the highest-priority interrupt. The full 16-bit interrupt vector is located at starting address {I[7:0], IVECT[7:0]} where I[7:0] is the eZ80® CPU's Interrupt Page Address Register.

Table 11. Interrupt Vector Sources by Priority

Vector	Source	Vector	Source	Vector	Source	Vector	Source
00h	Unused	1Ah	UART 1	34h	Port B 2	4Eh	Port C 7
02h	Unused	1Ch	I^2C	36h	Port B 3	50h	Port D 0
04h	Unused	1Eh	SPI	38h	Port B 4	52h	Port D 1
06h	Unused	20h	Unused	3Ah	Port B 5	54h	Port D 2
08h	Unused	22h	Unused	3Ch	Port B 6	56h	Port D 3
0Ah	PRT 0	24h	Unused	3Eh	Port B 7	58h	Port D 4
0Ch	PRT 1	26h	Unused	40h	Port C 0	5Ah	Port D 5
0Eh	PRT 2	28h	Unused	42h	Port C 1	5Ch	Port D 6
10h	PRT 3	2Ah	Unused	44h	Port C 2	5Eh	Port D 7
12h	PRT 4	2Ch	Unused	46h	Port C 3	60h	Unused
14h	PRT 5	2Eh	Unused	48h	Port C 4	62h	Unused
16h	RTC	30h	Port B 0	4Ah	Port C 5	64h	Unused
18h	UART 0	32h	Port B 1	4Ch	Port C 6	66h	Unused

Note: Absolute locations 00h, 08h, 10h, 18h, 20h, 28h, 30h, 38h, and 66h are reserved for hardware reset, NMI, and the RST instruction.

The user's program should store the interrupt service routine starting address in the two-byte interrupt vector locations. For example, for ADL mode the two-byte address for the SPI interrupt service routine would be stored at {00h, I[7:0], 1Eh} and {00h, I[7:0], 1Fh}. In Z80 mode, the two-byte address for the SPI interrupt service routine would be stored at {MBASE[7:0], I[7:0], 1Eh} and {MBASE, I[7:0], 1Fh}. The least significant byte is stored at the lower address.

When any one or more of the interrupt requests (IRQs) become active, an interrupt request is generated by the interrupt controller and sent to the CPU. The corresponding 8-bit interrupt vector for the highest priority interrupt is placed on the 8-bit interrupt vector bus, IVECT[7:0]. The interrupt vector bus is internal to the eZ80L92 and is therefore not visible externally. The response time of the eZ80® CPU to an interrupt request is a function of the current instruction being executed as well as the number of WAIT states being asserted. The interrupt vector, {I[7:0], IVECT[7:0]}, is visible on the address bus, ADDR[15:0], when the interrupt service routine begins. The response of the eZ80® CPU to a vectored interrupt on the eZ80L92 is explained in Table 12. Interrupt sources are required to be active until the Interrupt Service Routine (ISR) starts. It is recommended that the Interrupt Page Address Register (I) value be changed by the user from its default value of 00h as this address can create conflicts between the nonmaskable interrupt vector, the RST instruction addresses, and the maskable interrupt vectors.

Table 12. Vectored Interrupt Operation

Memory Mode	ADL Bit	MADL Bit	Operation
Z80 Mode	0	0	Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT [7:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is effectively {MBASE, PC[15:0]}. • Push the 2-byte return address PC[15:0] onto the ({MBASE,SPS}) stack. • The ADL mode bit remains cleared to 0. • The interrupt vector address is located at { MBASE, I[7:0], IVECT[7:0] }. • PC[15:0] ← ({ MBASE, I[7:0], IVECT[7:0] }). • The ending Program Counter is effectively {MBASE, PC[15:0]} • The interrupt service routine must end with RETI.
ADL Mode	1	0	Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT [7:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is PC[23:0]. • Push the 3-byte return address, PC[23:0], onto the SPL stack. • The ADL mode bit remains set to 1. • The interrupt vector address is located at { 00h, I[7:0], IVECT[7:0] }. • PC[15:0] ← ({ 00h, I[7:0], IVECT[7:0] }). • The ending Program Counter is { 00h, PC[15:0] }. • The interrupt service routine must end with RETI.

Table 12. Vectored Interrupt Operation (Continued)

Memory Mode	ADL Bit	MADL Bit	Operation
Z80 Mode	0	1	Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT[7:0], bus by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is effectively {MBASE, PC[15:0]}. • Push the 2-byte return address, PC[15:0], onto the SPL stack. • Push a 00h byte onto the SPL stack to indicate an interrupt from Z80 mode (because ADL = 0). • Set the ADL mode bit to 1. • The interrupt vector address is located at { 00h, I[7:0], IVECT[7:0] }. • PC[15:0] ← ({ 00h, I[7:0], IVECT[7:0] }). • The ending Program Counter is { 00h, PC[15:0] }. • The interrupt service routine must end with RETI.L
ADL Mode	1	1	Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT [7:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is PC[23:0]. • Push the 3-byte return address, PC[23:0], onto the SPL stack. • Push a 01h byte onto the SPL stack to indicate a restart from ADL mode (because ADL = 1). • The ADL mode bit remains set to 1. • The interrupt vector address is located at {00h, I[7:0], IVECT[7:0]}. • PC[15:0] ← ({ 00h, I[7:0], IVECT[7:0] }). • The ending Program Counter is { 00h, PC[15:0] }. • The interrupt service routine must end with RETI.L

Nonmaskable Interrupts

An active Low input on the $\overline{\text{NMI}}$ pin generates an interrupt request to the eZ80® CPU. This nonmaskable interrupt is always serviced by the eZ80® CPU, regardless of the state of the Interrupt Enable flags (IEF1 and IEF2). The nonmaskable interrupt is prioritized higher than all maskable interrupts. The response of the eZ80® CPU to a nonmaskable interrupt is described in detail in the eZ80® CPU User Manual (UM0077).

Chip Selects and Wait States

The eZ80L92 generates four Chip Selects for external devices. Each Chip Select may be programmed to access either memory space or I/O space. The Memory Chip Selects can be individually programmed on a 64 KB boundary. The I/O Chip Selects can each choose a 256-byte section of I/O space. In addition, each Chip Select may be programmed for up to 7 wait states.

Memory and I/O Chip Selects

Each of the Chip Selects can be enabled for either the memory address space or the I/O address space, but not both. To select the memory address space for a particular Chip Select, CSx_IO (CS*x*_CTL[4]) must be reset to 0. To select the I/O address space for a particular Chip Select, CSx_IO must be set to 1. After RESET, the default is for all Chip Selects to be configured for the memory address space. For either the memory address space or the I/O address space, the individual Chip Selects must be enabled by setting CSx_EN (CS*x*_CTL[3]) to 1.

Memory Chip Select Operation

Operation of each of the Memory Chip Selects is controlled by three control registers. To enable a particular Memory Chip Select, the following conditions must be met:

- The Chip Select is enabled by setting CSx_EN to 1
- The Chip Select is configured for Memory by clearing CSx_IO to 0
- The address is in the associated Chip Select range:

  CS*x*_LBR[7:0] ≤ ADDR[23:16] ≤ CS*x*_UBR[7:0]

- No higher priority (lower number) Chip Select meets the above conditions
- A memory access instruction must be executing

If all of the foregoing conditions are met to generate a Memory Chip Select, then the following actions occur:

- The appropriate Chip Select—$\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS2}}$, or $\overline{\text{CS3}}$—is asserted (driven Low)
- $\overline{\text{MREQ}}$ is asserted (driven Low)
- Depending upon the instruction, either $\overline{\text{RD}}$ or $\overline{\text{WR}}$ is asserted (driven Low)

If the upper and lower bounds are set to the same value (CS*x*_UBR = CS*x*_LBR), then a particular Chip Select is valid for a single 64 KB page.

Memory Chip Select Priority

A lower-numbered Chip Select is granted priority over a higher-numbered Chip Select. For example, if the address space of Chip Select 0 overlaps the Chip Select 1 address space, Chip Select 0 is active.

RESET States

On RESET, Chip Select 0 is active for all addresses, because its Lower Bound register resets to `00h` and its Upper Bound register resets to `FFh`. All of the other Chip Select Lower and Upper Bound registers reset to `00h`.

Memory Chip Select Example

The use of Memory Chip Selects is demonstrated in Figure 4. The associated control register values indicated in Table 13. In this example, all 4 Chip Selects are enabled and configured for memory addresses. Also, CS1 overlaps with CS0. Because CS0 is prioritized higher than CS1, CS1 is not active for much of its defined address space.

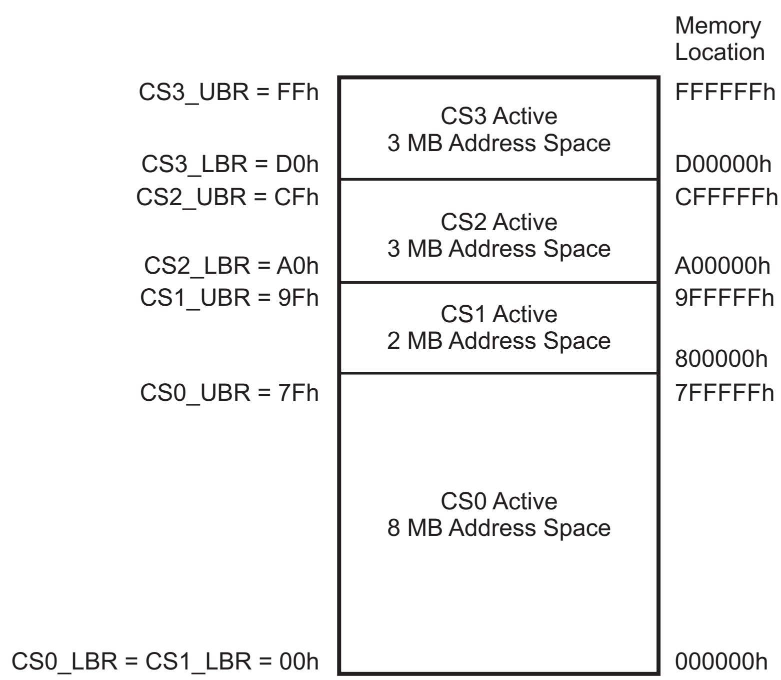


Figure 4. Memory Chip Select Example

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 can only be active when the CPU is performing I/O instructions. Because 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 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 can always be 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 met:

- 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

If all of the foregoing conditions are met to generate an I/O Chip Select, then the following actions occur:

- The appropriate Chip Select—CS0, CS1, CS2, or CS3—is asserted (driven Low)
- IORQ is asserted (driven Low)
- Depending upon the instruction, either RD or WR is asserted (driven Low)

WAIT States

For each of the Chip Selects, programmable WAIT states can be asserted to provide external devices with additional clock cycles to complete their Read or Write operations. The number of WAIT states for a particular Chip Select is controlled by the 3-bit field CS*x*_WAIT (CS*x*_CTL[7:5]). The WAIT states can be independently programmed to provide 0 to 7 WAIT states for each Chip Select. The WAIT states idle the CPU for the specified number of system clock cycles.

WAIT Input Signal

Similar to the programmable WAIT states, an external peripheral can drive the WAIT input pin to force the CPU to provide additional clock cycles to complete its Read or Write operation. Driving the WAIT pin Low stalls the CPU. The CPU resumes operation on the first rising edge of the internal system clock following deassertion of the WAIT pin.

Caution: If the WAIT pin is to be driven by an external device, the corresponding Chip Select for the device must be programmed to provide at least one WAIT state. Due to input sampling of the WAIT input pin (shown in Figure 5), one programmable WAIT state is required to allow the external peripheral sufficient time to assert the WAIT pin. It is recommended that the corresponding Chip Select for the external device be programmed to provide the maximum number of WAIT states (seven).

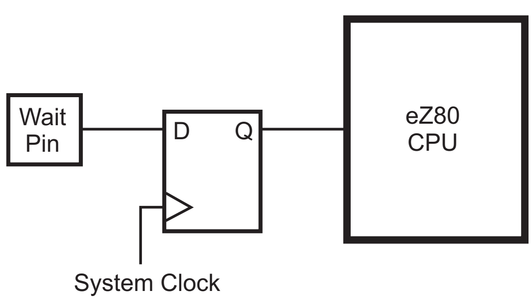


Figure 5. Wait Input Sampling Block Diagram

An example of WAIT state operation is illustrated in Figure 6. In this example, the Chip Select is configured to provide a single WAIT state. The external peripheral being accessed drives the WAIT pin Low to request assertion of an additional WAIT state. If the WAIT pin is asserted for additional system clock cycles, WAIT states are added until the WAIT pin is deasserted (High).

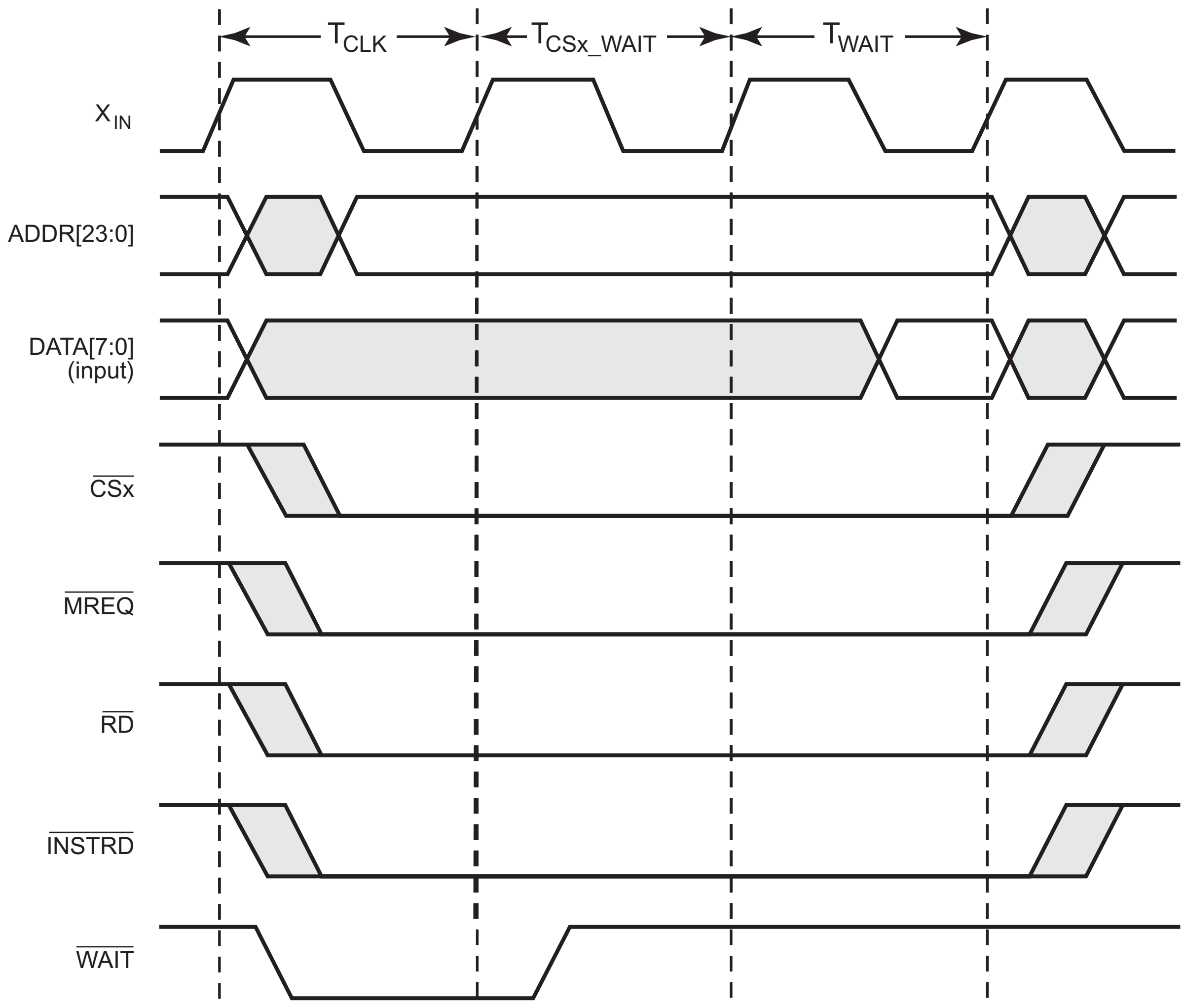


Figure 6. Wait State Operation Example (Read Operation)

Chip Selects During Bus Request/Bus Acknowledge Cycles

When the CPU relinquishes the address bus to an external peripheral in response to an external bus request (BUSREQ), it drives the bus acknowledge pin (BUSACK) Low. The external peripheral can then drive the address bus (and data bus). The CPU continues to generate Chip Select signals in response to the address on the bus. External devices cannot access the internal registers of the eZ80L92.

Bus Mode Controller

The bus mode controller allows the address and data bus timing and signal formats of the eZ80L92 to be configured to connect seamlessly with external eZ80®, Z80-, Intel-, or Motorola-compatible devices. Bus modes for each of the chip selects can be configured independently using the Chip Select Bus Mode Control Registers. The number of eZ80® system clock cycles per bus mode state is also independently programmable. For Intel bus mode, multiplexed address and data can be selected in which the lower byte of the address and the data byte both use the data bus, DATA[7:0]. Each of the bus modes is explained in more detail in the following sections.

eZ80 Bus Mode

Chip selects configured for eZ80 Bus Mode do not modify the bus signals from the CPU. The timing diagrams for external Memory and I/O Read and Write operations are shown in the AC Characteristics section on page 204. The default mode for each chip select is eZ80 mode.

Z80 Bus Mode

Chip selects configured for Z80 mode modify the eZ80® bus signals to match the Z80 microprocessor address and data bus interface signal format and timing. During Read operations, the Z80 Bus Mode employs three states (T1, T2, and T3) as described in Table 14.

Table 14. Z80 Bus Mode Read States

STATE T1	The READ cycle begins in State T1. The CPU drives the address onto the address bus and the associated Chip Select signal is asserted.
STATE T2	During State T2, the $\overline{RD}$ signal is asserted. Depending upon the instruction, either the MREQ or IORQ signal is asserted. If the external 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 WAIT pin is driven High.
STATE T3	During State T3, no bus signals are altered. The data is latched by the eZ80L92 at the rising edge of the eZ80® system clock at the end of State T3.

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 $\overline{WR}$ signal is asserted. Depending upon the instruction, either the MREQ or IORQ signal is asserted. If the external 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 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 Figures 7 and 8. The Z80 Bus Mode states can be configured for 1 to 15 eZ80® system clock cycles. In the figures, each Z80 Bus Mode state is two eZ80® system clock cycles in duration. Figures 7 and 8 also illustrate the assertion of 1 WAIT state (T_{WAIT}) by the external peripheral during each Z80 Bus Mode cycle.

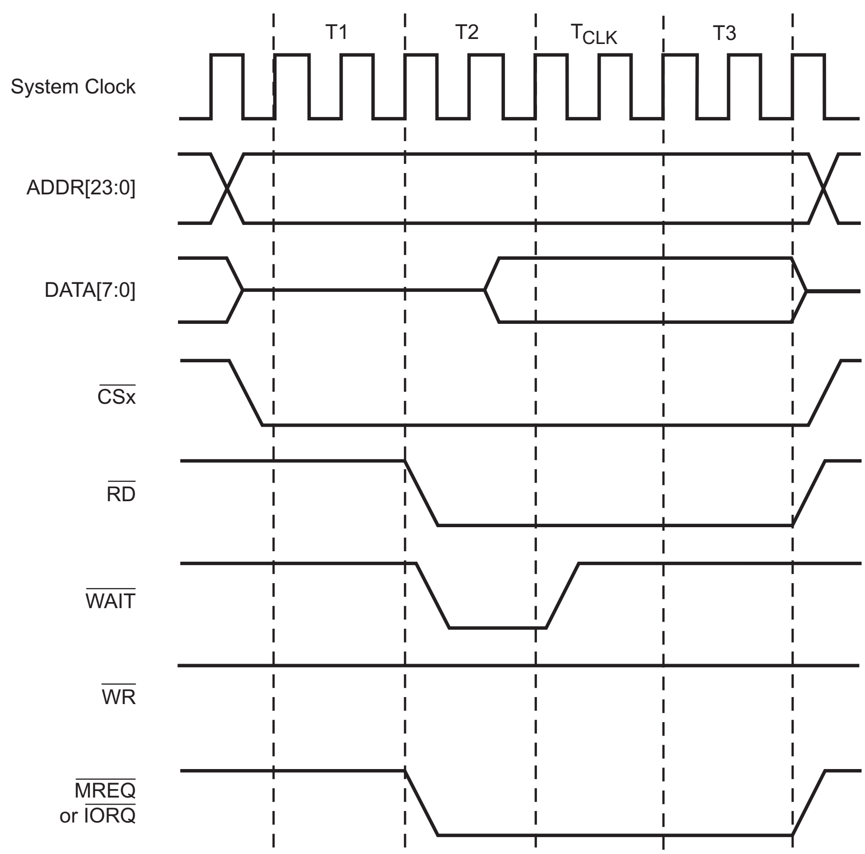


Figure 7. Z80 Bus Mode Read Timing Example

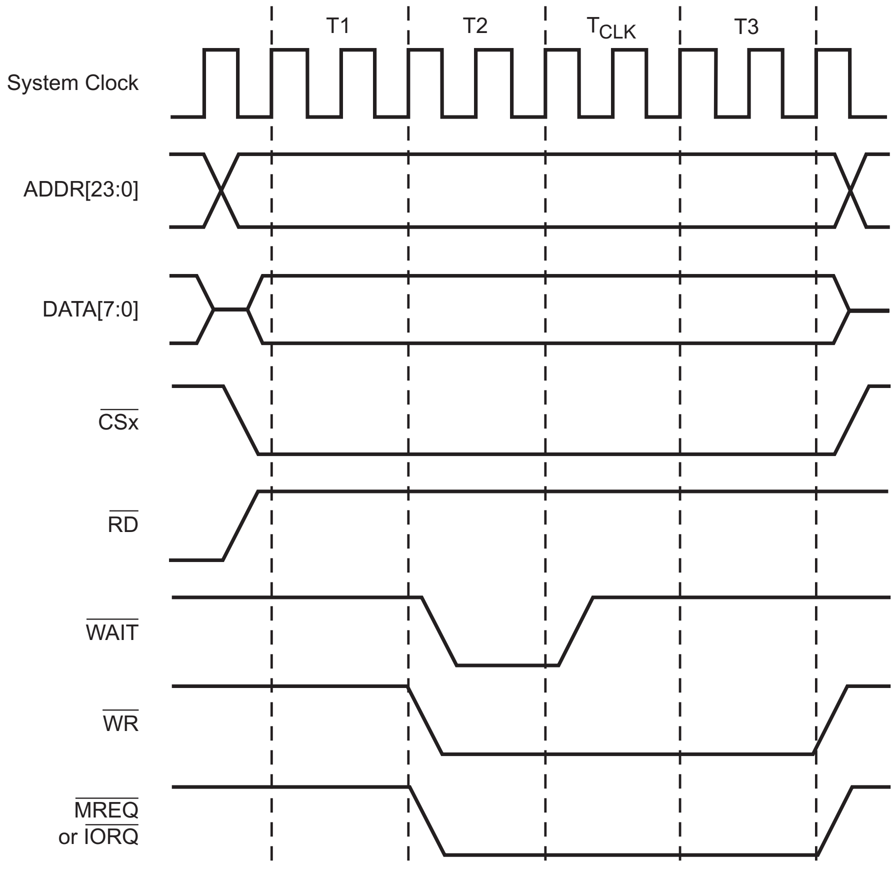


Figure 8. Z80 Bus Mode Write Timing Example

Intel Bus Mode

Chip selects configured for Intel Bus Mode modify the eZ80® bus signals to duplicate a four-state memory transfer similar to that found on Intel-style microprocessors. The bus signals and eZ80L92 pins are mapped as illustrated in Figure 9. In Intel Bus Mode, the user can select either multiplexed or nonmultiplexed address and data buses. In nonmultiplexed operation, the address and data buses are separate. In multiplexed operation, the lower byte of the address, ADDR[7:0], also appears on the data bus, DATA[7:0], during State T1 of the Intel Bus Mode cycle. During multiplexed operation, the lower byte of the address bus also appears on the address bus in addition to the data bus.

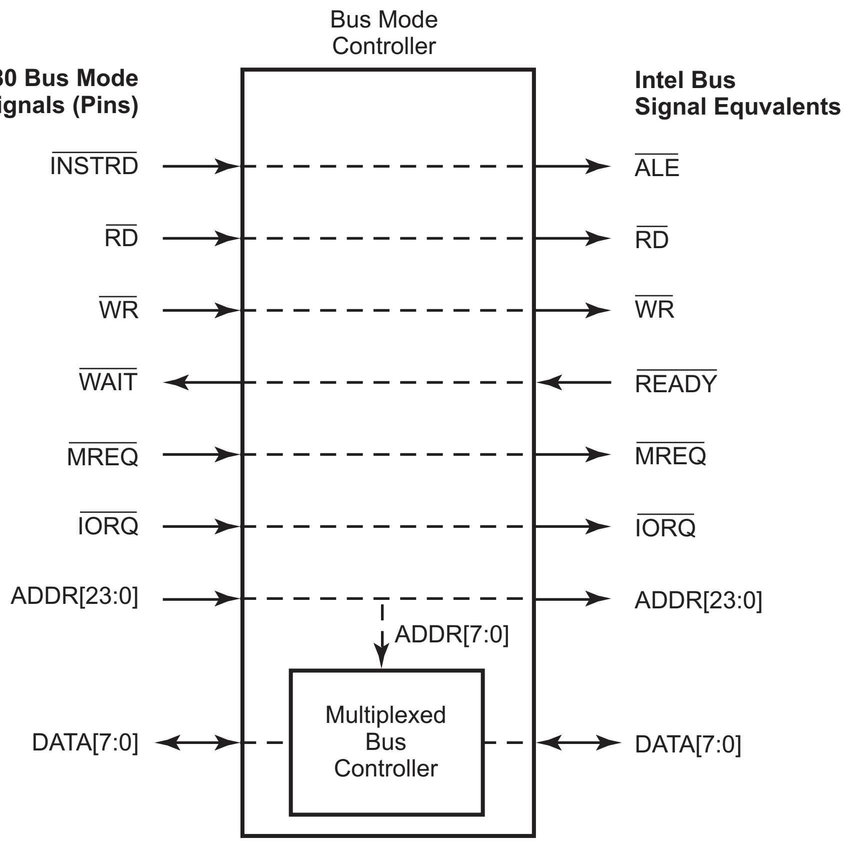


Figure 9. Intel™ Bus Mode Signal and Pin Mapping

Intel Bus Mode (Separate Address and Data Buses)

During Read operations with separate address and data buses, the Intel Bus Mode employs 4 states (T1, T2, T3, and T4) as described in Table 16.

Table 16. Intel™ Bus Mode READ States (Separate Address and Data Buses)

STATE T1	The Read cycle begins in State T1. The CPU drives the address onto the address bus and the associated Chip Select signal is asserted. 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 $\overline{RD}$ signal. Depending on the instruction, either the MREQ or IORQ signal is asserted.

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 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 MREQ or 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 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. Figures 10 and 11 also illustrate the assertion of one WAIT state (T_{WAIT}) by the selected peripheral.

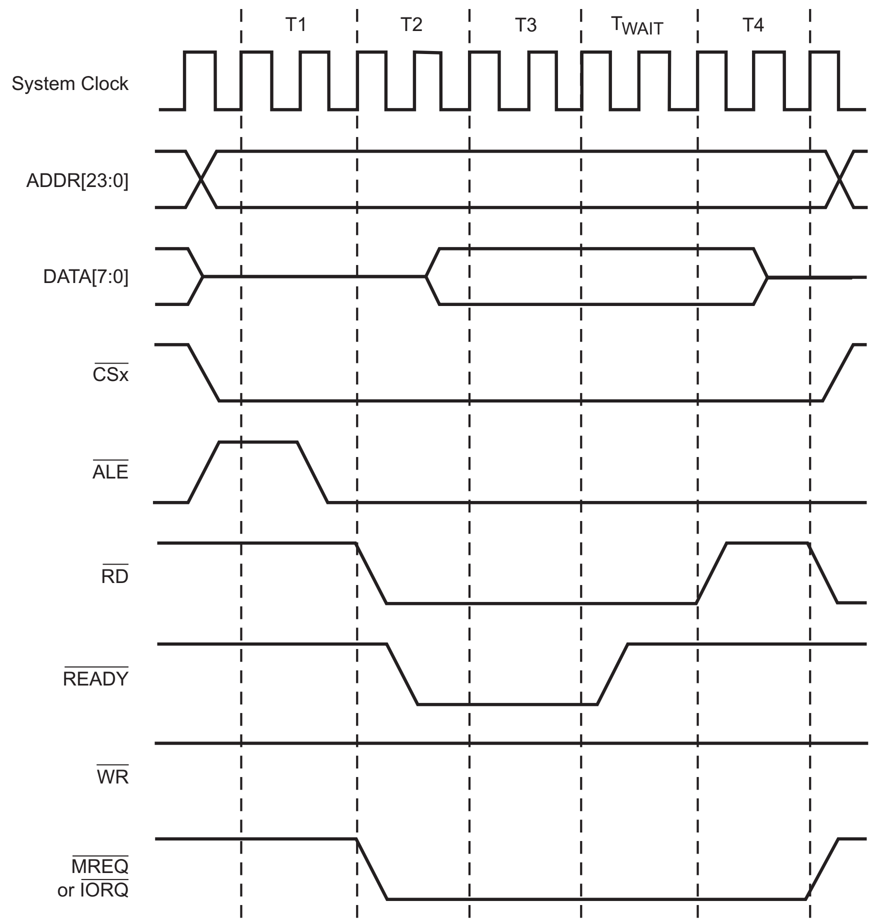


Figure 10. Intel™ Bus Mode Read Timing Example (Separate Address and Data Buses)

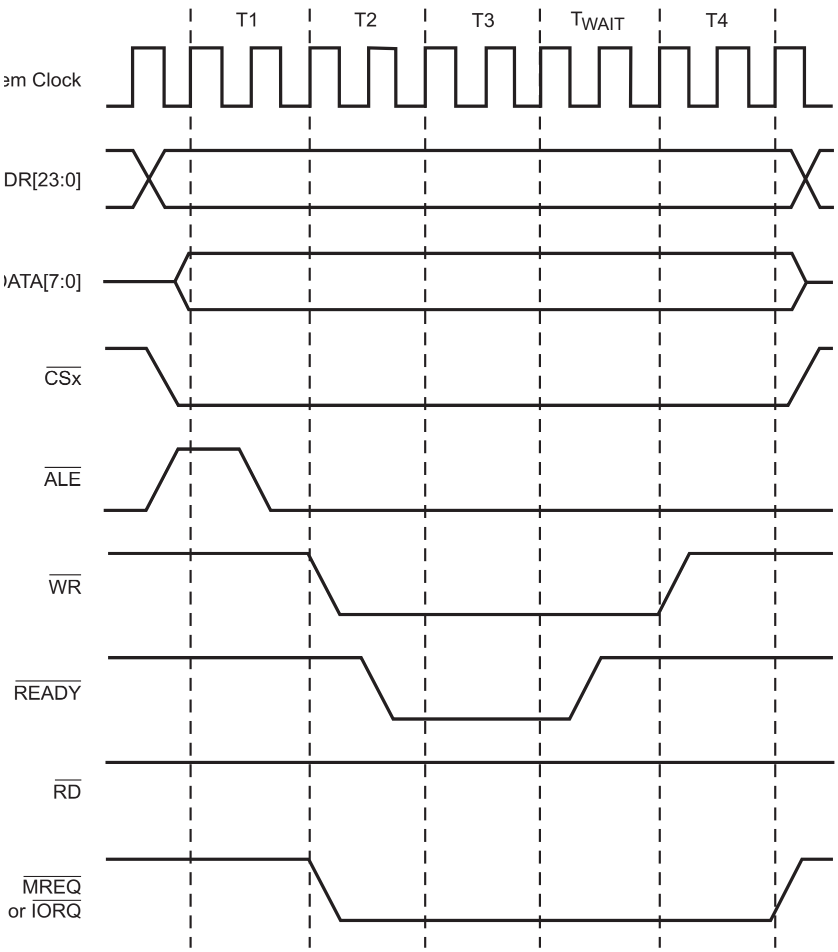


Figure 11. Intel™ Bus Mode Write Timing Example (Separate Address and Data Buses)

Intel™ Bus Mode (Multiplexed Address and Data Bus)

During Read operations with multiplexed address and data, the Intel™ Bus Mode employs 4 states (T1, T2, T3, and T4) as described in Table 18.

Table 18. Intel™ Bus Mode READ States (Multiplexed Address and Data Bus)

STATE T1	The Read cycle begins in State T1. The CPU drives the address onto the DATA bus and the associated Chip Select signal is asserted. 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 removes the address from the DATA bus and asserts the RD signal. Depending upon the instruction, either the MREQ or 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 latches the Read data at the beginning of State T4. The CPU deasserts the RD signal and completes the Intel™ Bus Mode cycle.

During Write operations with multiplexed address and data, the Intel™ Bus Mode employs 4 states (T1, T2, T3, and T4) as described in Table 19.

Table 19. Intel™ Bus Mode WRITE States (Multiplexed Address and Data Bus)

STATE T1	The Write cycle begins in State T1. The CPU drives the address onto the DATA bus and 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 removes the address from the DATA bus and drives the Write data onto the DATA bus. The WR signal is asserted to indicate a Write operation.
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 Write signal at the beginning of T4 identifying the end of the Write operation. The CPU holds the data and address buses through the end of T4. The bus cycle is completed at the end of T4.

Signal timing for Intel™ Bus Mode with multiplexed address and data is illustrated for a Read operation in Figure 12 and for a Write operation in Figure 13. In the figures, each Intel™ Bus Mode state is 2 eZ80® system clock cycles in duration. Figures 12 and 13 also illustrate the assertion of one WAIT state (T_{WAIT}) by the selected peripheral.

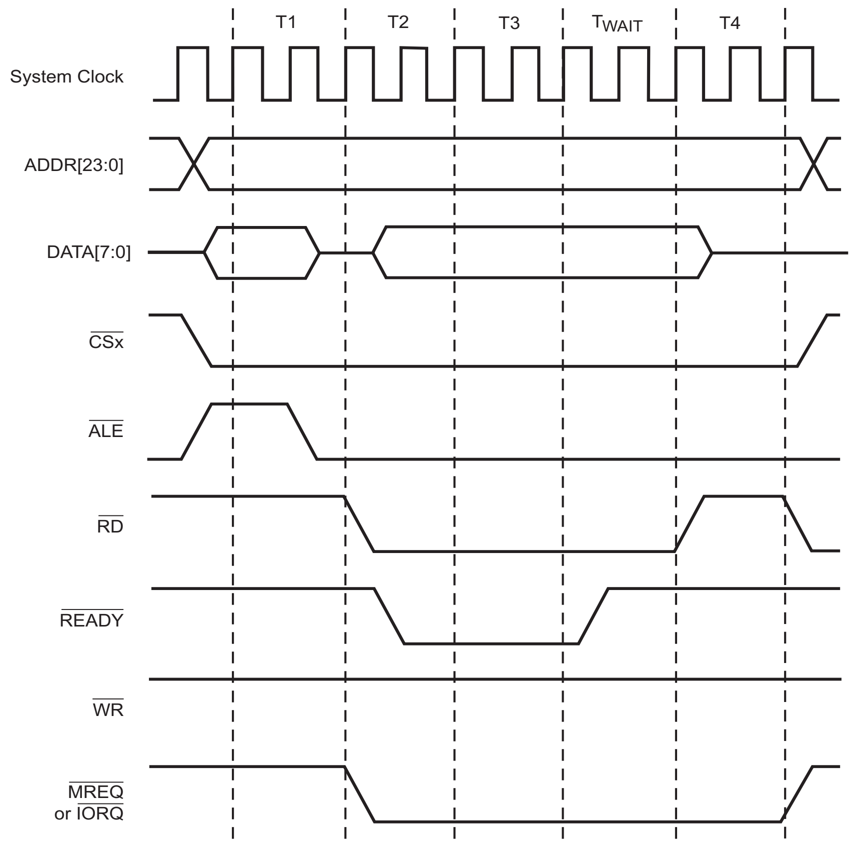


Figure 12. Intel™ Bus Mode Read Timing Example (Multiplexed Address and Data Bus)

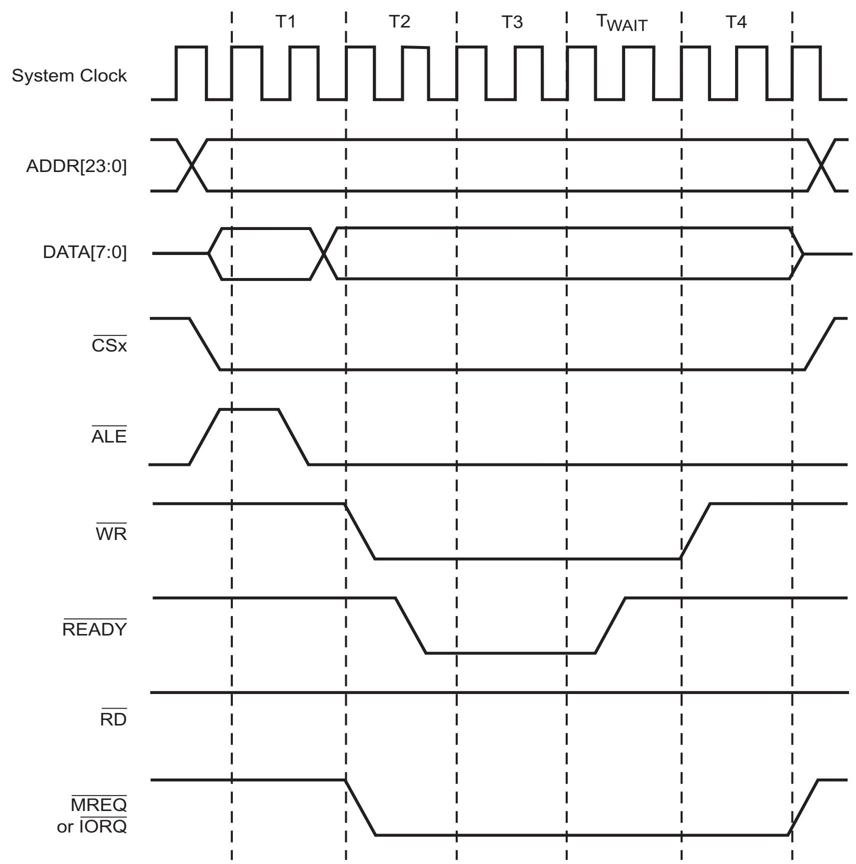


Figure 13. Intel™ Bus Mode Write Timing Example (Multiplexed Address and Data Bus)

Motorola Bus Mode

Chip selects configured for Motorola Bus Mode modify the eZ80® bus signals to duplicate an eight-state memory transfer similar to that found on Motorola-style microprocessors. The bus signals (and eZ80L92 I/O pins) are mapped as illustrated in Figure 14.

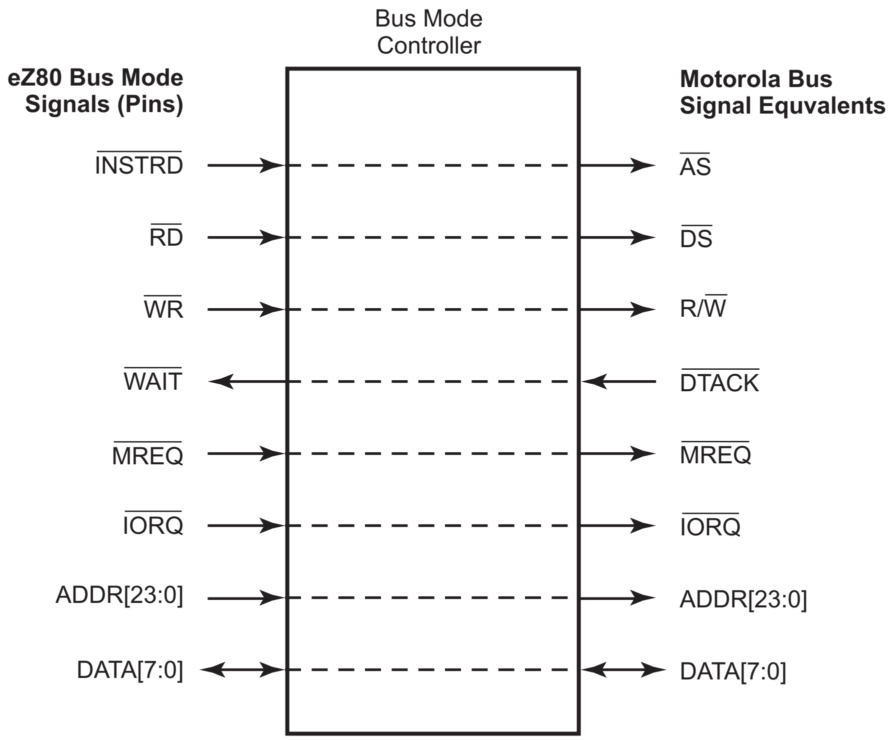


Figure 14. Motorola Bus Mode Signal and Pin Mapping

During Write operations, the Motorola Bus Mode employs 8 states (S0, S1, S2, S3, S4, S5, S6, and S7) as described in Table 20.

Table 20. Motorola Bus Mode Read States

STATE S0	The READ cycle starts in state S0. The CPU drives $R/\overline{W}$ High to identify a READ cycle.
STATE S1	Entering state S1, the CPU drives a valid address on the address bus, ADDR[23:0].
STATE S2	On the rising edge of state S2, the CPU asserts $\overline{AS}$ and $\overline{DS}$.
STATE S3	During state S3, no bus signals are altered.
STATE S4	During state S4, the CPU waits for a cycle termination signal $\overline{DTACK}$ ($\overline{WAIT}$), a peripheral signal. If the termination signal is not asserted at least one full CPU clock period prior to the rising clock edge at the end of S4, the CPU inserts WAIT (T_{WAIT}) states until DTACK is asserted. Each WAIT state is a full bus mode cycle.
STATE S5	During state S5, no bus signals are altered.

Table 20. Motorola Bus Mode Read States (Continued)

STATE S6	During state S6, data from the external peripheral device is driven onto the data bus.
STATE S7	On the rising edge of the clock entering state S7, the CPU latches data from the addressed peripheral device and deasserts AS and DS. The peripheral device deasserts DTACK at this time.

The eight states for a Write operation in Motorola Bus Mode are described in Table 21.

Table 21. Motorola Bus Mode WRITE States

STATE S0	The Write cycle starts in S0. The CPU drives R/$\overline{W}$ High (if a preceding Write cycle leaves R/W Low).
STATE S1	Entering S1, the CPU drives a valid address on the address bus.
STATE S2	On the rising edge of S2, the CPU asserts $\overline{AS}$ and drives R/$\overline{W}$ Low.
STATE S3	During S3, the data bus is driven out of the high-impedance state as the data to be written is placed on the bus.
STATE S4	At the rising edge of S4, the CPU asserts $\overline{DS}$. The CPU waits for a cycle termination signal DTACK (WAIT). If the termination signal is not asserted at least one full CPU clock period prior to the rising clock edge at the end of S4, the CPU inserts WAIT (T_{WAIT}) states until DTACK is asserted. Each WAIT state is a full bus mode cycle.
STATE S5	During S5, no bus signals are altered.
STATE S6	During S6, no bus signals are altered.
STATE S7	Upon entering S7, the CPU deasserts $\overline{AS}$ and $\overline{DS}$. As the clock rises at the end of S7, the CPU drives R/W High. The peripheral device deasserts DTACK at this time.

Signal timing for Motorola Bus Mode is illustrated for a Read operation in Figure 15 and for a Write operation in Figure 16. In these two figures, each Motorola Bus Mode state is 2 eZ80® system clock cycles in duration.

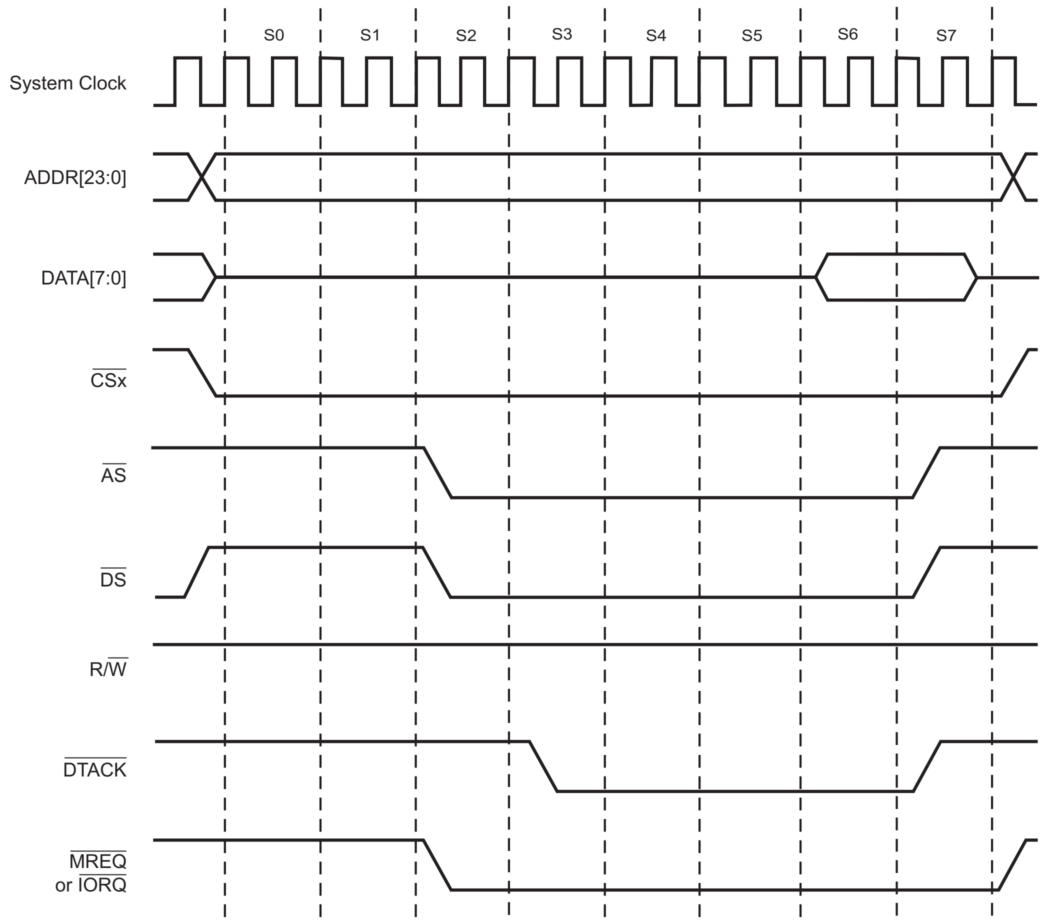


Figure 15. Motorola Bus Mode Read Timing Example

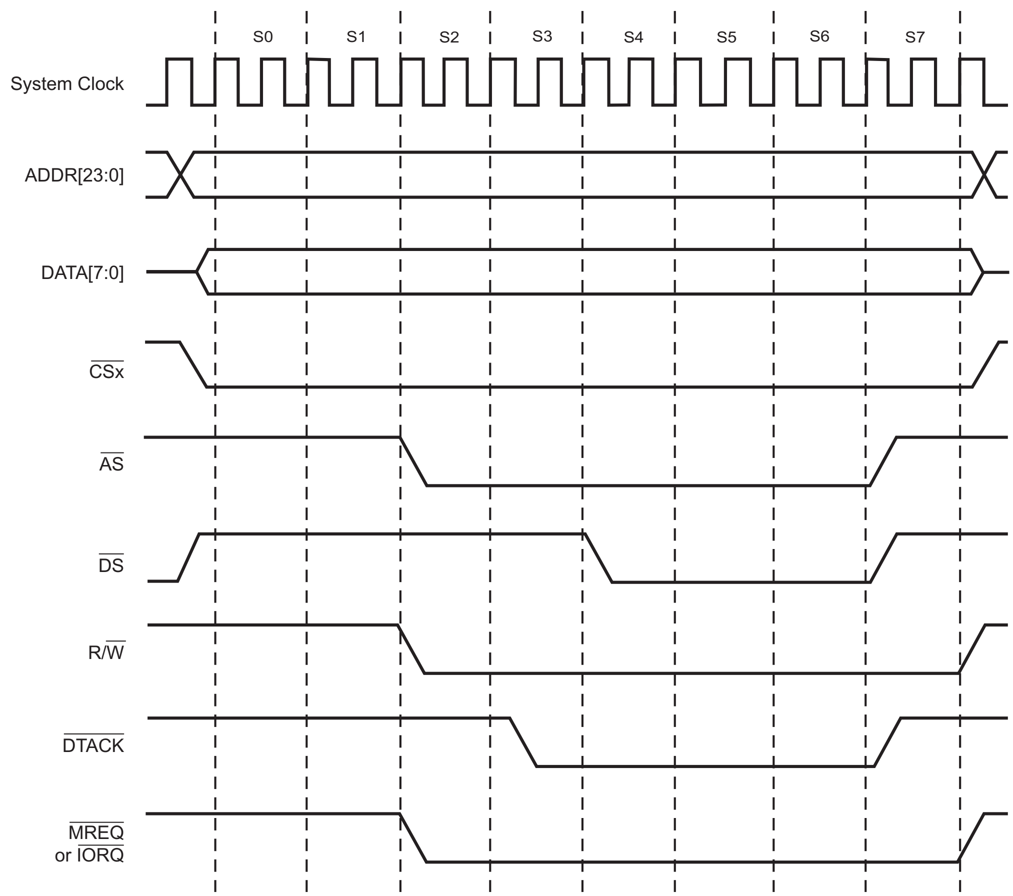


Figure 16. Motorola Bus Mode Write Timing Example

Switching Between Bus Modes

Each time the bus mode controller must switch from one bus mode to another, there is a one-cycle eZ80® system clock delay. An extra clock cycle is not required for repeated accesses in any of the bus modes; nor is it required when the eZ80L92 switches to eZ80 Bus Mode. The extra clock cycles are not shown in the timing examples. Due to the asynchronous nature of these bus protocols, the extra delay does not impact peripheral communication.

Chip Select Registers

Chip Select *x* Lower Bound Registers

For Memory Chip Selects, the Chip Select *x* Lower Bound register, detailed in Table 22, defines the lower bound of the address range for which the corresponding Memory Chip Select (if enabled) can be active. For I/O Chip Selects, this register defines the address to which ADDR[15:8] is compared to generate an I/O Chip Select. All Chip Select lower bound registers reset to `00h`.

Table 22. Chip Select x Lower Bound Registers (CS0_LBR = 00A8h, CS1_LBR = 00ABh, CS2_LBR = 00AEh, CS3_LBR = 00B1h)

Bit	7	6	5	4	3	2	1	0
CS0_LBR Reset	0	0	0	0	0	0	0	0
CS1_LBR Reset	0	0	0	0	0	0	0	0
CS2_LBR Reset	0	0	0	0	0	0	0	0
CS3_LBR 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
[7:0] CSx_LBR	00h–FFh	**For Memory Chip Selects (CSx_IO = 0)** This byte specifies the lower bound of the Chip Select address range. The upper byte of the address bus, ADDR[23:16], is compared to the values contained in these registers for determining whether a Memory Chip Select signal should be generated.
		For I/O Chip Selects (CSx_IO = 1) This byte specifies the Chip Select address value. ADDR[15:8] is compared to the values contained in these registers for determining whether an I/O Chip Select signal should be generated.

Chip Select *x* Upper Bound Registers

For Memory Chip Selects, the Chip Select *x* Upper Bound registers, detailed in Table 23, defines the upper bound of the address range for which the corresponding Chip Select (if enabled) can be active. For I/O Chip Selects, this register produces no effect. The reset state for the Chip Select 0 Upper Bound register is `FFh`, while the reset state for the other Chip Select upper bound registers is `00h`.

Table 23. Chip Select x Upper Bound Registers (CS0_UBR = 00A9h, CS1_UBR = 00ACh, CS2_UBR = 00AFh, CS3_UBR = 00B2h)

Bit	7	6	5	4	3	2	1	0
CS0_UBR Reset	1	1	1	1	1	1	1	1
CS1_UBR Reset	0	0	0	0	0	0	0	0
CS2_UBR Reset	0	0	0	0	0	0	0	0
CS3_UBR 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
[7:0] CSx_UBR	00h–FFh	**For Memory Chip Selects (CSx_IO = 0)** This byte specifies the upper bound of the Chip Select address range. The upper byte of the address bus, ADDR[23:16], is compared to the values contained in these registers for determining whether a Chip Select signal should be generated.
		For I/O Chip Selects (CSx_IO = 1) No effect.

Chip Select *x* Control Registers

The Chip Select *x* Control register, detailed in Table 24, enables the Chip Selects, specifies the type of Chip Select, and sets the number of WAIT states. The reset state for the Chip Select 0 Control register is `E8h`, while the reset state for the 3 other Chip Select control registers is `00h`.

Table 24. Chip Select x Control Registers (CS0_CTL = 00AAh, CS1_CTL = 00ADh, CS2_CTL = 00B0h, CS3_CTL = 00B3h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CS0_CTL Reset	1	1	1	0	1	0	0	0
CS1_CTL Reset	0	0	0	0	0	0	0	0
CS2_CTL Reset	0	0	0	0	0	0	0	0
CS3_CTL Reset	0	0	0	0	0	0	0	0
CPU Access	R/W	R/W	R/W	R/W	R/W	R	R	R

Note: R/W = Read/Write; R = Read Only.

Bit Position	**Value**	**Description**
[7:5] CSx_WAIT*	000	0 WAIT states are asserted when this Chip Select is active.
	001	1 WAIT state is asserted when this Chip Select is active.
	010	2 WAIT states are asserted when this Chip Select is active.
	011	3 WAIT states are asserted when this Chip Select is active.
	100	4 WAIT states are asserted when this Chip Select is active.
	101	5 WAIT states are asserted when this Chip Select is active.
	110	6 WAIT states are asserted when this Chip Select is active.
	111	7 WAIT states are asserted when this Chip Select is active.
4 CSx_IO	0	Chip Select is configured as a Memory Chip Select.
	1	Chip Select is configured as an I/O Chip Select.
3 CSx_EN	0	Chip Select is disabled.
	1	Chip Select is enabled.
[2:0]	000	Reserved.

Note: *These WAIT state settings apply only to the default eZ80 bus mode. See Table 25.

Chip Select *x* Bus Mode Control Registers

The Chip Select Bus Mode register, detailed in Table 25, configures the Chip Select for eZ80®, Z80, Intel™, or Motorola Bus Modes. Changing the bus mode allows the eZ80L92 to interface to peripherals based on the Z80-, Intel™-, or Motorola-style asynchronous bus interfaces. When a bus mode other than eZ80® is programmed for a particular Chip Select, the CSx_WAIT setting in that Chip Select Control Register is ignored.

Table 25. Chip Select *x* Bus Mode Control Registers (CS0_BMC = 00F0h, CS1_BMC = 00F1h, CS2_BMC = 00F2h, CS3_BMC = 00F3h)

Bit	7	6	5	4	3	2	1	0
CS0_BMC Reset	0	0	0	0	0	0	1	0
CS1_BMC Reset	0	0	0	0	0	0	1	0
CS2_BMC Reset	0	0	0	0	0	0	1	0
CS3_BMC Reset	0	0	0	0	0	0	1	0
CPU Access	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W

Note: R/W = Read/Write; R = Read Only.

Bit Position	Value	Description
[7:6] BUS_MODE	00	eZ80® bus mode.
	01	Z80 bus mode.
	10	Intel™ bus mode.
	11	Motorola bus mode.
5 AD_MUX	0	Separate address and data.
	1	Multiplexed address and data—appears on data bus DATA[7:0].
4	0	Reserved.

Bit Position	Value	Description
[3:0] BUS_CYCLE	0000	Not valid.
	0001	Each bus mode state is 1 eZ80® clock cycle in duration.[1, 2, 3]
	0010	Each bus mode state is 2 eZ80® clock cycles in duration.
	0011	Each bus mode state is 3 eZ80® clock cycles in duration.
	0100	Each bus mode state is 4 eZ80® clock cycles in duration.
	0101	Each bus mode state is 5 eZ80® clock cycles in duration.
	0110	Each bus mode state is 6 eZ80® clock cycles in duration.
	0111	Each bus mode state is 7 eZ80® clock cycles in duration.
	1000	Each bus mode state is 8 eZ80® clock cycles in duration.
	1001	Each bus mode state is 9 eZ80® clock cycles in duration.
	1010	Each bus mode state is 10 eZ80® clock cycles in duration.
	1011	Each bus mode state is 11 eZ80® clock cycles in duration.
	1100	Each bus mode state is 12 eZ80® clock cycles in duration.
	1101	Each bus mode state is 13 eZ80® clock cycles in duration.
	1110	Each bus mode state is 14 eZ80® clock cycles in duration.
	1111	Each bus mode state is 15 eZ80® clock cycles in duration.

Notes:
1. Setting the BUS_CYCLE to 1 in Intel Bus Mode causes the ALE pin to not function properly.
2. Use of the external WAIT input pin in Z80 Mode requires that BUS_CYCLE is set to a value greater than 1.
3. These BUS_CYCLE values are not valid in eZ80 bus mode. See Table 24.

Watch-Dog Timer

Watch-Dog Timer Overview

The Watch-Dog 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 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 nonmaskable interrupt (NMI)
- A WDT time-out RESET indicator flag

Figure 17 illustrates the block diagram for the Watch-Dog Timer.

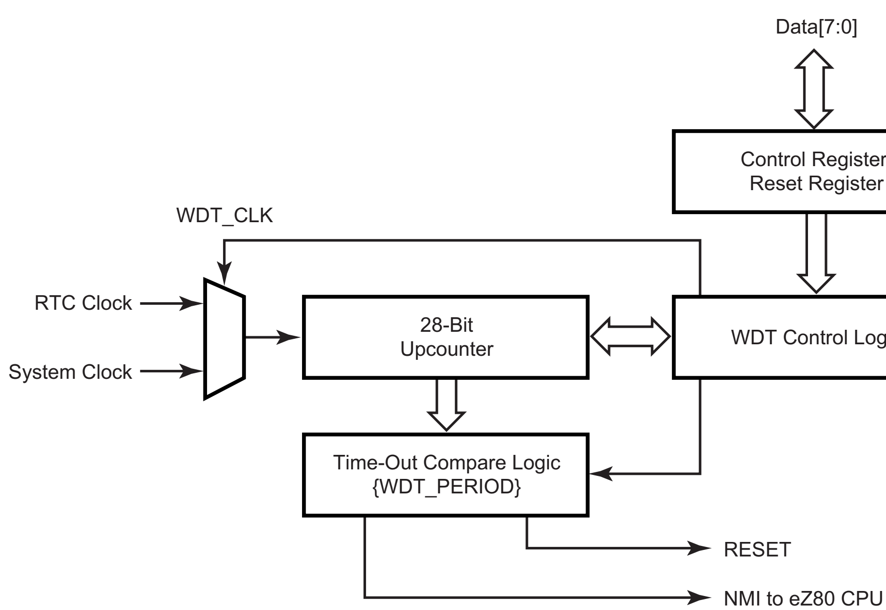


Figure 17. Watch-Dog Timer Block Diagram

Watch-Dog Timer Operation

Enabling and Disabling the WDT

The Watch-Dog Timer is disabled upon a system reset (RESET). To enable the WDT, the application program must set the WDT_EN bit (bit 7) of the WDT_CTL register. When enabled, the WDT cannot be disabled without a RESET.

Time-Out Period Selection

There are four choices of time-out periods for the WDT—2^{18}, 2^{22}, 2^{25}, and 2^{27} system clock cycles. The WDT time-out period is defined by the WDT_PERIOD field of the WDT_CTL register (WDT_CTL[1:0]). The approximate time-out periods for two different WDT clock sources is listed in Table 26.

Table 26. Watch-Dog Timer Approximate Time-Out Delays

Clock Source	Divider Value	Time Out Delay
32.768KHz Crystal Oscillator	2^{18}	8.00s
32.768KHz Crystal Oscillator	2^{22}	128s
32.768KHz Crystal Oscillator	2^{25}	1024s
32.768KHz Crystal Oscillator	2^{27}	4096s
20MHz System Clock	2^{18}	13.1ms
20MHz System Clock	2^{22}	209.7ms
20MHz System Clock	2^{25}	1.68s
20MHz System Clock	2^{27}	6.71s
50MHz System Clock	2^{18}	5.2ms*
50MHz System Clock	2^{22}	83.9ms*
50MHz System Clock	2^{25}	0.67s
50MHz System Clock	2^{27}	2.68s

RESET Or NMI Generation

Upon a WDT time-out, the RST_FLAG in the WDT_CTL register is set to 1. In addition, the WDT can cause a RESET or send a nonmaskable interrupt (NMI) signal to the CPU. The default operation is for the WDT to cause a RESET. It asserts/deasserts on the rising edge of the clock. The RST_FLAG bit can be polled by the CPU to determine the source of the RESET event.

If the NMI_OUT bit in the WDT_CTL register is set to 1, then upon time-out, the WDT asserts an NMI for CPU processing. The RST_FLAG bit can be polled by the CPU to determine the source of the NMI event, provided that the last RESET was not caused by the WDT.

Watch-Dog Timer Registers

Watch-Dog Timer Control Register

The Watch-Dog Timer Control register, detailed in Table 27, is an 8-bit Read/Write register used to enable the Watch-Dog Timer, set the time-out period, indicate the source of the most recent RESET, and select the required operation upon WDT time-out.

Table 27. Watch-Dog Timer Control Register (WDT_CTL = 0093h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0/1	0	0	0	0	0
CPU Access	R/W	R/W	R	R/W	R/W	R	R/W	R/W

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

Bit Position	**Value**	**Description**
7 WDT_EN	0	WDT is disabled.
	1	WDT is enabled. When enabled, the WDT cannot be disabled without a full RESET.
6 NMI_OUT	0	WDT time-out resets the CPU.
	1	WDT time-out generates a nonmaskable interrupt (NMI) to the CPU.
5 RST_FLAG*	0	RESET caused by external full-chip reset or ZDI reset.
	1	RESET caused by WDT time-out. This flag is set by the WDT time-out, even if the NMI_OUT flag is set to 1. The CPU can poll this bit to determine the source of the RESET or NMI.
[4:3] WDT_CLK	00	WDT clock source is system clock.
	01	WDT clock source is Real-Time Clock source (32KHz on-chip oscillator or 50/60Hz input as set by RTC_CTRL[4]) .
	10	Reserved.
	11	Reserved.
2 RESERVED	0	Reserved.

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

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.

Watch-Dog Timer Reset Register

The Watch-Dog Timer Reset register, detailed in Table 28, is an 8-bit Write-Only register. The Watch-Dog 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. Watch-Dog 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

Programmable Reload Timers Overview

The eZ80L92 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.

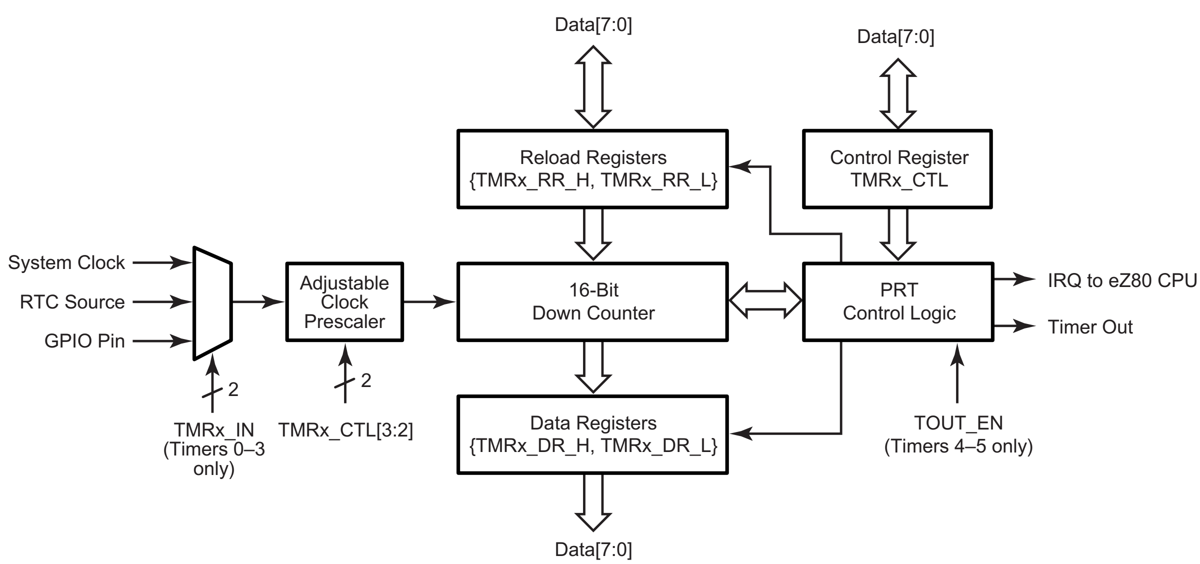


Figure 18. Programmable Reload Timer Block Diagram

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 of the timer is achieved by loading `0001h`. Maximum duration is achieved by loading `0000h`, because the timer first rolls over to `FFFFh` and then continues counting down to `0000h`.

The time-out period of the PRT is returned by the following equation:

$$\text{PRT Time-Out Period} = \frac{\text{Clock Divider Ratio x Reload Value}}{\text{System Clock Frequency}}$$

To calculate the time-out period with the above equation when using an initial value of `0000h`, enter a reload value of 65536 (`FFFFh` + 1).

Minimum time-out duration is 4 times longer than the input clock period and is generated by setting the clock divider ratio to 1:4 and the reload value to `0001h`. Maximum time-out duration is 2^{24} (16,777,216) times longer than the input clock period and is generated by setting the clock divider ratio to 1:256 and the reload value to `0000h`.

SINGLE PASS Mode

In SINGLE PASS mode, when the end-of-count value, `0000h`, is reached, counting halts, the timer is disabled, and the PRT_EN bit resets to 0. To restart the timer, the CPU must reenable the timer by setting the PRT_EN bit to 1. An example of a PRT operating in SINGLE PASS mode is illustrated in Figure 19. Timer register information is indicated in Table 29.

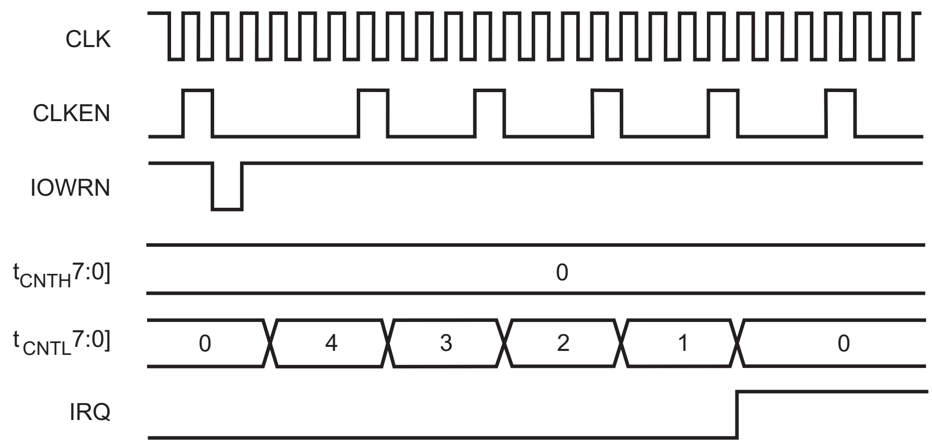


Figure 19. PRT Single Pass Mode Operation Example

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.

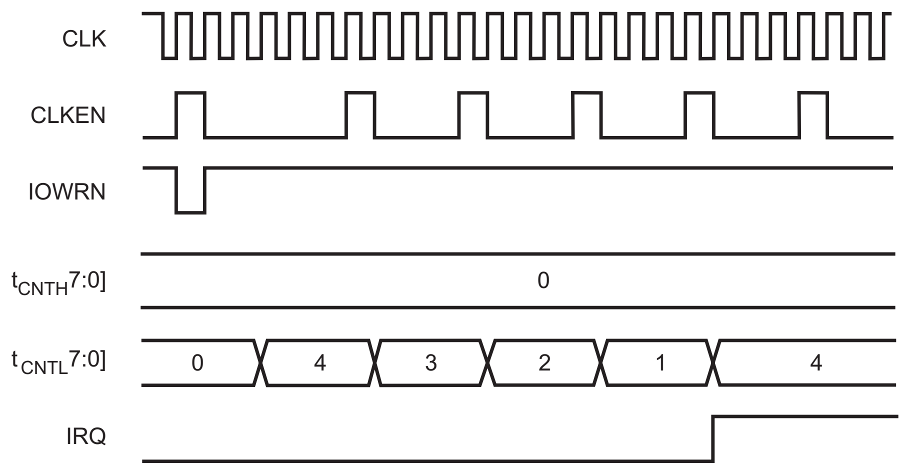


Figure 20. PRT CONTINUOUS Mode Operation Example

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 32768Hz) 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.

Event Counter

When Timers 0–3 are configured to take their inputs from port input pins PB0 and PB1, they function as event counters. For event counting, the clock prescaler is bypassed. The PRT counters decrement on every rising edge of the port pin. The port pins must be configured as inputs. Due to the input sampling on the pins, the event input signal frequency is limited to one-half the system clock frequency. Input sampling on the port pins results in the PRT counter being updated on the fifth rising edge of the system clock after the rising edge occurs at the port pin.

Timer Output

Two of the Programmable Reload Timers (Timers 4 and 5) can be directed to GPIO Port B output pins (PB4 and PB5, respectively). To enable the Timer Out feature, the GPIO port pin must be configured for alternate functions. After reset, the Timer Output feature is disabled by default. The GPIO output pin toggles each time the PRT reaches its end-of-count value. In CONTINUOUS mode operation, the disabling of the Timer Output feature results in a Timer Output signal period that is twice the PRT time-out period. Examples of the Timer Output operation are illustrated in Figure 21 and Table 31. In these examples, the GPIO output is assumed to be Low (0) when the Timer Output function is enabled.

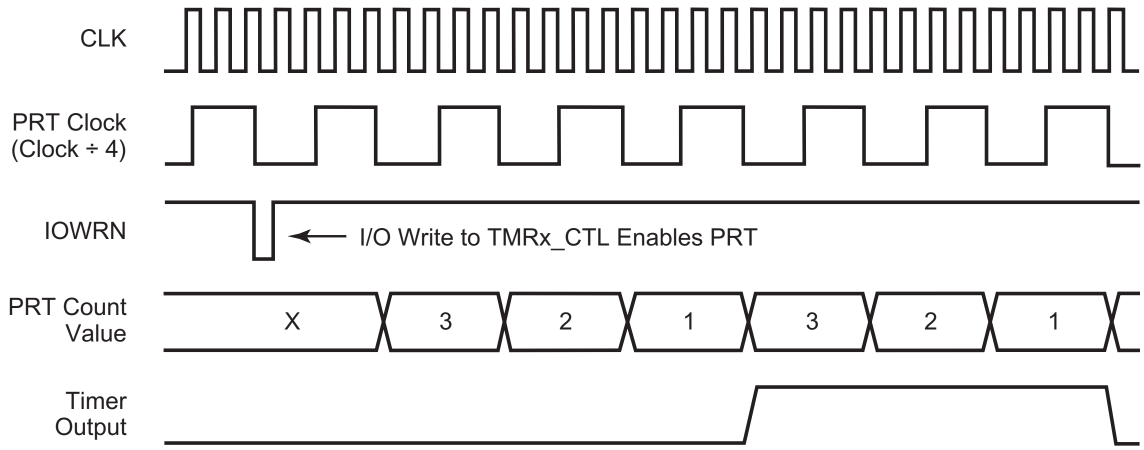


Figure 21. PRT Timer Output Operation Example

Table 31. PRT Timer Out 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 Reload Value	{TMRx_RR_H, TMRx_RR_L}	0003h

Programmable Reload Timer Registers

Each programmable reload timer is controlled using five 8-bit registers. These registers are the Timer Control register, Timer Reload Low Byte register, Timer Reload High Byte register, Timer Data Low Byte register, and Timer Data High Byte register.

The Timer Control register can be read or written to. The timer reload registers are Write-Only and are located at the same I/O address as the timer data registers, which are Read-Only.

Timer Control Registers

The Timer Control register, detailed in Table 32, is used to control operation of the timer, including enabling the timer, selecting the clock divider, enabling the interrupt, selecting between CONTINUOUS and SINGLE PASS modes, and enabling the auto-reload feature.

Table 32. Timer Control Registers (TMR0_CTL = 0080h, TMR1_CTL = 0083h, TMR2_CTL = 0086h, TMR3_CTL = 0089h, TMR4_CTL = 008Ch, or TMR5_CTL = 008Fh)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
CPU Access	R	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 PRT_IRQ	0	The timer does not reach its end-of-count value. This bit is reset to 0 every time the TMRx_CTL register is read.
	1	The timer reaches its end-of-count value. If IRQ_EN is set to 1, an interrupt signal is sent to the CPU. This bit remains 1 until the TMRx_CTL register is read.
6 IRQ_EN	0	Timer interrupt requests are disabled.
	1	Timer interrupt requests are enabled.
5	0	Reserved.
4 PRT_MODE	0	The timer operates in SINGLE PASS mode. PRT_EN (bit 0) is reset to 0, and counting stops when the end-of-count value is reached.
	1	The timer operates in CONTINUOUS mode. The timer reload value is written to the counter when the end-of-count value is reached.
[3:2] CLK_DIV	00	Clock ÷ 4 is the timer input source.
	01	Clock ÷ 16 is the timer input source.
	10	Clock ÷ 64 is the timer input source.
	11	Clock ÷ 256 is the timer input source.
1 RST_EN	0	The automatic reload and restart function is disabled.
	1	The automatic reload and restart function is enabled. When a 1 is written to RST_EN, the values in the reload registers are loaded into the downcounter and the timer restarts.
0 PRT_EN	0	The programmable reload timer is disabled.
	1	The programmable reload timer is enabled.

Timer Data Registers—Low Byte

This Read-Only register returns the Low byte of the current count value of the selected timer. The Timer Data Register—Low Byte, detailed in Table 33, can be read while the timer is in operation. Reading the current count value does not affect timer operation. To read the 16-bit data of the current count value, {TMRx_DR_H[7:0], TMRx_DR_L[7:0]}, first read the Timer Data Register—Low Byte and then read the Timer Data Register—High Byte. The Timer Data Register—High Byte value is latched when a Read of the Timer Data Register—Low Byte occurs.

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

Table 33. Timer Data Registers—Low Byte (TMR0_DR_L = 0081h, TMR1_DR_L = 0084h, TMR2_DR_L = 0087h, TMR3_DR_L = 008Ah, TMR4_DR_L = 008Dh, or TMR5_DR_L = 0090h)

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.

Bit Position	Value	Description
[7:0] TMRx_DR_L	00h–FFh	These bits represent the Low byte of the 2-byte timer data value, {TMRx_DR_H[7:0], TMRx_DR_L[7:0]}. Bit 7 is bit 7 of the 16-bit timer data value. Bit 0 is bit 0 (lsb) of the 16-bit timer data value.

Timer Data Registers—High Byte

This Read-Only register returns the High byte of the current count value of the selected timer. The Timer Data Register—High Byte, detailed in Table 34, can be read while the timer is in operation. Reading the current count value does not affect timer operation. To read the 16-bit data of the current count value, {TMRx_DR_H[7:0], TMRx_DR_L[7:0]}, first read the Timer Data Register—Low Byte and then read the Timer Data Register—High Byte. The Timer Data Register—High Byte value is latched when a Read of the Timer Data Register—Low Byte occurs.

Note: The timer data registers and timer reload registers share the same address space.

Table 34. Timer Data Registers—High Byte (TMR0_DR_H = 0082h, TMR1_DR_H = 0085h, TMR2_DR_H = 0088h, TMR3_DR_H = 008Bh, TMR4_DR_H = 008Eh, or TMR5_DR_H = 0091h)

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.

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

Timer Reload Registers—Low Byte

The Timer Reload Register—Low Byte, detailed in Table 35, stores the least significant byte (LSB) 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 35. Timer Reload Registers—Low Byte (TMR0_RR_L = 0081h, TMR1_RR_L = 0084h, TMR2_RR_L = 0087h, TMR3_RR_L = 008Ah, TMR4_RR_L = 008Dh, or TMR5_RR_L = 0090h)

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_L	00h–FFh	These bits represent the Low byte of the 2-byte timer reload value, {TMRx_RR_H[7:0], TMRx_RR_L[7:0]}. Bit 7 is bit 7 of the 16-bit timer reload value. Bit 0 is bit 0 (lsb) of the 16-bit timer reload value.

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.

Table 37. Timer Input Source Select Register (TMR_ISS = 0092h)

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
[7:6] TMR3_IN	00	The timer counts at the system clock divided by the prescaler.
	01	The timer event input is the Real-Time Clock source (32KHz or 50/60Hz—refer to the Real-Time Clock section on page 86 for details).
	10	The timer event input is the GPIO Port B pin 1.
	11	The timer event input is the GPIO Port B pin 1.
[5:4] TMR2_IN	00	The timer counts at the system clock divided by the prescaler.
	01	The timer event input is the Real-Time Clock source (32KHz or 50/60Hz—refer to the Real-Time Clock section on page 86 for details).
	10	The timer event input is the GPIO Port B pin 0.
	11	The timer event input is the GPIO Port B pin 0.
[3:2] TMR1_IN	00	The timer counts at the system clock divided by the prescaler.
	01	The timer event input is the Real-Time Clock source (32KHz or 50/60Hz—refer to the Real-Time Clock section on page 86 for details).
	10	The timer event input is the GPIO Port B pin 1.
	11	The timer event input is the GPIO Port B pin 1.
[1:0] TMR0_IN	00	Timer counts at system clock divided by prescaler.
	01	Timer event input is Real-Time Clock source (32KHz or 50/60Hz—refer to the Real-Time Clock section on page 86 for details).
	10	The timer event input is the GPIO Port B pin 0.
	11	The timer event input is the GPIO Port B pin 0.

Real-Time Clock

Real-Time Clock Overview

The Real-Time Clock (RTC) keeps time by maintaining a count of seconds, minutes, hours, day-of-the-week, day-of-the-month, year, and century. The current time is kept in 24-hour format. The format for all count and alarm registers is selectable between binary and binary-coded-decimal (BCD). The calendar operation maintains the correct day of the month and automatically compensates for leap year. A simplified block diagram of the RTC and the associated on-chip, low-power, 32KHz oscillator is illustrated in Figure 22. Connections to an external battery supply and 32KHz crystal network are also demonstrated in Figure 22.

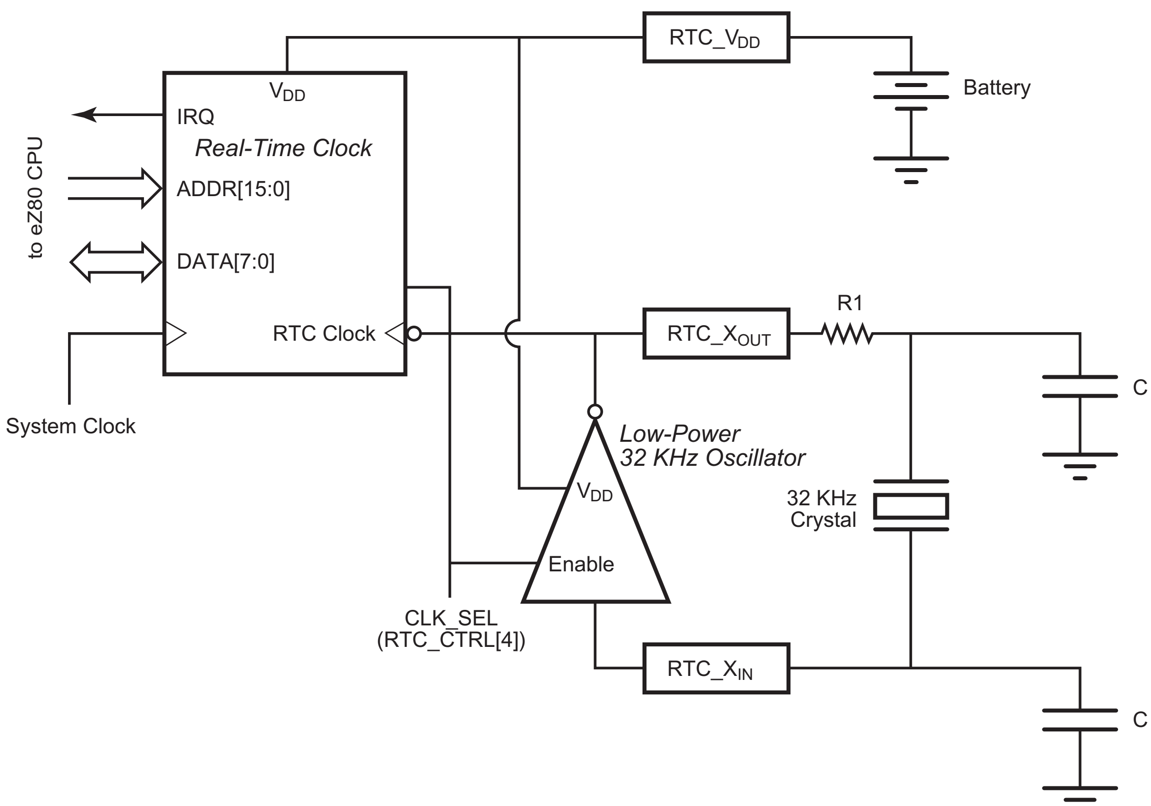


Figure 22. Real-Time Clock and 32KHz Oscillator Block Diagram

Real-Time Clock Alarm

The clock can be programmed to generate an alarm condition when the current count matches the alarm set-point registers. Alarm registers are available for seconds, minutes, hours, and day-of-the-week. Each alarm can be independently enabled. To generate an alarm condition, the current time must match ***all*** enabled alarm values. For example, if the day-of-the-week and hour alarms are both enabled, the alarm only occurs at the specified hour on the specified day. The alarm triggers an interrupt if the interrupt enable bit, INT_EN, is set. The alarm flag, ALARM, and corresponding interrupt to the CPU are cleared by reading the RTC_CTRL register.

Alarm value registers and alarm control registers can be written at any time. Alarm conditions are generated when the count value matches the alarm value. The comparison of alarm and count values occurs whenever the RTC count increments (one time every second). The RTC can also be forced to perform a comparison at any time by writing a 0 to RTC_UNLOCK (RTC_UNLOCK is not required to be changed to a 1 first).

Real-Time Clock Oscillator and Source Selection

The RTC count is driven by either the on-chip 32768 Hz crystal oscillator or a 50/60 Hz power-line frequency input connected to the 32 KHz RTC_X_{OUT} pin. An internal divider compensates for each of these options. The clock source and power-line frequencies are selected in the RTC_CTRL register. Writing to the RTC_CTRL register resets the clock divider.

Real-Time Clock Battery Backup

The power supply pin (RTC_V_{DD}) for the Real-Time Clock and associated low-power 32 KHz oscillator is isolated from the other power supply pins on the eZ80L92. To ensure that the RTC continues to keep time in the event of loss of line power to the application, a battery can be used to supply power to the RTC and the oscillator via the RTC_V_{DD} pin. All V_{SS} (ground) pins should be connected together on the printed circuit assembly.

Real-Time Clock Recommended Operation

Following a RESET from a powered-down condition, the counter values of the RTC are undefined and all alarms are disabled. After a RESET from a powered-down condition, the following procedure is recommended:

- Write to RTC_CTRL to set RTC_UNLOCK and CLK_SEL
- Write values to the RTC count registers to set the current time

- Write values to the RTC alarm registers to set the appropriate alarm conditions
- Write to RTC_CTRL to clear RTC_UNLOCK; clearing the RTC_UNLOCK bit resets and enables the clock divider

Real-Time Clock Registers

The real-time clock registers are accessed via the address and data bus using I/O instructions. RTC_UNLOCK controls access to the RTC count registers. When unlocked (RTC_UNLOCK = 1), the RTC count is disabled and the count registers are Read/Write. When locked (RTC_UNLOCK = 0), the RTC count is enabled and the count registers are Read-Only. The default, at RESET, is for the RTC to be locked.

Real-Time Clock Seconds Register

This register contains the current seconds count. The value in the RTC_SEC register is unchanged by a RESET. The current setting of BCD_EN determines whether the values in this register are binary (BCD_EN = 0) or binary-coded decimal (BCD_EN = 1). Access to this register is Read-Only if the RTC is locked and Read/Write if the RTC is unlocked. See Table 38.

Table 38. Real-Time Clock Seconds Register (RTC_SEC = 00E0h)

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

Note: X = Unchanged by RESET; R/W* = Read-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] TEN_SEC	0–5	The tens digit of the current seconds count.
[3:0] SEC	0–9	The ones digit of the current seconds count.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] SEC	00h–3Bh	The current seconds count.

Real-Time Clock Minutes Register

This register contains the current minutes count. See Table 39.

Table 39. Real-Time Clock Minutes Register (RTC_MIN = 00E1h)

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

Note: X = Unchanged by RESET; R/W* = Read-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] TEN_MIN	0–5	The tens digit of the current minutes count.
[3:0] MIN	0–9	The ones digit of the current minutes count.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] MIN	00h–3Bh	The current minutes count.

Real-Time Clock Hours Register

This register contains the current hours count. See Table 40.

Table 40. Real-Time Clock Hours Register (RTC_HRS = 00E2h)

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

Note: X = Unchanged by RESET; R/W* = Read-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] TEN_HRS	0–2	The tens digit of the current hours count.
[3:0] HRS	0–9	The ones digit of the current hours count.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] HRS	00h–17h	The current hours count.

Real-Time Clock Day-of-the-Week Register

This register contains the current day-of-the-week count. The RTC_DOW register begins counting at `01h`. See Table 41.

Table 41. Real-Time Clock Day-of-the-Week Register (RTC_DOW = 00E3h)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	X	X	X	X
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-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4]	0000	Reserved.
[3:0] DOW	1-7	The current day-of-the-week.count.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:4]	0000	Reserved.
[3:0] DOW	01h–07h	The current day-of-the-week count.

Real-Time Clock Day-of-the-Month Register

This register contains the current day-of-the-month count. The RTC_DOM register begins counting at `01h`. See Table 42.

Table 42. Real-Time Clock Day-of-the-Month Register (RTC_DOM = 00E4h)

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

Note: X = Unchanged by RESET; R/W* = Read-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] TENS_DOM	0–3	The tens digit of the current day-of-the-month count.
[3:0] DOM	0–9	The ones digit of the current day-of-the-month count.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] DOM	01h–1Fh	The current day-of-the-month count.

Real-Time Clock Month Register

This register contains the current month count. See Table 43.

Table 43. Real-Time Clock Month Register (RTC_MON = 00E5h)

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

Note: X = Unchanged by RESET; R/W* = Read-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] TENS_MON	0–1	The tens digit of the current month count.
[3:0] MON	0–9	The ones digit of the current month count.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] MON	01h–0Ch	The current month count.

Real-Time Clock Year Register

This register contains the current year count. See Table 44.

Table 44. Real-Time Clock Year Register (RTC_YR = 00E6h)

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

Note: X = Unchanged by RESET; R/W* = Read-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] TENS_YR	0–9	The tens digit of the current year count.
[3:0] YR	0–9	The ones digit of the current year count.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] YR	00h–63h	The current year count.

Real-Time Clock Century Register

This register contains the current century count. See Table 45.

Table 45. Real-Time Clock Century Register (RTC_CEN = 00E7h)

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

Note: X = Unchanged by RESET; R/W* = Read-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] TENS_CEN	0–9	The tens digit of the current century count.
[3:0] CEN	0–9	The ones digit of the current century count.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] CEN	00h–63h	The current century count.

Real-Time Clock Alarm Seconds Register

This register contains the alarm seconds value. See Table 46.

Table 46. Real-Time Clock Alarm Seconds Register (RTC_ASEC = 00E8h)

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

Note: X = Unchanged by RESET; R/W = Read/Write.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] ATEN_SEC	0–5	The tens digit of the alarm seconds value.
[3:0] ASEC	0–9	The ones digit of the alarm seconds value.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] ASEC	00h–3Bh	The alarm seconds value.

Real-Time Clock Alarm Minutes Register

This register contains the alarm minutes value. See Table 47.

Table 47. Real-Time Clock Alarm Minutes Register (RTC_AMIN = 00E9h)

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

Note: X = Unchanged by RESET; R/W = Read/Write.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] ATEN_MIN	0–5	The tens digit of the alarm minutes value.
[3:0] AMIN	0–9	The ones digit of the alarm minutes value.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] AMIN	00h–3Bh	The alarm minutes value.

Real-Time Clock Alarm Hours Register

This register contains the alarm hours value. See Table 48.

Table 48. Real-Time Clock Alarm Hours Register (RTC_AHRS = 00EAh)

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

Note: X = Unchanged by RESET; R/W = Read/Write.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4] ATEN_HRS	0–2	The tens digit of the alarm hours value.
[3:0] AHRS	0–9	The ones digit of the alarm hours value.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:0] AHRS	00h–17h	The alarm hours value.

Real-Time Clock Alarm Day-of-the-Week Register

This register contains the alarm day-of-the-week value. See Table 49.

Table 49. Real-Time Clock Alarm Day-of-the-Week Register (RTC_ADOW = 00EBh)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	X	X	X	X
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-only if RTC locked, Read/Write if RTC unlocked.

Binary-Coded-Decimal Operation (BCD_EN = 1)

Bit Position	Value	Description
[7:4]	0000	Reserved.
[3:0] ADOW	1-7	The alarm day-of-the-week.value.

Binary Operation (BCD_EN = 0)

Bit Position	Value	Description
[7:4]	0000	Reserved.
[3:0] ADOW	01h–07h	The alarm day-of-the-week value.

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

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.

CLK_SEL and FREQ_SEL select the RTC clock source. If the 32KHz crystal option is selected the oscillator is enabled and the internal prescaler is set to divide by 32768. If the power-line frequency option is selected, the prescale value is set by FREQ_SEL, and the 32 Khz oscillator is disabled. See Table 51.

Table 51. Real-Time Clock Control Register (RTC_CTRL = 00EDh)

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

Note: X = Unchanged by RESET; R = Read-only; R/W = Read/Write.

Bit Position	**Value**	**Description**
7 ALARM	0	Alarm interrupt is inactive.
	1	Alarm interrupt is active.
6 INT_EN	0	Interrupt on alarm condition is disabled.
	1	Interrupt on alarm condition is enabled.
5 BCD_EN	0	RTC count and alarm value registers are binary.
	1	RTC count and alarm value registers are binary-coded decimal (BCD).
4 CLK_SEL	0	RTC clock source is crystal oscillator output (32768 Hz). On-chip 32768Hz oscillator is enabled.
	1	RTC clock source is power-line frequency input. On-chip 32768 Hz oscillator is disabled.
3 FREQ_SEL	0	Power-line frequency is 60 Hz.
	1	Power-line frequency is 50 Hz.
2	0	Reserved.
1 SLP_WAKE	0	RTC does not generate a Sleep-Mode Recovery reset.
	1	RTC Alarm generates a Sleep-Mode Recovery reset.
0 RTC_UNLOCK	0	RTC count registers are locked to prevent write access. RTC counter is enabled.
	1	RTC count registers are unlocked to allow write access. RTC counter is disabled.

Universal Asynchronous Receiver/ Transmitter

The UART module implements all of the logic required to support various asynchronous communications protocols. The module also implements two separate 16-byte-deep FIFOs for both transmission and reception. A block diagram of the UART is illustrated in Figure 23.

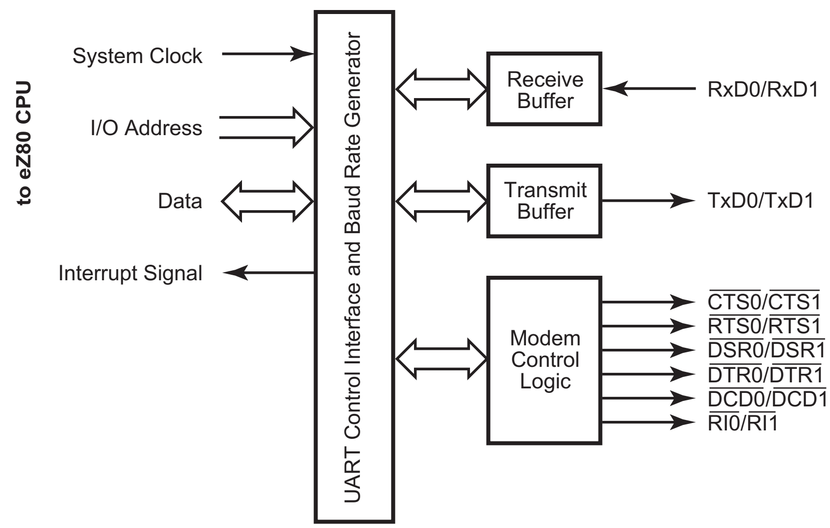


Figure 23.UART Block Diagram

The UART module provides the following asynchronous communication protocol-related features and functions:

- 5-, 6-, 7-, or 8-bit data transmission
- Even/odd or no parity bit generation and detection
- Start and stop bit generation and detection (supports up to two stop bits)
- Line break detection and generation
- Receiver overrun and framing errors detection
- Logic and associated I/O to provide modem handshake capability

UART Functional Description

The UART function implements:

- The transmitter and associated control logic
- The receiver and associated control logic
- The modem interface and associated logic

UART Transmitter

The transmitter block controls the data transmitted on the TxD output. It implements the FIFO, accessed through the UARTx_THR register, the transmit shift register, the parity generator, and control logic for the transmitter to control parameters for the asynchronous communication protocol.

The UARTx_THR is a Write-Only register. The processor writes the data byte to be transmitted into this register. In the FIFO mode, up to 16 data bytes can be written via the UARTx_THR register. The data byte from the FIFO is transferred to the transmit shift register at the appropriate time and transmitted out on TxD output. After SYNC_RESET, the UARTx_THR register is empty. Therefore, the Transmit Holding Register Empty (THRE) bit (bit 5 of the UARTx_LSR register) is 1 and an interrupt is sent to the processor (if interrupts are enabled). The processor can reset this interrupt by loading data into the UARTx_THR register, which clears the transmitter interrupt.

The transmit shift register places the byte to be transmitted on the TxD signal serially. The least-significant bit of the byte to be transmitted is shifted out first and the most significant bit is shifted out last. The control logic within the block adds the asynchronous communication protocol bits to the data byte being transmitted. The transmitter block obtains the parameters for the protocol from the bits programmed via the UARTx_LCTL register. The TxD output is set to 1 if the transmitter is idle (it does not contain any data to be transmitted).

The transmitter operates with the Baud Rate Generator (BRG) clock. The data bits are placed on the TxD output one time every 16 BRG clock cycles. The transmitter block also implements a parity generator that attaches the parity bit to the byte, if programmed.

UART Receiver

The receiver block controls the data reception from the RxD signal. The receiver block implements a receiver shift register, receiver line error condition monitoring logic and receiver data ready logic. It also implements the parity checker.

The UARTx_RBR is a Read-Only register of the module. The processor reads received data from this register. The condition of the UARTx_RBR register is monitored by the DR bit (bit 0 of the UARTx_LSR register). The DR bit is 1 when a data byte is received and transferred to the UARTx_RBR register from the receiver shift register. The DR bit is reset only when the processor reads all of the received data bytes. If the number of bits received is less than eight, the unused most significant bits of the data byte read are 0.

The receiver uses the clock from the BRG for receiving the data. This clock must be 16 times the appropriate baud rate. The receiver synchronizes the shift clock on the falling edge of the RxD input start bit. It then receives a complete byte according to the set parameters. The receiver also implements logic to detect framing errors, parity errors, overrun errors, and break signals.

UART Modem Control

The modem control logic provides two outputs and four inputs for handshaking with the modem. Any change in the modem status inputs, except RI, is detected and an interrupt can be generated. For RI, an interrupt is generated only when the trailing edge of the RI is detected. The module also provides LOOP mode for self-diagnostics.

UART Interrupts

There are five different sources of interrupts from the UART. The five sources of interrupts are:

- Transmitter
- Receiver (three different interrupts)
- Modem status

UART Transmitter Interrupt

The transmitter interrupt is generated if there is no data available for transmission. This interrupt can be disabled using the individual interrupt enable bit or cleared by writing data into the UARTx_THR register.

UART Receiver Interrupts

A receiver interrupt can be generated by three possible sources. The first source, a receiver data ready, indicates that one or more data bytes are received and are ready to be read. This interrupt is generated if the number of bytes in the receiver FIFO is greater than or equal to the trigger level. If the FIFO is not enabled, the interrupt is generated if the receive buffer contains a data byte. This interrupt is cleared by reading the UARTx_RBR.

The second interrupt source is the receiver time-out. A receiver time-out interrupt is generated when there are fewer data bytes in the receiver FIFO than the trigger level and there are no READs and WRITEs to or from the receiver FIFO for four consecutive byte times. When the receiver time-out interrupt is generated, it is cleared only after emptying the entire receive FIFO.

The first two interrupt sources from the receiver (data ready and time-out) share an interrupt enable bit.

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 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 interrupts are enabled (except for the transmit interrupt) and the application is ready to use the module for transmission/reception.

Data Transfers—Transmit. To transmit data, the application enables the transmit interrupt. An interrupt is immediately expected in response. The application reads the UARTx_IIR register and determines that the interrupt occurs due to an empty UARTx_THR register. When the application determines this occurrence, the application writes the transmit data bytes to the UARTx_THR register. The number of bytes that the application writes depends on whether or not the FIFO is enabled. If the FIFO is enabled, the application can write 16 bytes at a time. If not, the application can write one byte at a time. As a result of the first write, the interrupt is deactivated. The processor then waits for the next interrupt. When the interrupt is raised by the UART module, the processor repeats the same process until it exhausts all of the data for transmission.

To control and check the modem status, the application sets up the modem by writing to the UARTx_MCTL register and reading the UARTx_MCTL register before starting the process mentioned above.

Data Transfers—Receive. The receiver is always enabled, and it continually checks for the start bit on the RxD input signal. When an interrupt is raised by the UART module, the application reads the UARTx_IIR register and determines the cause for the interrupt. If the cause is a line status interrupt, the application reads the UARTx_LSR register, reads the data byte and then can discard the byte or take other appropriate action. If the interrupt is caused by a receive-data-ready condition, the application alternately reads the UARTx_LSR and UARTx_RBR registers and removes all of the received data bytes. It reads the UARTx_LSR register before reading the UARTx_RBR register to determine that there is no error in the received data.

To control and check modem status, the application sets up the modem by writing to the UARTx_MCTL register and reading the UARTx_MSR register before starting the process mentioned above.

Poll Mode Transfers. When interrupts are disabled, all data transfers are referred to as poll mode transfers. In poll mode transfers, the application must continually poll the UARTx_LSR register to transmit or receive data without enabling the interrupts. The same holds true for the UARTx_MSR register. If the interrupts are not enabled, the data in the UARTx_IIR register cannot be used to determine the cause of interrupt.

Baud Rate Generator

The Baud Rate Generator consists of a 16-bit downcounter, two registers, and associated decoding logic. The initial value of the Baud Rate Generator is defined by the two BRG Divisor Latch registers, {UARTx_BRG_H, UARTx_BRG_L}. At the rising edge of each system clock, the BRG decrements until it reaches the value `0001h`. On the next system clock rising edge, the BRG reloads the initial value from {UARTx_BRG_H, UARTx_BRG_L) and outputs a pulse to indicate the end-of-count. Calculate the UART data rate with the following equation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency}}{16 \times \text{(UART Baud Rate Generator Divisor)}}$$

Upon RESET, the 16-bit BRG divisor value resets to `0002h`. A minimum BRG divisor value of `0001h` is also valid, and effectively bypasses the BRG. A software Write to either the Low- or High-byte registers for the BRG Divisor Latch causes both the Low and High bytes to load into the BRG counter, and causes the count to restart.

The divisor registers can only be accessed if bit 7 of the UART Line Control register (UART*x*_LCTL) is set to 1. After reset, this bit is reset to 0.

Recommended Usage of the Baud Rate Generator

The following is the normal sequence of operations that should occur after the eZ80L92 is powered on to configure the Baud Rate Generator:

- Set UARTx_LCTL[7] to 1 to enable access of the BRG divisor registers
- Program the UARTx_BRG_L and UARTx_BRG_H registers
- Clear UARTx_LCTL[7] to 0 to disable access of the BRG divisor registers

BRG Control Registers

UART Baud Rate Generator Registers—Low and High Bytes

The registers hold the Low and High bytes of the 16-bit divisor count loaded by the processor for UART baud rate generation. The 16-bit clock divisor value is returned by {UART*x*_BRG_H, UART*x*_BRG_L}, where *x* is either 0 or 1 to identify the two available UART devices. Upon RESET, the 16-bit BRG divisor value resets to `0002h`. The initial 16-bit divisor value must be between `0002h` and `FFFFh` as the values `0000h` and `0001h` are invalid, and proper operation is not guaranteed. As a result, the minimum BRG clock divisor ratio is 2.

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. The count is then restarted.

Bit 7 of the associated UART Line Control register (UARTx_LCTL) must be set to 1 to access this register. See Tables 52 and 53. Refer to the UART Line Control Registers (UARTx_LCTL) on page 113 for more information.

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 should always be written with a value of 0. Read/Write attributes, reset conditions, and bit descriptions of all of the UART registers are provided in this section.

UART Transmit Holding Registers

If less than eight bits are programmed for transmission, the lower bits of the byte written to this register are selected for transmission. The transmit FIFO is mapped at this address. The user can write up to 16 bytes for transmission at one time to this address if the FIFO is enabled by the application. If the FIFO is disabled, this buffer is only one byte deep.

These registers share the same address space as the UARTx_RBR and UARTx_BRG_L registers. See Table 54.

Table 54. UART Transmit Holding Registers (UART0_THR = 00C0h, UART1_THR = 00D0h)

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: W = Write only.								

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

UART Receive Buffer Registers

The bits in this register reflect the data received. If less than eight bits are programmed for receive, the lower bits of the byte reflect the bits received whereas upper unused bits are 0. The receive FIFO is mapped at this address. If the FIFO is disabled, this buffer is only one byte deep.

These registers share the same address space as the UARTx_THR and UARTx_BRG_L registers. See Table 55.

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.

Bit Position	Value	Description
1 TIE	0	Transmit interrupt is disabled.
	1	Transmit interrupt is enabled. Interrupt is generated when the transmit FIFO/buffer is empty indicating no more bytes available for transmission.
0 RIE	0	Receive interrupt is disabled.
	1	Receive interrupt and receiver time-out interrupt are enabled. Interrupt is generated if the FIFO/buffer contains data ready to be read or if the receiver times out.

UART Interrupt Identification Registers

The Read-Only UARTx_IIR register allows the user to check whether the FIFO is enabled and the status of interrupts. These registers share the same I/O addresses as the UARTx_FCTL registers. See Tables 57 and 58.

Table 57. UART Interrupt Identification Registers (UART0_IIR = 00C2h, UART1_IIR = 00D2h)

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

Bit Position	Value	Description
[7:6] FSTS	00	FIFO is disabled.
	11	FIFO is enabled.
[5:4]	00	Reserved
[3:1] INSTS	000– 110	**Interrupt Status Code** The code indicated in these three bits is valid only if INTBIT is 1. If two internal interrupt sources are active and their respective enable bits are High, only the higher priority interrupt is seen by the application. The lower-priority interrupt code is indicated only after the higher-priority interrupt is serviced. Table 58 lists the interrupt status codes.
0 INTBIT	0	There is an active interrupt source within the UART.
	1	There is not an active interrupt source within the UART.

Table 58. UART Interrupt Status Codes

INSTS Value	Priority	Interrupt Type
011	Highest	Receiver Line Status
010	Second	Receive Data Ready or Trigger Level
110	Third	Character Time-out
001	Fourth	Transmit Buffer Empty
000	Lowest	Modem Status

UART FIFO Control Registers

This register is used to monitor trigger levels, clear FIFO pointers, and enable or disable the FIFO. The UARTx_FCTL registers share the same I/O addresses as the UARTx_IIR registers. See Table 59.

Table 59. UART FIFO Control Registers (UART0_FCTL = 00C2h, UART1_FCTL = 00D2h)

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
[7:6] TRIG	00	Receive FIFO trigger level set to 1. Receive data interrupt is generated when there is 1 byte in the FIFO. Valid only if FIFO is enabled.
	01	Receive FIFO trigger level set to 4. Receive data interrupt is generated when there are 4 bytes in the FIFO. Valid only if FIFO is enabled.
	10	Receive FIFO trigger level set to 8. Receive data interrupt is generated when there are 8 bytes in the FIFO. Valid only if FIFO is enabled.
	11	Receive FIFO trigger level set to 14. Receive data interrupt is generated when there are 14 bytes in the FIFO. Valid only if FIFO is enabled.
[5:3]	000	Reserved.

Bit Position	Value	Description
2 CLRTXF	0	No effect.
	1	Clear the transmit FIFO and reset the transmit FIFO pointer. Valid only if the FIFO is enabled.
1 CLRRXF	0	No effect.
	1	Clear the receive FIFO, clear the receive error FIFO, and reset the receive FIFO pointer. Valid only if the FIFO is enabled.
0 FIFOEN	0	Transmit and receive FIFOs are disabled. Transmit and receive buffers are only 1 byte deep.
	1	Transmit and receive FIFOs are enabled.

UART Line Control Registers

This register is used to control the communication control parameters. See Tables 60 and 61.

Table 60. UART Line Control Registers (UART0_LCTL = 00C3h, UART1_LCTL = 00D3h)

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 DLAB	0	Access to the UART registers at I/O addresses UARTx_RBR, UARTx_THR, and UARTx_IER is enabled.
	1	Access to the Baud Rate Generator registers at I/O addresses UARTx_BRG_L and UARTx_BRG_H is enabled.

Bit Position	Value	Description
6 SB	0	Do not send a BREAK signal.
	1	**Send Break** UART sends continuous zeroes on the transmit output from the next bit boundary. The transmit data in the transmit shift register is ignored. After forcing this bit High, the TxD output is 0 only after the bit boundary is reached. Just before forcing TxD to 0, the transmit FIFO is cleared. Any new data written to the transmit FIFO during a break should be written only after the THRE bit of UARTx_LSR register goes High. This new data is transmitted after the UART recovers from the break. After the break is removed, the UART recovers from the break for the next BRG edge.
5 FPE	0	Do not force a parity error.
	1	Force a parity error. When this bit and the party enable bit (PEN) are both 1, an incorrect parity bit is transmitted with the data byte.
4 EPS	0	Use odd parity for transmission. The total number of 1 bits in the transmit data plus parity bit is odd.
	1	Use even parity for transmission. The total number of 1 bits in the transmit data plus parity bit is even.
3 PEN	0	Parity bit transmit and receive is disabled.
	1	Parity bit transmit and receive is enabled. For transmit, a parity bit is generated and transmitted with every data character. For receive, the parity is checked for every incoming data character.
[2:0] CHAR	000– 111	**UART Character Parameter Selection** See Table 61 for a description of the values.

Table 61. UART Character Parameter Definition

CHAR[2:0]	Character Length (Tx/Rx Data Bits)	Stop Bits (Tx Stop Bits)
000	5	1
001	6	1
010	7	1
011	8	1

Table 61. UART Character Parameter Definition

CHAR[2:0]	Character Length (Tx/Rx Data Bits)	Stop Bits (Tx Stop Bits)
100	5	2
101	6	2
110	7	2
111	8	2

UART Modem Control Registers

This register is used to control and check the modem status. See Table 62.

Table 62. UART Modem Control Registers (UART0_MCTL = 00C4h, UART1_MCTL = 00D4h)

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

Bit Position	Value	Description
[7:5]	000b	Reserved.
4 LOOP	0	LOOP BACK mode is not enabled.
	1	LOOP BACK mode is enabled. The UART operates in internal LOOP BACK mode. The transmit data output port is disconnected from the internal transmit data output and set to 1. The receive data input port is disconnected and internal receive data is connected to internal transmit data. The modem status input ports are disconnected and the four bits of the modem control register are connected as modem status inputs. The two modem control output ports (OUT1&2) are set to their inactive state
3 OUT2	0–1	No function in normal operation. In LOOP BACK mode, this bit is connected to the DCD bit in the UART Status Register.

Bit Position	Value	Description
2 OUT1	0–1	No function in normal operation. In LOOP BACK mode, this bit is connected to the RI bit in the UART Status Register.
1 RTS	0–1	**Request to Send.** In normal operation, the $\overline{RTS}$ output port is the inverse of this bit. In LOOP BACK mode, this bit is connected to the CTS bit in the UART Status Register.
0 DTR	0–1	Data Terminal Ready. In normal operation, the $\overline{DTR}$ output port is the inverse of this bit. In LOOP BACK mode, this bit is connected to the DSR bit in the UART Status Register.

UART Line Status Registers

This register is used to show the status of UART interrupts and registers. See Table 63.

Table 63. UART Line Status Registers (UART0_LSR = 00C5h, UART1_LSR = 00D5h)

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

Bit Position	Value	Description
7 ERR	0	Always 0 when operating with the FIFO disabled. With the FIFO enabled, this bit is reset when the UARTx_LSR register is read and there are no more bytes with error status in the FIFO.
	1	Error detected in the FIFO. There is at least 1 parity, framing or break indication error in the FIFO.
6 TEMT	0	Transmit holding register/FIFO is not empty or transmit shift register is not empty or transmitter is not idle.
	1	Transmit holding register/FIFO and transmit shift register are empty; and the transmitter is idle. This bit cannot be set to 1 during the BREAK condition. This bit only becomes 1 after the BREAK command is removed.

Bit Position	Value	Description
5 THRE	0	Transmit holding register/FIFO is not empty.
	1	Transmit holding register/FIFO is empty. This bit cannot be set to 1 during the BREAK condition. This bit only becomes 1 after the BREAK command is removed.
4 BI	0	Receiver does not detect a BREAK condition. This bit is reset to 0 when the UARTx_LSR register is read.
	1	Receiver detects a BREAK condition on the receive input line. This bit is 1 if the duration of BREAK condition on the receive data is longer than one character transmission time, the time depends on the programming of the UARTx_LSR register. In case of FIFO only one null character is loaded into the receiver FIFO with the framing error. The framing error is revealed to the eZ80® whenever that particular data is read from the receiver FIFO.
3 FE	0	No framing error detected for character at the top of the FIFO. This bit is reset to 0 when the UARTx_LSR register is read.
	1	Framing error detected for the character at the top of the FIFO. This bit is set to 1 when the stop bit following the data/ parity bit is logic 0.
2 PE	0	The received character at the top of the FIFO does not contain a parity error. This bit is reset to 0 when the UARTx_LSR register is read.
	1	The received character at the top of the FIFO contains a parity error.
1 OE	0	The received character at the top of the FIFO does not contain an overrun error. This bit is reset to 0 when the UARTx_LSR register is read.
	1	Overrun error is detected. If the FIFO is not enabled, this indicates that the data in the receive buffer register was not read before the next character was transferred into the receiver buffer register. If the FIFO is enabled, this indicates the FIFO was already full when an additional character was received by the receiver shift register. The character in the receiver shift register is not put into the receiver FIFO.

Bit Position	Value	Description
0 DR	0	This bit is reset to 0 when the UARTx_RBR register is read or all bytes are read from the receiver FIFO.
	1	Data ready. If the FIFO is not enabled, this bit is set to 1 when a complete incoming character is transferred into the receiver buffer register from the receiver shift register. If the FIFO is enabled, this bit is set to 1 when a character is received and transferred to the receiver FIFO.

UART Modem Status Registers

This register is used to show the status of the UART signals. See Table 64.

Table 64. UART Modem Status Registers (UART0_MSR = 00C6h, UART1_MSR = 00D6h)

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 DCD	0–1	**Data Carrier Detect** In NORMAL mode, this bit reflects the inverted state of the DCDx input pin. In LOOP BACK mode, this bit reflects the value of the UARTx_MCTL[3] = out2.
6 RI	0–1	**Ring Indicator** In NORMAL mode, this bit reflects the inverted state of the $\overline{RIx}$ input pin. In LOOP BACK mode, this bit reflects the value of the UARTx_MCTL[2] = out1.
5 DSR	0–1	**Data Set Ready** In NORMAL mode, this bit reflects the inverted state of the DSRx input pin. In LOOP BACK mode, this bit reflects the value of the UARTx_MCTL[0] = DTR.
4 CTS	0–1	**Clear to Send** In NORMAL mode, this bit reflects the inverted state of the CTSx input pin. In LOOP BACK mode, this bit reflects the value of the UARTx_MCTL[1] = RTS.

Bit Position	Value	Description
3 DDCD	0–1	**Delta Status Change of $\overline{DCD}$** This bit is set to 1 whenever the DCDx pin changes state. This bit is reset to 0 when the UARTx_MSR register is read.
2 TERI	0–1	**Trailing Edge Change on $\overline{RI}$.** This bit is set to 1 whenever a falling edge is detected on the RIx pin. This bit is reset to 0 when the UARTx_MSR register is read.
1 DDSR	0–1	**Delta Status Change of $\overline{DSR}$** This bit is set to 1 whenever the DSRx pin changes state. This bit is reset to 0 when the UARTx_MSR register is read.
0 DCTS	0–1	**Delta Status Change of $\overline{CTS}$** This bit is set to 1 whenever the CTSx pin changes state. This bit is reset to 0 when the UARTx_MSR register is read.

UART Scratch Pad Registers

The UARTx_SPR register can be used by the system as a general-purpose Read/Write register. See Table 65.

Table 65. UART Scratch Pad Registers (UART0_SPR = 00C7h, UART1_SPR = 00D7h)

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
[7:0] SPR	00h–FFh	UART scratch pad register is available for use as a general-purpose Read/Write register.

Infrared Encoder/Decoder

The eZ80L92 contains a UART to infrared encoder/decoder (endec). The infrared encoder/decoder is integrated with the on-chip UART0 to allow easy communication between the eZ80® CPU and IrDA Physical Layer Specification Version 1.3 compliant infrared transceivers as illustrated in Figure 24. Infrared communication provides secure, reliable, high-speed, low-cost, point-to-point communication between PCs, PDAs, mobile telephones, printers and other infrared enabled devices.

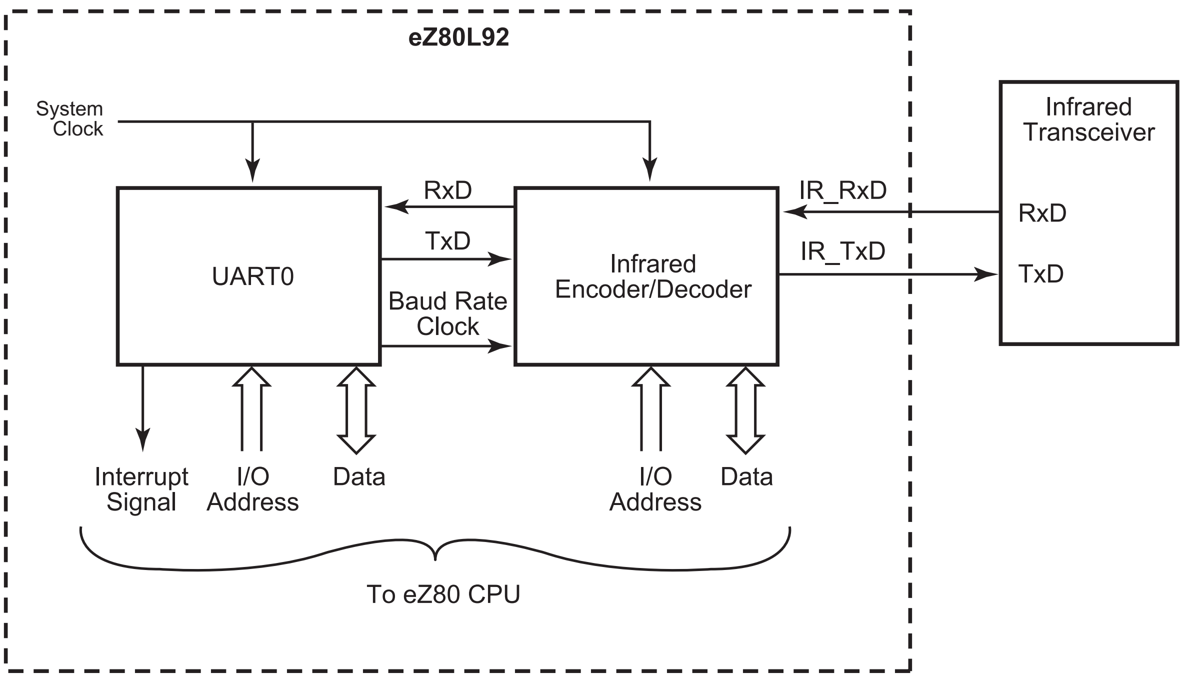


Figure 24. Infrared System Block Diagram

Functional Description

When the infrared encoder/decoder is enabled, the transmit data from the on-chip UART is encoded as digital signals in accordance with the IrDA standard and output to the infrared transceiver. Likewise, data received from the infrared transceiver is decoded by the infrared encoder/decoder and passed to the UART. Communication is half-duplex meaning that simultaneous data transmission and reception is not allowed.

The baud rate is set by the UART Baud Rate Generator and supports IrDA standard baud rates from 9600 bits/s to 115.2 KBPS. Higher baud rates are possible, but do not meet IrDA specifications. The UART must be enabled to use the infrared encoder/decoder. Refer to the section covering the Universal Asynchronous Receiver/Transmitter, on page 102, for more information on the UART and its Baud Rate Generator.

Transmit

The data to be transmitted via the IR transceiver is first sent to UART0. The UART transmit signal (TxD) and Baud Rate Clock are used by the infrared encoder/decoder to generate the modulation signal (IR_TXD) that drives the infrared transceiver. Each UART bit is 16-clocks wide. If the data to be transmitted is a logical 1 (High), the IR_TXD signal remains Low (0) for the full 16-clock period. If the data to be transmitted is a logical 0, a 3-clock High (1) pulse is output following a 7-clock Low (0) period. Following the 3-clock High pulse, a 6-clock Low pulse completes the full 16-clock data period. Data transmission is illustrated in Figure 25. During data transmission, the IR receive function should be disabled by clearing the IR_RXEN bit in the IR_CTL reg to 0. This prevents transmitter to receiver cross-talk.

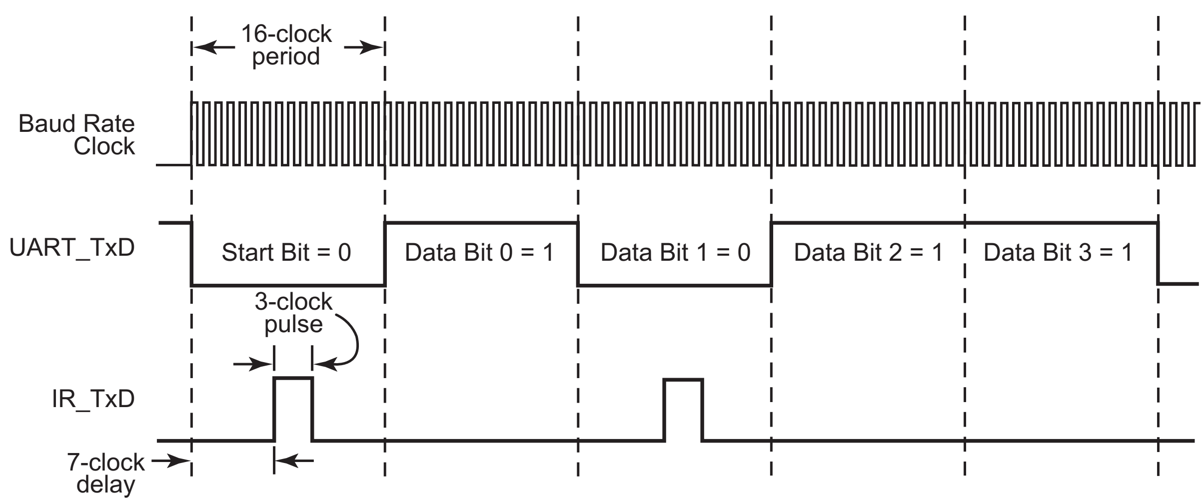


Figure 25. Infrared Data Transmission

Receive

Data received from the IR transceiver via the IR_RXD signal is decoded by the infrared encoder/decoder and passed to the UART. The IR_RXEN bit in the IR_CTL register must be set to enable the receiver decoder. The SIR data format uses half duplex communication therefore the UART should not be allowed to transmit while the receiver decoder is enabled. The UART Baud Rate Clock is used by the infrared encoder/decoder to generate the demodulated signal (RxD) that drives the UART. Each UART bit is 16-clocks wide. If the data to be received is a logical 1 (High), the IR_RXD signal remains High (1) for the full 16-clock period. If the data to be received is a logical 0, a 3-clock Low (0) pulse is output following a 7-clock High (1) period. Following the 3-clock Low pulse, is a 6-clock High pulse to complete the full 16-clock data period. Data transmission is illustrated in Figure 26.

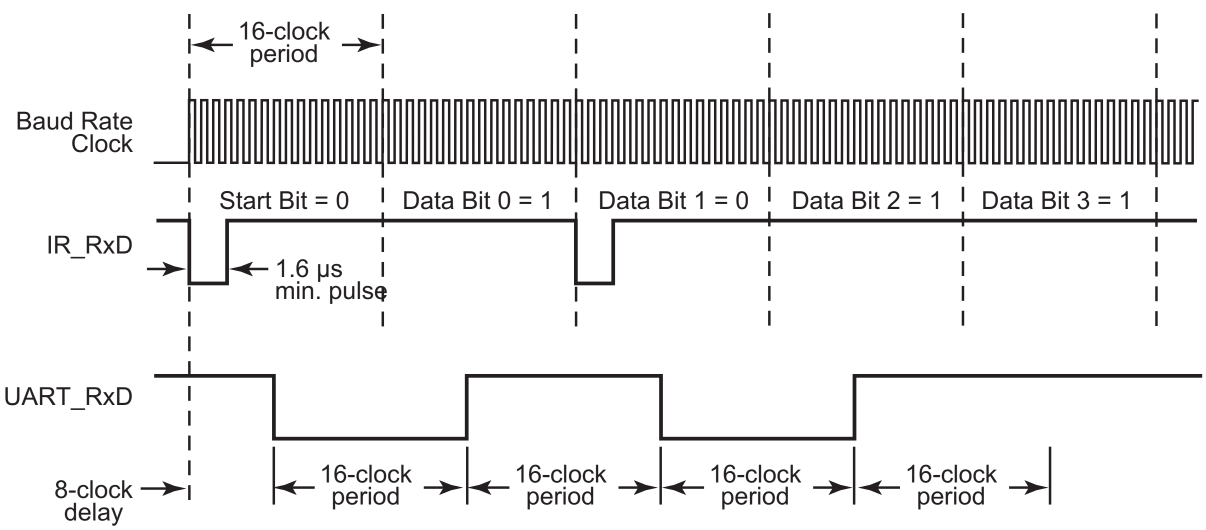


Figure 26. Infrared Data Reception

Jitter

Due to the inherent sampling of the received IR_RXD signal by the BIt Rate Clock, some jitter can be expected on the first bit in any sequence of data. However, all subsequent bits in the received data stream are a fixed 16-clock periods wide.

Infrared Encoder/Decoder Signal Pins

The infrared encoder/decoder signal pins (IR_TXD and IR_RXD) are multiplexed with General-Purpose I/O (GPIO) pins. These GPIO pins must be configured for alternate function operation for the infrared encoder/decoder to operate.

The remaining six UART0 pins ($\overline{\text{CTS0}}$, $\overline{\text{DCD0}}$, $\overline{\text{DSR0}}$, $\overline{\text{DTR0}}$, $\overline{\text{RTS0}}$ and $\overline{\text{RI0}}$) are not required for use with the infrared encoder/decoder. The UART0 modem status interrupt should be disabled to prevent unwanted interrupts from these pins. The

GPIO pins corresponding to these six unused UART0 pins can be used for inputs, outputs, or interrupt sources. Recommended GPIO Port D control register settings are provided in Table 66. Refer to the section covering the General-Purpose Input/Output, on page 39 for additional information on setting the GPIO Port modes

Table 66. GPIO Mode Selection when using the IrDA Encoder/Decoder

GPIO Port D Bits	Allowable GPIO Port Mode	Allowable Port Mode Functions
PD0	7	Alternate Function
PD1	7	Alternate Function
PD2–PD7	Any other than GPIO Mode 7 (1, 2, 3, 4, 5, 6, 8, or 9)	Output, Input, Open-Drain, Open-Source, Level-sensitive Interrupt Input, or Edge-Triggered Interrupt Input

Loopback Testing

Both internal and external loopback testing can be accomplished with the endec on the eZ80L92. Internal loopback testing is enabled by setting the LOOP_BACK bit to 1. During internal loopback, IR_TXD output signal is inverted and connected on-chip to the IR_RXD input. External loopback testing of the off-chip IrDA transceiver may be accomplished by transmitting data from the UART while the receiver is enabled (IR_RXEN set to 1).

Infrared Encoder/Decoder Register

After a RESET, the infrared encoder/decoder register is set to its default value. Any writes to unused register bits are ignored and reads return a value of 0. Unused bits within a register must always be written with a value of 0. The IR_CTL register is described in Table 67.

Table 67. Infrared Encoder/Decoder Control Register (IR_CTL = 00BFh)

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
[7:3]	000000	Reserved.
2 LOOP_BACK	0	Internal LOOP BACK mode is disabled.
	1	Internal LOOP BACK mode is enabled. IR_TXD output is inverted and connected to IR_RXD input for internal loop back testing.
1 IR_RXEN	0	IR_RXD data is ignored.
	1	IR_RXD data is passed to UART0 RxD.
0 IR_EN	0	Infrared Encoder/Decoder is disabled.
	1	Infrared Encoder/Decoder is enabled.

Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a synchronous interface allowing several SPI-type devices to be interconnected. The SPI is a full-duplex, synchronous, character-oriented communication channel that employs a four-wire interface. The SPI block consists of a transmitter, receiver, baud rate generator, and control unit. During an SPI transfer, data is sent and received simultaneously by both the master and the slave SPI devices.

In a serial peripheral interface, separate signals are required for data and clock. The SPI may be configured as either a master or a slave. The connection of two SPI devices (one master and one slave) and the direction of data transfer is demonstrated in Figures 27 and 28.

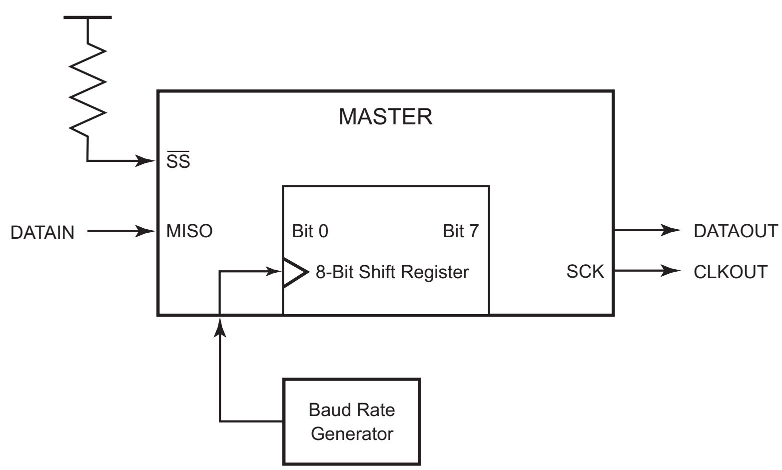


Figure 27. SPI Master Device

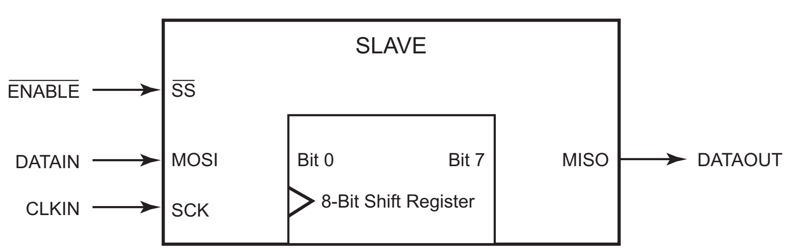


Figure 28. SPI Slave Device

SPI Signals

The four basic SPI signals are:

- MISO (Master-In, Slave-Out)
- MOSI (Master-Out, Slave-In)
- SCK (SPI Serial Clock)
- $\overline{SS}$ (Slave Select)

These SPI signals are discussed in the following paragraphs. Each signal is described in both MASTER and SLAVE modes.

Master-In, Slave-Out

The Master-In, Slave-Out (MISO) pin is configured as an input in a master device and as an output in a slave device. It is one of the two lines that transfer serial data, with the most significant bit sent first. The MISO pin of a slave device is placed in a high-impedance state if the slave is not selected. When the SPI is not enabled, this signal is in a high-impedance state.

Master-Out, Slave-In

The Master-Out, Slave-In (MOSI) pin is configured as an output in a master device and as an input in a slave device. It is one of the two lines that transfer serial data, with the most significant bit sent first. When the SPI is not enabled, this signal is in a high-impedance state.

Slave Select

The active Low Slave Select ($\overline{SS}$) input signal is used to select the SPI as a slave device. It must be Low prior to all data communication and must stay Low for the duration of the data transfer.

The $\overline{SS}$ input signal must be High for the SPI to operate as a master device. If the SS signal goes Low, a Mode Fault error flag (MODF) is set in the SPI_SR register. See the SPI Status Register (SPI_SR) on page 133 for more information.

When CPHA (Clock Phase) is set to 0, the shift clock is the logical OR of $\overline{SS}$ with SCK. In this clock phase mode, SS must go High between successive characters in an SPI message. When CPHA is set to 1, SS can remain Low for several SPI characters. In cases where there is only one SPI slave, its SS line could be tied Low as long as CPHA is set to 1. See the SPI Control Register (SPI_CTL) on page 132 for more information on CPHA.

Serial Clock

The Serial Clock (SCK) is used to synchronize data movement both in and out of the device through its MOSI and MISO pins. The master and slave are each capable of exchanging a byte of data during a sequence of eight clock cycles. Because SCK is generated by the master, the SCK pin becomes an input on a slave device. The SPI contains an internal divide-by-two clock divider. In MASTER mode, the SPI serial clock is one-half the frequency of the clock signal created by the SPI's Baud Rate Generator.

As demonstrated in Figure 29 and Table 68, four possible timing relations may be chosen by using control bits CPOL and CPHA in the SPI Control register. See the SPI Control Register (SPI_CTL) on page 132. Both the master and slave must operate with the identical timing, CPOL (Clock Polarity), and CPHA. The master device always places data on the MOSI line a half-cycle before the clock edge (SCK signal), in order for the slave device to latch the data.

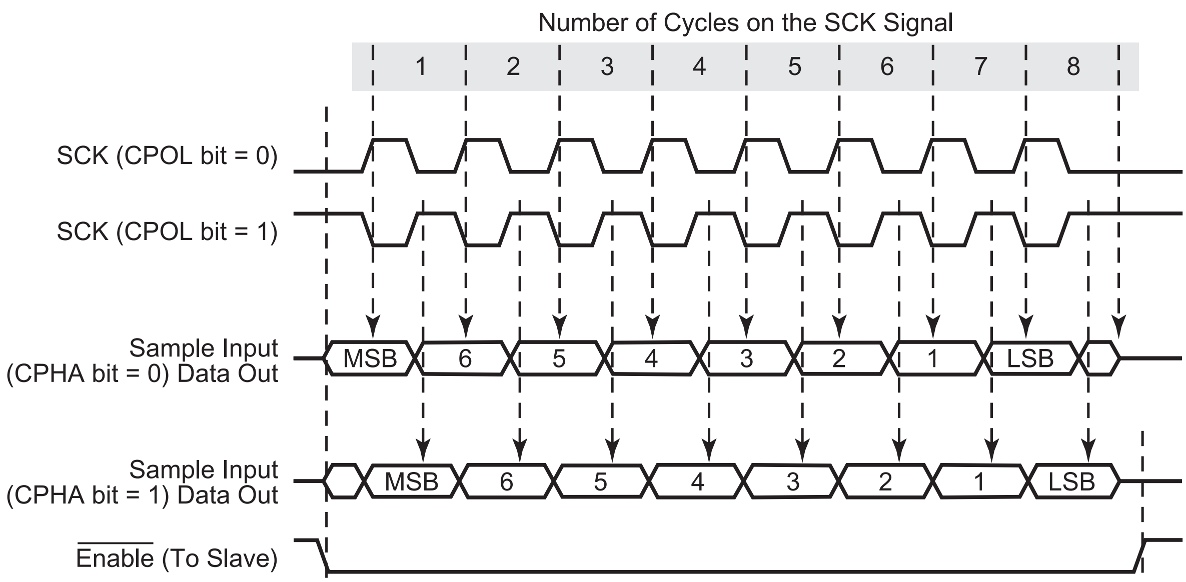


Figure 29. SPI Timing

Table 68. SPI Clock Phase (CPHA) and Clock Polarity (CPOL) Operation

CPHA	CPOL	SCK Transmit Edge	SCK Receive Edge	SCK Idle State	SS High Between Characters?
0	0	Falling	Rising	Low	Yes
0	1	Rising	Falling	High	Yes
1	0	Rising	Falling	Low	No
1	1	Falling	Rising	High	No

SPI Functional Description

When a master transmits to a slave device via the MOSI signal, the slave device responds by sending data to the master via the master's MISO signal. The resulting implication is a full-duplex transmission, with both data out and data in synchronized with the same clock signal. Thus the byte transmitted is replaced by the byte received and eliminates the requirement for separate transmit-empty and receive-full status bits. A single status bit, SPIF, is used to signify that the I/O operation is completed, see the SPI Status Register (SPI_SR) on page 133.

The SPI is double-buffered on Read, but not on Write. If a Write is performed during data transfer, the transfer occurs uninterrupted, and the Write is unsuccessful. This condition causes the WRITE COLLISION (WCOL) status bit in the SPI_SR register to be set. After a data byte is shifted, the SPIF flag of the SPI_SR register is set.

In SPI MASTER mode, the SCK pin is an output. It idles High or Low, depending on the CPOL bit in the SPI_CTL register, until data is written to the shift register. Data transfer is initiated by writing to the transmit shift register, SPI_TSR. Eight clocks are then generated to shift the eight bits of transmit data out the MOSI pin while shifting in eight bits of data on the MISO pin. After transfer, the SCK signal idles.

In SPI SLAVE mode, the start logic receives a logic Low from the $\overline{SS}$ pin and a clock input at the SCK pin, and the slave is synchronized to the master. Data from the master is received serially from the slave MOSI signal and loads the 8-bit shift register. After the 8-bit shift register is loaded, its data is parallel transferred to the Read buffer. During a Write cycle data is written into the shift register, then the slave waits for the SPI master to initiate a data transfer, supply a clock signal, and shift the data out on the slave's MISO signal.

If the CPHA bit in the SPI_CTL register is 0, a transfer begins when $\overline{SS}$ pin signal goes Low and the transfer ends when $\overline{SS}$ goes High after eight clock cycles on SCK. When the CPHA bit is set to 1, a transfer begins the first time SCK becomes active while $\overline{SS}$ is Low and the transfer ends when the SPIF flag gets set.

SPI Flags

Mode Fault

The Mode Fault flag (MODF) indicates that there may be a multimaster conflict for system control. The MODF bit is normally cleared to 0 and is only set to 1 when the master device's $\overline{SS}$ pin is pulled Low. When a mode fault is detected, the following occurs:

1. The MODF flag (SPI_SR[4]) is set to 1.
2. The SPI device is disabled by clearing the SPI_EN bit (SPI_CTL[5]) to 0.

3. The MASTER_EN bit (SPI_CTL[4]) is cleared to 0, forcing the device into SLAVE mode.
4. If the SPI interrupt is enabled by setting IRQ_EN (SPI_CTL[7]) High, an SPI interrupt is generated.

Clearing the Mode Fault flag is performed by reading the SPI Status register. The other SPI control bits (SPI_EN and MASTER_EN) must be restored to their original states by user software after the Mode Fault flag is cleared.

Write Collision

The WRITE COLLISION flag, WCOL (SPI_SR[5]), is set to 1 when an attempt is made to write to the SPI Transmit Shift register (SPI_TSR) while data transfer occurs. Clearing the WCOL bit is performed by reading SPI_SR with the WCOL bit set.

SPI Baud Rate Generator

The SPI's Baud Rate Generator creates a lower frequency clock from the high-frequency system clock. The Baud Rate Generator output is used as the clock source by the SPI.

Baud Rate Generator Functional Description

The SPI's Baud Rate Generator consists of a 16-bit downcounter, two 8-bit registers, and associated decoding logic. The Baud Rate Generator's initial value is defined by the two BRG Divisor Latch registers, {SPI_BRG_H, SPI_BRG_L}. At the rising edge of each system clock, the BRG decrements until it reaches the value `0001h`. On the next system clock rising edge, the BRG reloads the initial value from {SPI_BRG_H, SPI_BRG_L) and outputs a pulse to indicate the end-of-count. Calculate the SPI Data Rate with the following equation:

$$\text{SPI Data Rate (bits/s)} = \frac{\text{System Clock Frequency}}{\text{2 X SPI Baud Rate Generator Divisor}}$$

Upon RESET, the 16-bit BRG divisor value resets to `0002h`. When the SPI is operating as a Master, the BRG divisor value must be set to a value of `0003h` or greater. When the SPI is operating as a Slave, the BRG divisor value must be set to a value of `0004h` or greater. A software Write to either the Low- or High-byte registers for the BRG Divisor Latch causes both the Low and High bytes to load into the BRG counter, and causes the count to restart.

Data Transfer Procedure with SPI Configured as the Master

1. Load the SPI Baud Rate Generator Registers, SPI_BRG_H and SPI_BRG_L.
2. External device must deassert the $\overline{\text{SS}}$ pin if currently asserted.
3. Load the SPI Control Register, SPI_CTL.
4. Assert the $\overline{\text{ENABLE}}$ pin of the slave device using a GPIO pin.
5. Load the SPI Transmit Shift Register, SPI_TSR.
6. When the SPI data transfer is complete, deassert the $\overline{\text{ENABLE}}$ pin of the slave device.

Data Transfer Procedure with SPI Configured as a 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 paragraphs.

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 Tables 69 and 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}.

Table 70. SPI Baud Rate Generator Register—High Byte (SPI_BRG_H = 00B9h)

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
[7:0] SPI_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 {SPI_BRG_H, SPI_BRG_L}.

SPI Control Register

This register is used to control and setup the serial peripheral interface. The SPI should be disabled prior to making any changes to CPHA or CPOL. See Table 71.

Table 71. SPI Control Register (SPI_CTL = 00BAh)

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

Bit Position	Value	Description
7 IRQ_EN	0	SPI system interrupt is disabled.
	1	SPI system interrupt is enabled.
6	0	Reserved.
5 SPI_EN	0	SPI is disabled.
	1	SPI is enabled.
4 MASTER_EN	0	When enabled, the SPI operates as a slave.
	1	When enabled, the SPI operates as a master.
3 CPOL	0	Master SCK pin idles in a Low (0) state.
	1	Master SCK pin idles in a High (1) state.
2 CPHA	0	SS must go High after transfer of every byte of data.
	1	SS can remain Low to transfer any number of data bytes.
[1:0]	00	Reserved.

SPI Status Register

The SPI Status Read-Only register returns the status of data transmitted using the serial peripheral interface. Reading the SPI_SR register clears Bits 7, 6, and 4 to a logical 0. See Table 72.

Table 72. SPI Status Register (SPI_SR = 00BBh)

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.								

Bit Position	Value	Description
7 SPIF	0	SPI data transfer is not finished.
	1	SPI data transfer is finished. If enabled, an interrupt is generated. This bit flag is cleared to 0 by a Read of the SPI_SR register.
6 WCOL	0	An SPI write collision is not detected.
	1	An SPI write collision is detected. This bit flag is cleared to 0 by a Read of the SPI_SR registers.
5	0	Reserved.
4 MODF	0	A mode fault (multimaster conflict) is not detected.
	1	A mode fault (multimaster conflict) is detected. This bit flag is cleared to 0 by a Read of the SPI_SR register.
[3:0]	0000	Reserved.

SPI Transmit Shift Register

The SPI Transmit Shift register (SPI_TSR) is used by the SPI master to transmit data onto the SPI serial bus to the slave device. A Write to the SPI_TSR register places data directly into the shift register for transmission. A Write to this register within an SPI device configured as a master initiates transmission of the byte of the data loaded into the register. At the completion of transmitting a byte of data, the SPIF status bit (SPI_SR[7]) is set to 1 in both the master and slave devices.

The SPI Transmit Shift Write-Only register shares the same address space as the SPI Receive Buffer Read-Only register. See Table 73.

Table 73. SPI Transmit Shift Register (SPI_TSR = 00BCh)

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: W = Write only.								

Bit Position	Value	Description
[7:0] TX_DATA	00h–FFh	SPI transmit data.

SPI Receive Buffer Register

The SPI Receive Buffer register (SPI_RBR) is used by the SPI slave to receive data from the serial bus. The SPIF bit must be cleared prior to a second transfer of data from the shift register or an overrun condition exists. In cases of overrun the byte that caused the overrun is lost.

The SPI Receive Buffer Read-Only register shares the same address space as the SPI Transmit Shift Write-Only register. See Table 74.

Table 74. SPI Receive Buffer Register (SPI_RBR = 00BCh)

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] RX_DATA	00h–FFh	SPI received data.

I²C Serial I/O Interface

I²C General Characteristics

The I²C serial I/O bus is a two-wire communication interface that can operate in four modes:

- MASTER TRANSMIT
- MASTER RECEIVE
- SLAVE TRANSMIT
- SLAVE RECEIVE

The I²C interface consists of the Serial Clock (SCL) and the Serial Data (SDA). Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via an external pull-up resistor. When the bus is free, both lines are High. The output stages of devices connected to the bus must be configured as open-drain outputs. Data on the I²C bus can be transferred at a rate of up to 100 KBPS in STANDARD mode, or up to 400 KBPS in FAST mode. One clock pulse is generated for each data bit transferred.

Clocking Overview

If another device on the I²C bus drives the clock line when the I²C is in MASTER mode, the I²C synchronizes its clock to the I²C bus clock. The High period of the clock is determined by the device that generates the shortest High clock period. The Low period of the clock is determined by the device that generates the longest Low clock period.

A slave may stretch the Low period of the clock to slow down the bus master. The Low period may also be stretched for handshaking purposes. This can be done after each bit transfer or each byte transfer. The I²C stretches the clock after each byte transfer until the IFLG bit in the I2C_CTL register is cleared.

Bus Arbitration Overview

In MASTER mode, the I²C checks that each transmitted logic 1 appears on the I²C bus as a logic 1. If another device on the bus overrules and pulls the SDA signal Low, arbitration is lost. If arbitration is lost during the transmission of a data byte or a Not-Acknowledge bit, the I²C returns to the idle state. If arbitration is lost during the transmission of an address, the I²C switches to SLAVE mode so that it can recognize its own slave address or the general call address.

Data Validity

The data on the SDA line must be stable during the High period of the clock. The High or Low state of the data line can only change when the clock signal on the SCL line is Low as illustrated in Figure 30.

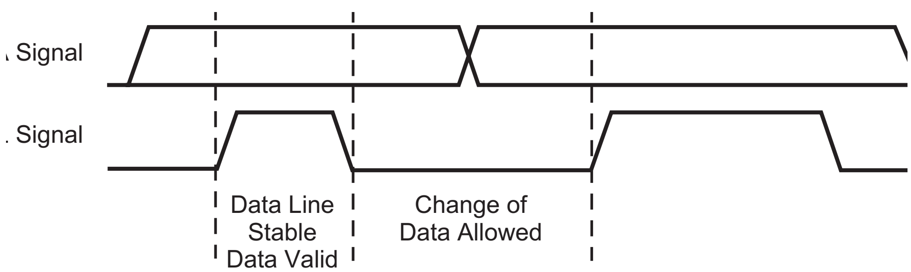


Figure 30. I^{2}C Clock and Data Relationship

START and STOP Conditions

Within the I^{2}C bus protocol, unique situations arise which are defined as START and STOP conditions. See Figure 31. A High-to-Low transition on the SDA line while SCL is High indicates a START condition. A Low-to-High transition on the SDA line while SCL is High defines a STOP condition.

START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition. The bus is considered to be free a defined time after the STOP condition.

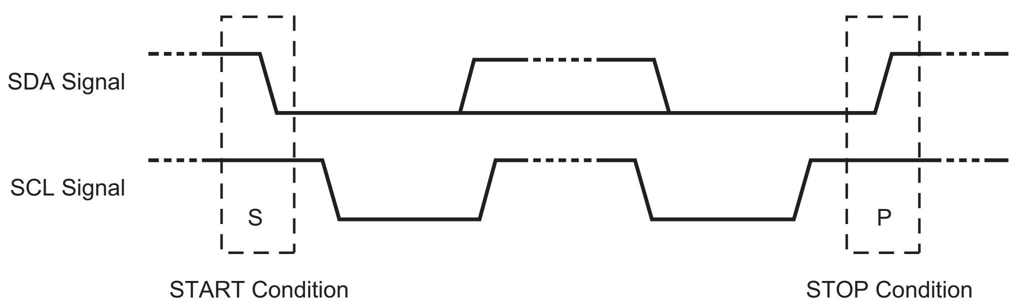


Figure 31. START and STOP Conditions In I^{2}C Protocol

Transferring Data

Byte Format

Every character transferred on the SDA line must be a single 8-bit byte. The number of bytes that can be transmitted per transfer is unrestricted. Each byte must be followed by an Acknowledge (ACK)[1]. Data is transferred with the most significant bit (msb) first. See Figure 32. A receiver can hold the SCL line Low to force the transmitter into a wait state. Data transfer then continues when the receiver is ready for another byte of data and releases SCL.

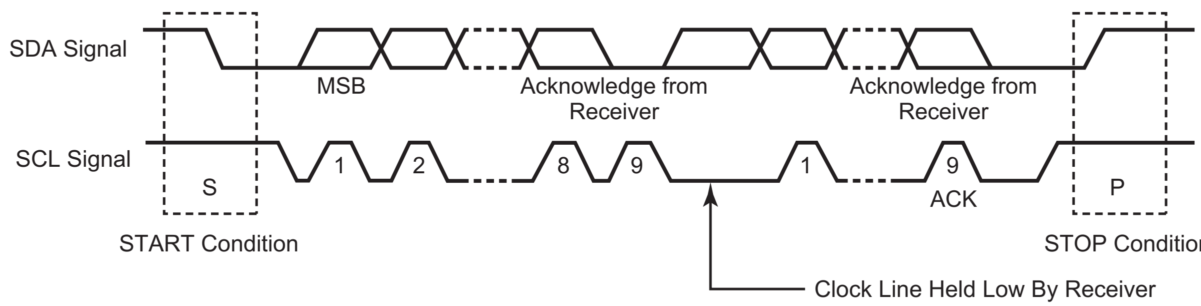


Figure 32. I²C Frame Structure

Acknowledge

Data transfer with an ACK function is obligatory. The ACK-related clock pulse is generated by the master. The transmitter releases the SDA line (High) during the ACK clock pulse. The receiver must pull down the SDA line during the ACK clock pulse so that it remains stable Low during the High period of this clock pulse. See Figure 33.

A receiver that is addressed is obliged to generate an ACK after each byte is received. When a slave-receiver doesn't acknowledge the slave address (for example, unable to receive because it's performing some real-time function), the data line must be left High by the slave. The master then generates a STOP condition to abort the transfer.

If a slave-receiver acknowledges the slave address, but cannot receive any more data bytes, the master must abort the transfer. The abort is indicated by the slave generating the Not Acknowledge (NACK) on the first byte to follow. The slave leaves the data line High and the master generates the STOP condition.

1. ACK is defined as a general Acknowledge bit. By contrast, the I²C Acknowledge bit is represented as AAK, bit 2 of the I²C Control Register, which identifies which ACK signal to transmit. See Table 84 on page 151.

If a master-receiver is involved in a transfer, it must signal the end of data to the slave-transmitter by not generating an ACK on the final byte that is clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate a STOP or a repeated START condition.

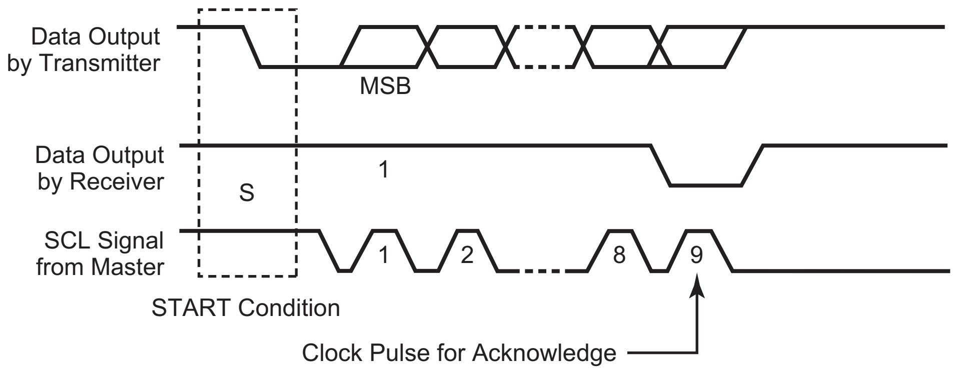


Figure 33. I^{2}C Acknowledge

Clock Synchronization

All masters generate their own clocks on the SCL line to transfer messages on the I^{2}C bus. Data is only valid during the High period of each clock.

Clock synchronization is performed using the wired AND connection of the I^{2}C interfaces to the SCL line, meaning that a High-to-Low transition on the SCL line causes the relevant devices to start counting from their Low period. When a device clock goes Low, it holds the SCL line in that state until the clock High state is reached. See Figure 34. The Low-to-High transition of this clock, however, may not change the state of the SCL line if another clock is still within its Low period. The SCL line is held Low by the device with the longest Low period. Devices with shorter Low periods enter a High wait-state during this time.

When all devices concerned count off their Low period, the clock line is released and goes High. There is no difference between the device clocks and the state of the SCL line, and all of the devices start counting their High periods. The first device to complete its High period again pulls the SCL line Low. In this way, a synchronized SCL clock is generated with its Low period determined by the device with the longest clock Low period, and its High period determined by the one with the shortest clock High period.

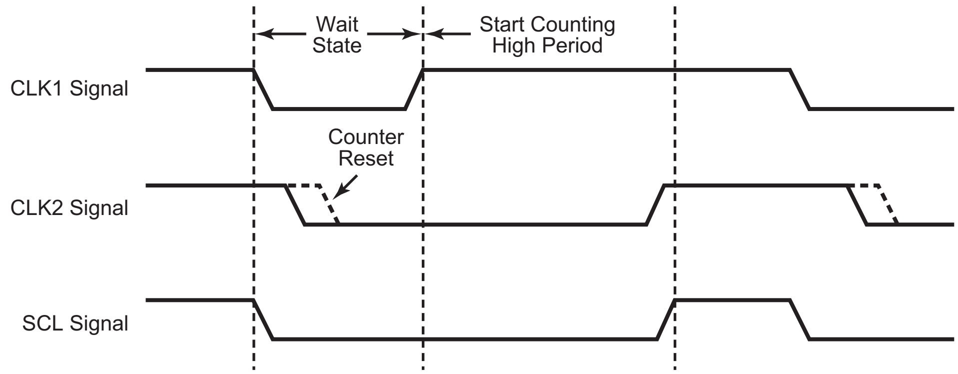


Figure 34. Clock Synchronization In I^{2}C Protocol

Arbitration

A master may start a transfer only if the bus is free. Two or more masters may generate a START condition within the minimum hold time of the START condition which results in a defined START condition to the bus. Arbitration takes place on the SDA line, while the SCL line is at the High level, in such a way that the master which transmits a High level, while another master is transmitting a Low level switches off its data output stage because the level on the bus doesn't correspond to its own level.

Arbitration can continue for many bits. Its first stage is comparison of the address bits. If the masters are each trying to address the same device, arbitration continues with comparison of the data. Because address and data information on the I^{2}C bus is used for arbitration, no information is lost during this process. A master which loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration.

If a master also incorporates a slave function and it loses arbitration during the addressing stage, it's possible that the winning master is trying to address it. The losing master must switch over immediately to its slave-receiver mode. Figure 34 illustrates the arbitration procedure for two masters. Of course, more may be involved (depending on how many masters are connected to the bus). The moment there is a difference between the internal data level of the master generating DATA 1 and the actual level on the SDA line, its data output is switched off, which means that a High output level is then connected to the bus. As a result, the data transfer initiated by the winning master is not affected. Because control of the I^{2}C bus is decided solely on the address and data sent by competing masters, there is no central master, nor any order of priority on the bus.

Special attention must be paid if, during a serial transfer, the arbitration procedure is still in progress at the moment when a repeated START condition or a STOP condition is transmitted to the I^{2}C bus. If it is possible for such a situation to occur, the masters involved must send this repeated START condition or STOP condition at the same position in the format frame. In other words, arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

Clock Synchronization for Handshake

The Clock synchronizing mechanism can function as a handshake, enabling receivers to cope with fast data transfers, on either a byte or bit level. The byte level allows a device to receive a byte of data at a fast rate, but allows the device more time to store the received byte or to prepare another byte for transmission. Slaves hold the SCL line Low after reception and acknowledge the byte, forcing the master into a wait state until the slave is ready for the next byte transfer in a handshake procedure.

Operating Modes

Master Transmit

In MASTER TRANSMIT mode, the I^{2}C transmits a number of bytes to a slave receiver.

Enter MASTER TRANSMIT mode by setting the STA bit in the I2C_CTL register to 1. The I^{2}C then tests the I^{2}C bus and transmits a START condition when the bus is free. When a START condition is transmitted, the IFLG bit is 1 and the status code in the I2C_SR register is `08h`. Before this interrupt is serviced, the I2C_DR register must be loaded with either a 7-bit slave address or the first part of a 10-bit slave address, with the lsb cleared to 0 to specify TRANSMIT mode. The IFLG bit should now be cleared to 0 to prompt the transfer to continue.

After the 7-bit slave address (or the first part of a 10-bit address) plus the Write bit are transmitted, the IFLG is set again. A number of status codes are possible in the I2C_SR register. See Table 75.

Table 75. I^{2}C Master Transmit Status Codes

Code	I^{2}C State	Microprocessor Response	Next I^{2}C Action
18h	Addr+W transmitted[1], ACK received	For a 7-bit address: write byte to DATA, clear IFLG	Transmit data byte, receive ACK
		Or set STA, clear IFLG	Transmit repeated START
		Or set STP, clear IFLG	Transmit STOP
		Or set STA & STP, clear IFLG	Transmit STOP then START
		For a 10-bit address: write extended address byte to DATA, clear IFLG	Transmit extended address byte
20h	Addr+W transmitted, ACK not received	Same as code 18h	Same as code 18h
38h	Arbitration lost	Clear IFLG	Return to idle
		Or set STA, clear IFLG	Transmit START when bus is free
68h	Arbitration lost, +W received, ACK transmitted	Clear IFLG, AAK = 0[2]	Receive data byte, transmit NACK
		Or clear IFLG, AAK = 1	Receive data byte, transmit ACK
78h	Arbitration lost, General call addr received, ACK transmitted	Same as code 68h	Same as code 68h
B0h	Arbitration lost, SLA+R received, ACK transmitted	Write byte to DATA, clear IFLG, clear AAK = 0	Transmit last byte, receive ACK
		Or write byte to DATA, clear IFLG, set AAK = 1	Transmit data byte, receive ACK

Notes:
1. W is defined as the write bit; i.e., the lsb is cleared to 0.
2. AAK is defined as the I^{2}C Acknowledge bit.

If 10-bit addressing is being used, then the status code is `18h` or `20h` after the first part of a 10-bit address plus the Write bit are successfully transmitted.

After this interrupt is serviced and the second part of the 10-bit address is transmitted, the I2C_SR register contains one of the codes in Table 76.

Table 76. I^{2}C 10-Bit Master Transmit Status Codes

Code	I^{2}C State	Microprocessor Response	Next I^{2}C Action
38h	Arbitration lost	Clear IFLG	Return to idle
		Or set STA, clear IFLG	Transmit START when bus free
68h	Arbitration lost, SLA+W received, ACK transmitted	Clear IFLG, clear AAK = 0	Receive data byte, transmit NACK
		Or clear IFLG, set AAK = 1	Receive data byte, transmit ACK
B0h	Arbitration lost, SLA+R received, ACK transmitted	Write byte to DATA, clear IFLG, clear AAK = 0	Transmit last byte, receive ACK
		Or write byte to DATA, clear IFLG, set AAK = 1	Transmit data byte, receive ACK
D0h	Second Address byte + W transmitted, ACK received	Write byte to DATA, clear IFLG	Transmit data byte, receive ACK
		Or set STA, clear IFLG	Transmit repeated START
		Or set STP, clear IFLG	Transmit STOP
		Or set STA & STP, clear IFLG	Transmit STOP then START
D8h	Second Address byte + W transmitted, ACK not received	Same as code D0h	Same as code D0h

If a repeated START condition is transmitted, the status code is `10h` instead of `08h`.

After each data byte is transmitted, the IFLG is 1 and one of the status codes listed in Table 77 is in the I2C_SR register.

Table 77. I²C Master Transmit Status Codes For Data Bytes

Code	I²C State	Microprocessor Response	Next I²C Action
28h	Data byte transmitted, ACK received	Write byte to DATA, clear IFLG	Transmit data byte, receive ACK
		Or set STA, clear IFLG	Transmit repeated START
		Or set STP, clear IFLG	Transmit STOP
		Or set STA & STP, clear IFLG	Transmit START then STOP
30h	Data byte transmitted, ACK not received	Same as code 28h	Same as code 28h
38h	Arbitration lost	Clear IFLG	Return to idle
		Or set STA, clear IFLG	Transmit START when bus free

When all bytes are transmitted, the microprocessor should write a 1 to the STP bit in the I2C_CTL register. The I²C then transmits a STOP condition, clears the STP bit and returns to the idle state.

Master Receive

In MASTER RECEIVE mode, the I²C receives a number of bytes from a slave transmitter.

After the START condition is transmitted, the IFLG bit is 1 and the status code `08h` is loaded in the I2C_SR register. The I2C_DR register should be loaded with the slave address (or the first part of a 10-bit slave address), with the lsb set to 1 to signify a READ. The IFLG bit should be cleared to 0 as a prompt for the transfer to continue.

When the 7-bit slave address (or the first part of a 10-bit address) and the Read bit are transmitted, the IFLG bit is set and one of the status codes listed in Table 78 is in the I2C_SR register.

Table 78. I^2C Master Receive Status Codes

Code	I^2C State	Microprocessor Response	Next I^2C Action
40h	Addr + R transmitted, ACK received	For a 7-bit address, clear IFLG, AAK = 0	Receive data byte, transmit NACK
		Or clear IFLG, AAK = 1	Receive data byte, transmit ACK
		For a 10-bit address Write extended address byte to DATA, clear IFLG	Transmit extended address byte
48h	Addr + R transmitted, ACK not received	For a 7-bit address: Set STA, clear IFLG	Transmit repeated START
		Or set STP, clear IFLG	Transmit STOP
		Or set STA & STP, clear IFLG	Transmit STOP then START
		For a 10-bit address: Write extended address byte to DATA, clear IFLG	Transmit extended address byte
38h	Arbitration lost	Clear IFLG	Return to idle
		Or set STA, clear IFLG	Transmit START when bus is free
68h	Arbitration lost, SLA+W received, ACK transmitted	Clear IFLG, clear AAK = 0	Receive data byte, transmit NACK
		Or clear IFLG, set AAK = 1	Receive data byte, transmit ACK
78h	Arbitration lost, General call addr received, ACK transmitted	Same as code 68h	Same as code 68h
B0h	Arbitration lost, SLA+R received, ACK transmitted	Write byte to DATA, clear IFLG, clear AAK = 0	Transmit last byte, receive ACK
		Or write byte to DATA, clear IFLG, set AAK = 1	Transmit data byte, receive ACK
R = Read bit; that is, the lsb is set to 1.			

If 10-bit addressing is being used, the slave is first addressed using the full 10-bit address plus the Write bit. The master then issues a restart followed by the first part of the 10-bit address again, but with the Read bit. The status code then becomes 40h or 48h. It is the responsibility of the slave to remember that it had been selected prior to the restart.

If a repeated START condition is received, the status code is 10h instead of 08h.

After each data byte is received, the IFLG is set and one of the status codes listed in Table 79 is in the I2C_SR register.

Table 79. I^2C Master Receive Status Codes For Data Bytes

Code	I^2C State	Microprocessor Response	Next I^2C Action
50h	Data byte received, ACK transmitted	Read DATA, clear IFLG, clear AAK = 0	Receive data byte, transmit NACK
		Or read DATA, clear IFLG, set AAK = 1	Receive data byte, transmit ACK
58h	Data byte received, NACK transmitted	Read DATA, set STA, clear IFLG	Transmit repeated START
		Or read DATA, set STP, clear IFLG	Transmit STOP
		Or read DATA, set STA & STP, clear IFLG	Transmit STOP then START
38h	Arbitration lost in NACK bit	Same as master transmit	Same as master transmit

When all bytes are received, a NACK should be sent, then the microprocessor should write a 1 to the STP bit in the I2C_CTL register. The I^2C then transmits a STOP condition, clears the STP bit and returns to the idle state.

Slave Transmit

In SLAVE TRANSMIT mode, a number of bytes are transmitted to a master receiver.

The I^2C enters SLAVE TRANSMIT mode when it receives its own slave address and a Read bit after a START condition. The I^2C then transmits an acknowledge bit (if the AAK bit is set to 1) and sets the IFLG bit in the I2C_CTL register and the I2C_SR register contains the status code A8h.

Note: When 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 after a restart. An interrupt is generated, IFLG is set but the status does not change. No second address byte is sent by the master. It is up to the slave to remember it had been selected prior to the restart.

I²C goes from MASTER mode to SLAVE TRANSMIT mode when arbitration is lost during the transmission of an address, and the slave address and Read bit are received. This action is represented by the status code `B0h` in the I2C_SR register.

The data byte to be transmitted is loaded into the I2C_DR register and the IFLG bit cleared. After the I²C transmits the byte and receives an acknowledge, the IFLG bit is set and the I2C_SR register contains `B8h`. When the final byte to be transmitted is loaded into the I2C_DR register, the AAK bit is cleared when the IFLG is cleared. After the final byte is transmitted, the IFLG is set and the I2C_SR register contains `C8h` and the I²C 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 I²C then returns to the idle state.

If a STOP condition is detected after an acknowledge bit, the I²C returns to the idle state.

Slave Receive

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

The I²C enters SLAVE RECEIVE mode when it receives its own slave address and a Write bit (lsb = 0) after a START condition. The I²C 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 I²C 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 I²C 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 I²C generates an interrupt only after the second address byte is received. The I²C sets the IFLG bit and loads the status code as described above.

I²C 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.

I^2C Registers

Addressing

The processor interface provides access to six 8-bit registers: four Read/Write registers, one Read-Only register and two Write-Only registers, as indicated in Table 80.

Table 80. I^2C Register Descriptions

Register	Description
I2C_SAR	Slave address register
I2C_XSAR	Extended slave address register
I2C_DR	Data byte register
I2C_CTL	Control register
I2C_SR	Status register (Read-Only)
I2C_CCR	Clock Control register (Write-Only)
I2C_SRR	Software reset register (Write-Only)

Resetting the I^2C Registers

Hardware reset. When the I^2C is reset by a hardware reset of the eZ80L92, the I2C_SAR, I2C_XSAR, I2C_DR and I2C_CTL registers are cleared to `00h`; while the I2C_SR register is set to `F8h`.

Software Reset. Perform a software reset by writing any value to the I^2C Software Reset Register (I2C_SRR). A software reset sets the I^2C back to idle and the STP, STA, and IFLG bits of the I2C_CTL register to 0.

I²C Slave Address Register

The I2C_SAR register provides the 7-bit address of the I²C when in SLAVE mode and allows 10-bit addressing in conjunction with the I2C_XSAR register. I2C_SAR[7:1] = sla[6:0] is the 7-bit address of the I²C when in 7-bit SLAVE mode. When the I²C receives this address after a START condition, it enters SLAVE mode. I2C_SAR[7] corresponds to the first bit received from the I²C bus.

When the register receives an address starting with `F7h` to `F0h` (I2C_SAR[7:3] = 11110b), the I²C recognizes that a 10-bit slave addressing mode is being selected. The I²C sends an ACK after receiving the I2C_SAR byte (the device does not generate an interrupt at this point). After the next byte of the address (I2C_XSAR) is received, the I²C 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 81.

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	I²C not enabled to recognize the General Call Address.
	1	I²C enabled to recognize the General Call Address.

I²C Extended Slave Address Register

The I2C_XSAR register is used in conjunction with the I2C_SAR register to provide 10-bit addressing of the I²C 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 I²C recognizes that a 10-bit slave addressing mode is being selected. The I²C 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 I²C 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. I²C 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
[7:0] SLAX	00h–FFh	Least significant 8 bits of the 10-bit extended slave address.

I²C Data Register

This register contains the data byte/slave address to be transmitted or the data byte just received. In transmit mode, the most significant bit of the byte is transmitted first. In receive mode, the first bit received is placed in the most significant bit of the register. After each byte is transmitted, the I2C_DR register contains the byte that is present on the bus in case a lost arbitration event occurs. See Table 83.

Table 83. I²C Data Register (I2C_DR = 00CAh)

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
[7:0] DATA	00h–FFh	I²C data byte.

I²C Control Register

The I2C_CTL register is a control register that is used to control the interrupts and the master slave relationships on the I²C bus.

When the Interrupt Enable bit (IEN) is set to 1, the interrupt line goes High when the IFLG is set to 1. When IEN is cleared to 0, the interrupt line always remains Low.

When the Bus Enable bit (ENAB) is set to 0, the I^2C bus inputs SCLx and SDAx are ignored and the I^2C module does not respond to any address on the bus. When ENAB is set to 1, the I^2C responds to calls to its slave address and to the general call address if the GCE bit (I2C_SAR[0]) is set to 1.

When the Master Mode Start bit (STA) is set to 1, the I^2C enters MASTER mode and sends a START condition on the bus when the bus is free. If the STA bit is set to 1 when the I^2C module is already in MASTER mode and one or more bytes are transmitted, then a repeated START condition is sent. If the STA bit is set to 1 when the I^2C block is being accessed in SLAVE mode, the I^2C completes the data transfer in SLAVE mode and then enters MASTER mode when the bus is released. The STA bit is automatically cleared after a START condition is set. Writing a 0 to this bit produces no effect.

If the Master Mode Stop bit (STP) is set to 1 in MASTER mode, a STOP condition is transmitted on the I^2C bus. If the STP bit is set to 1 in slave move, the I^2C module operates as if a STOP condition is received, but no STOP condition is transmitted. If both STA and STP bits are set, the I^2C block first transmits the STOP condition (if in MASTER mode) and then transmit the START condition. The STP bit is cleared automatically. Writing a 0 to this bit produces no effect.

The I^2C Interrupt Flag (IFLG) is set to 1 automatically when any of 30 of the possible 31 I^2C states is entered. The only state that does not set the IFLG bit is state F8h. If IFLG is set to 1 and the IEN bit is also set, an interrupt is generated. When IFLG is set by the I^2C, the Low period of the I^2C bus clock line is stretched and the data transfer is suspended. When a 0 is written to IFLG, the interrupt is cleared and the I^2C clock line is released.

When the I^2C Acknowledge bit (AAK) is set to 1, an Acknowledge is sent during the acknowledge clock pulse on the I^2C bus if:

- Either the whole of a 7-bit slave address or the first or second byte of a 10-bit slave address is received
- The general call address is received and the General Call Enable bit in I2C_SAR is set to 1
- A data byte is received while in MASTER or SLAVE modes

When AAK is cleared to 0, a NACK is sent when a data byte is received in MASTER or SLAVE mode. If AAK is cleared to 0 in the Slave Transmitter mode, the byte in the I2C_DR register is assumed to be the final byte. After this byte is transmitted, the I^2C block enter states `C8h`, then returns to the idle state. The I^2C module does not respond to its slave address unless AAK is set. See Table 84.

Table 84. I²C Control Registers (I2C_CTL = 00CBh)

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	R

Note: R/W = Read/Write; R = Read Only.

Bit Position	Value	Description
7 IEN	0	I²C interrupt is disabled.
	1	I²C interrupt is enabled.
6 ENAB	0	The I²C bus (SCL/SDA) is disabled and all inputs are ignored.
	1	The I²C bus (SCL/SDA) is enabled.
5 STA	0	Master mode START condition is sent.
	1	Master mode start-transmit START condition on the bus.
4 STP	0	Master mode STOP condition is sent.
	1	Master mode stop-transmit STOP condition on the bus.
3 IFLG	0	I²C interrupt flag is not set.
	1	I²C interrupt flag is set.
2 AAK	0	Not Acknowledge.
	1	Acknowledge.
[1:0]	00	Reserved.

I²C Status Register

The I2C_SR register is a Read-Only register that contains a 5-bit status code in the five most significant bits: the three least significant bits are always 0. The Read-Only I2C_SR registers share the same I/O addresses as the Write-Only I2C_CCR registers. See Table 85.

Table 85. I²C Status Registers (I2C_SR = 00CCh)

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

Note: R = Read only.

Bit Position	Value	Description
[7:3] STAT	00000–11111	5-bit I²C status code.
[2:0]	000	Reserved.

There are 29 possible status codes, as listed in Table 86. When the I2C_SR register contains the status code `F8h`, no relevant status information is available, no interrupt is generated and the IFLG bit in the I2C_CTL register is not set. All other status codes correspond to a defined state of the I²C.

When each of these states is entered, the corresponding status code appears in this register and the IFLG bit in the I2C_CTL register is set. When the IFLG bit is cleared, the status code returns to `F8h`.

Table 86. I²C Status Codes

Code	Status
00h	Bus error
08h	START condition transmitted
10h	Repeated START condition transmitted
18h	Address and Write bit transmitted, ACK received
20h	Address and Write bit transmitted, ACK not received
28h	Data byte transmitted in MASTER mode, ACK received
30h	Data byte transmitted in MASTER mode, ACK not received
38h	Arbitration lost in address or data byte

Table 86. I^2C Status Codes (Continued)

Code	Status
40h	Address and Read bit transmitted, ACK received
48h	Address and Read bit transmitted, ACK not received
50h	Data byte received in MASTER mode, ACK transmitted
58h	Data byte received in MASTER mode, NACK transmitted
60h	Slave address and Write bit received, ACK transmitted
68h	Arbitration lost in address as master, slave address and Write bit received, ACK transmitted
70h	General Call address received, ACK transmitted
78h	Arbitration lost in address as master, General Call address received, ACK transmitted
80h	Data byte received after slave address received, ACK transmitted
88h	Data byte received after slave address received, NACK transmitted
90h	Data byte received after General Call received, ACK transmitted
98h	Data byte received after General Call received, NACK transmitted
A0h	STOP or repeated START condition received in SLAVE mode
A8h	Slave address and Read bit received, ACK transmitted
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 I^2C 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 I^2C then returns to the idle state. No STOP condition is transmitted on the I^2C bus.

Note: The STP and STA bits may be set to 1 at the same time to recover from the bus error. The I^2C then sends a START.

I^2C Clock Control Register

The I2C_CCR register is a Write-Only register. The seven LSBs control the frequency at which the I^2C bus is sampled and the frequency of the I^2C clock line (SCL) when the I^2C 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. I^2C 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.
[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].

I^2C Software Reset Register

The I2C_SRR register is a Write-Only register. Writing any value to this register performs a software reset of the I^2C module. See Table 88.

Table 88. I^2C Software Reset Register (I2C_SRR = 00CDh)

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: W = Write-Only.

Bit Position	Value	Description
[7:0] SRR	00h–FFh	Writing any value to this register performs a software reset of the I^2C module.

ZiLOG Debug Interface

Introduction

The ZiLOG Debug Interface (ZDI) provides a built-in debugging interface to the eZ80® CPU. ZDI provides basic in-circuit emulation features including:

- Examining and modifying internal registers
- Examining and modifying memory
- Starting and stopping the user program
- Setting program and data BREAK points
- Single-stepping the user program
- Executing user-supplied instructions
- Debugging the final product with the inclusion of one small connector
- Downloading code into SRAM
- C source-level debugging using ZiLOG Developer Studio (ZDS)

The above features are built into the silicon. Control is provided via a two-wire interface that is connected to the ZPAK emulator. Figure 35 illustrates a typical setup using a a target board, ZPAK, and the host PC running ZiLOG Developer Studio. Refer to the ZiLOG website for more information on ZPAK and ZDS.

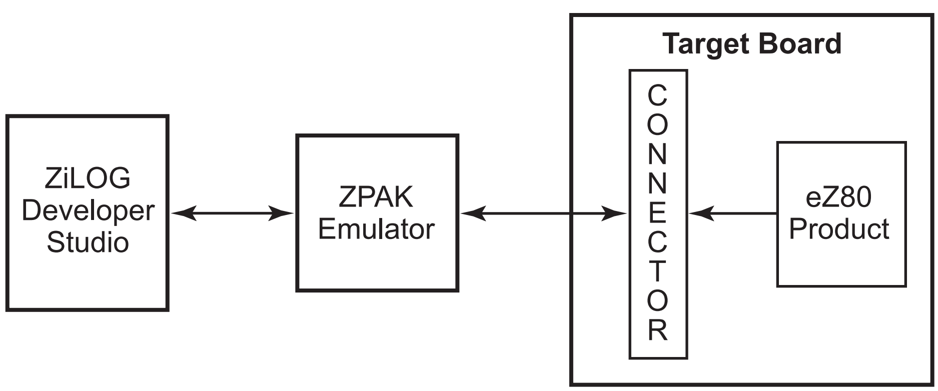


Figure 35. Typical ZDI Debug Setup

ZDI allows reading and writing of most internal registers without disturbing the state of the machine. READs and Writes to memory may occur as fast as the ZDI can download and upload data, with a maximum frequency of one-half the eZ80L92 system clock frequency. Table 89 lists the recommended frequencies of the ZDI clock in relation to the system clock.

Table 89. Recommended ZDI Clock vs. System Clock Frequency

System Clock Frequency	ZDI Clock Frequency
3–10 Mhz	1 Mhz
8–16 Mhz	2 Mhz
12–24 Mhz	4 Mhz
20–50 Mhz	8 Mhz

ZDI-Supported Protocol

ZDI supports a bidirectional serial protocol. The protocol defines any device that sends data as the *transmitter* and any receiving device as the *receiver*. The device controlling the transfer is the *master* and the device being controlled is the *slave*. The master always initiates the data transfers and provides the clock for both receive and transmit operations. The ZDI block on the eZ80L92 is considered a slave in all data transfers.

Figure 36 illustrates the schematic for building a connector on a target board. This connector allows the user to connect directly to the ZPAK emulator using a six-pin header.

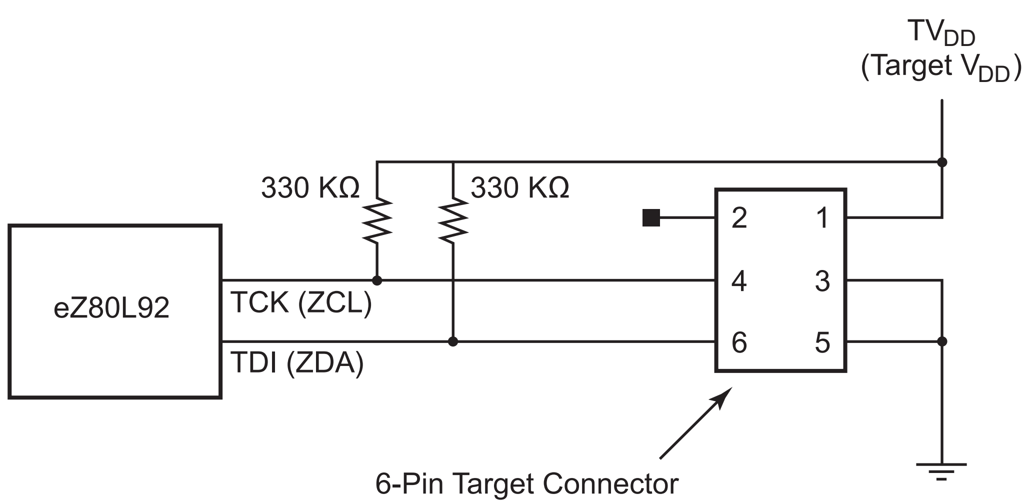


Figure 36. Schematic For Building a Target Board ZPAK Connector

ZDI Clock and Data Conventions

The two pins used for communication with the ZDI block are the ZDI Clock pin (ZCL) and the ZDI Data pin (ZDA). On the eZ80L92, the ZCL pin is shared with the TCK pin while the ZDA pin is shared with the TDI pin. The ZCL and ZDA pin functions are only available when the On-Chip Instrumentation is disabled and the ZDI is therefore enabled. For general data communication, the data value on the ZDA pin can change only when ZCL is Low (0). The only exception is the ZDI START bit, which is indicated by a High-to-Low transition (falling edge) on the ZDA pin while ZCL is High.

Data is shifted into and out of ZDI, with the most significant bit (bit 7) of each byte being first in time, and the least significant bit (bit 0) last in time. All information is passed between the master and the slave in 8-bit (single-byte) units. Each byte is transferred with nine clock cycles: eight to shift the data, and the ninth for internal operations.

ZDI START Condition

All ZDI commands are preceded by the ZDI START signal, which is a High-to-Low transition of ZDA when ZCL is High. The ZDI slave on the eZ80L92 continually monitors the ZDA and ZCL lines for the START signal and does not respond to any command until this condition is met. The master pulls ZDA Low, with ZCL High, to indicate the beginning of a data transfer with the ZDI block. Figures 37 and 38 illustrate a valid ZDI START signal prior to writing and reading data, respectively. A Low-to-High transition of ZDA while the ZCL is High produces no effect.

Data is shifted in during a Write to the ZDI block on the rising edge of ZCL, as illustrated in Figure 37. Data is shifted out during a Read from the ZDI block on the falling edge of ZCL as illustrated in Figure 38. When an operation is completed, the master stops during the ninth cycle and holds the ZCL signal High.

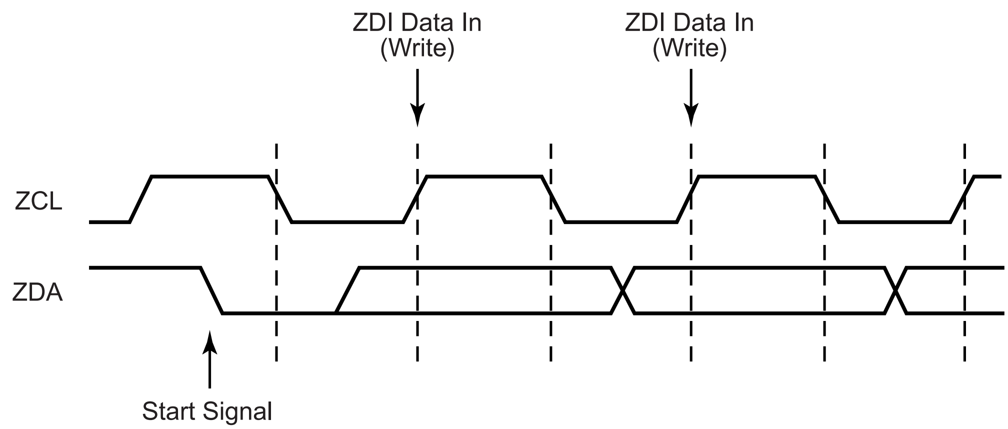


Figure 37. ZDI Write Timing

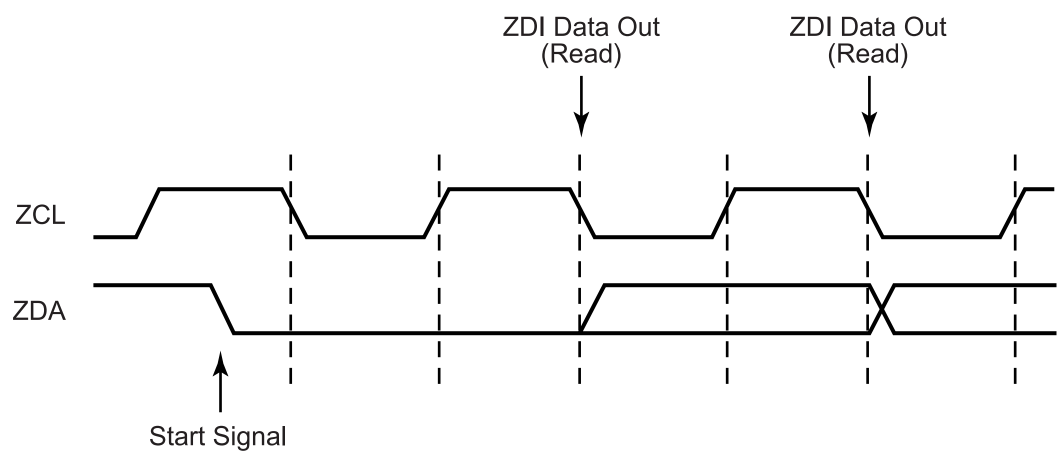


Figure 38. ZDI Read Timing

ZDI Single-Bit Byte Separator

Following each 8-bit ZDI data transfer, a single-bit byte separator is used. To initiate a new ZDI command, the single-bit byte separator must be High (logical 1) to allow for a new ZDI START command to be sent. For all other cases, the single-bit byte separator can be either Low (logical 0) or High (logical 1). When ZDI is configured to allow the CPU to accept external bus requests, the single-bit byte separator should be Low (logical 0) during all ZDI commands. This Low value indicates that ZDI is still operating and is not ready to relinquish the Bus. The CPU does not accept the external bus requests until the single-bit byte separator is a High (logical 1). For more information on accepting bus requests in ZDI DEBUG mode, please see the Bus Requests During ZDI Debug Mode section on page 163.

ZDI Register Addressing

Following a START signal the ZDI master must output the ZDI register address. All data transfers with the ZDI block use special ZDI registers. The ZDI control registers that reside in the ZDI register address space should not be confused with the eZ80L92 peripheral registers that reside in the I/O address space.

Many locations in the ZDI control register address space are shared by two registers, one for Read-Only access and one for Write-Only access. As an example, a Read from ZDI register address `00h` returns the eZ80® Product ID Low Byte while a Write to this same location, `00h`, stores the Low byte of one of the address match values used for generating BREAK points.

The format for a ZDI address is seven bits of address, followed by one bit for Read or WRITE control, and completed by a single-bit byte separator. The ZDI executes a Read or WRITE operation depending on the state of the R/W bit (0 = Write, 1 = Read). If no new START command is issued at completion of the Read or Write operation, the operation can be repeated. This allows repeated Read or WRITE operations without having to resend the ZDI command. A START signal must follow to initiate a new ZDI command. Figure 39 illustrates the timing for address Writes to ZDI registers.

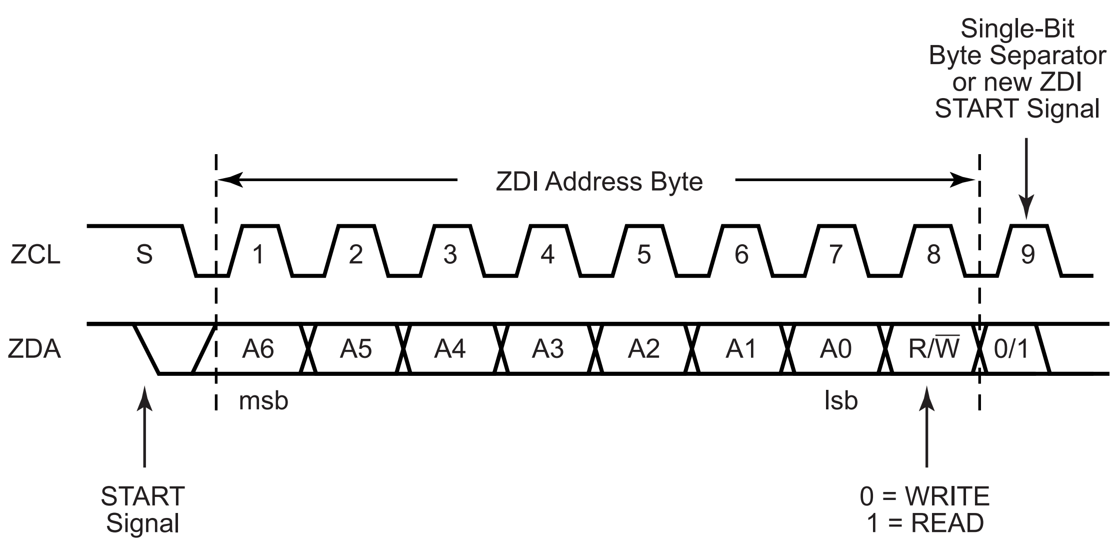


Figure 39. ZDI Address Write Timing

ZDI Write Operations

ZDI SINGLE-BYTE Write

For SINGLE-BYTE Write operations, the address and write control bit are first written to the ZDI block. Following the single-bit byte separator, the data is shifted into the ZDI block on the next 8 rising edges of ZCL. The master terminates activity after 8 clock cycles.Figure 40 illustrates the timing for ZDI SINGLE-BYTE WRITE operations.

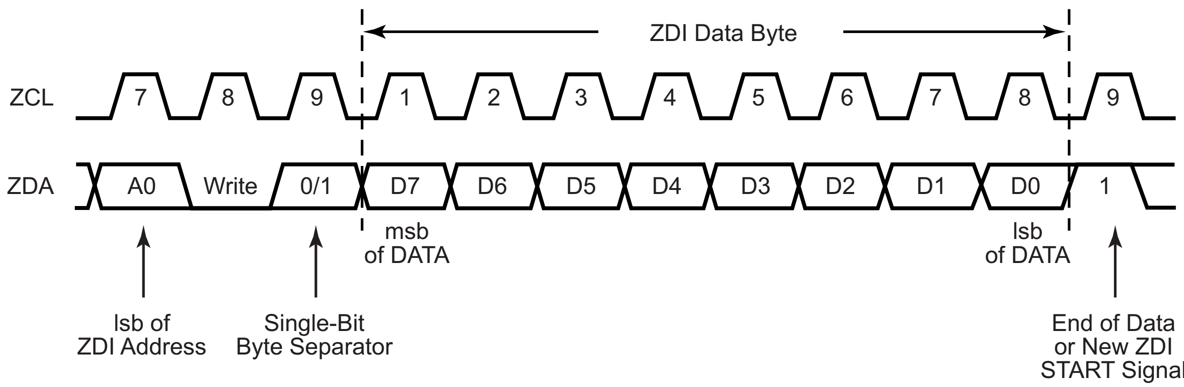


Figure 40. ZDI Single-Byte Data Write Timing

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.

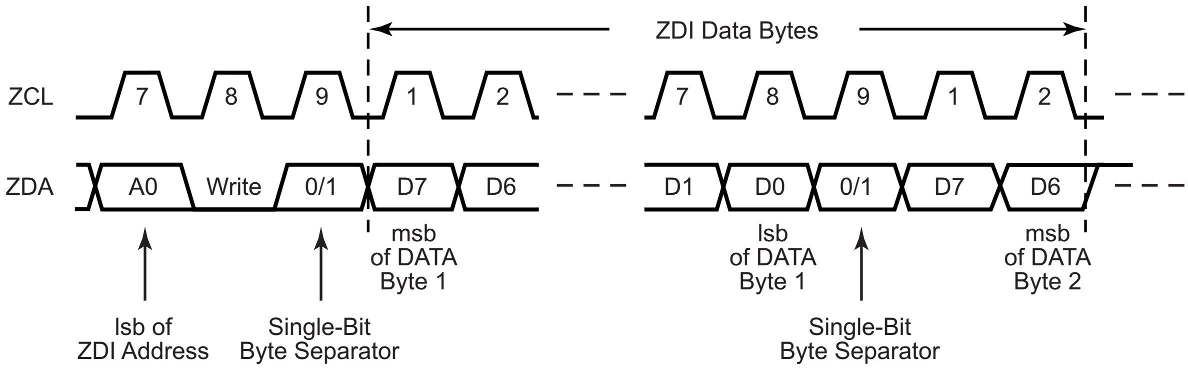


Figure 41. ZDI Block Data Write Timing

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/W bit of the ZDI register address is set to 1. Upon receipt of a slave address with the R/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.

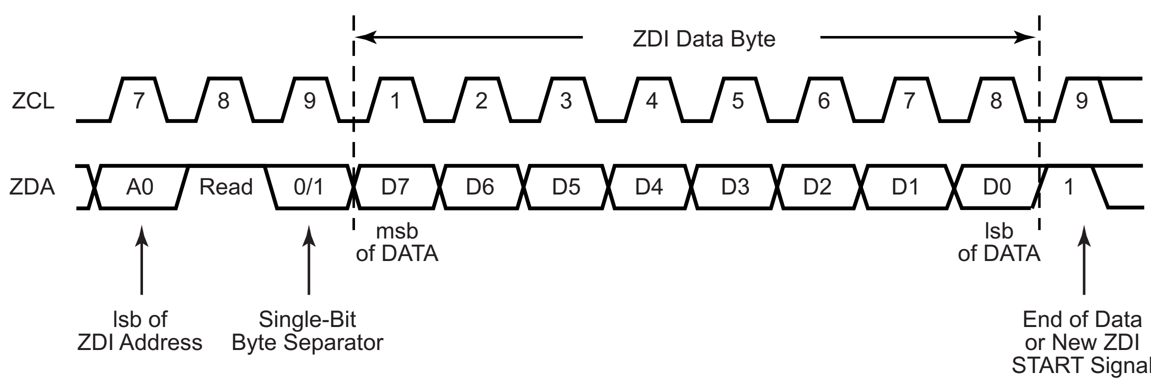


Figure 42. ZDI Single-Byte Data Read Timing

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.

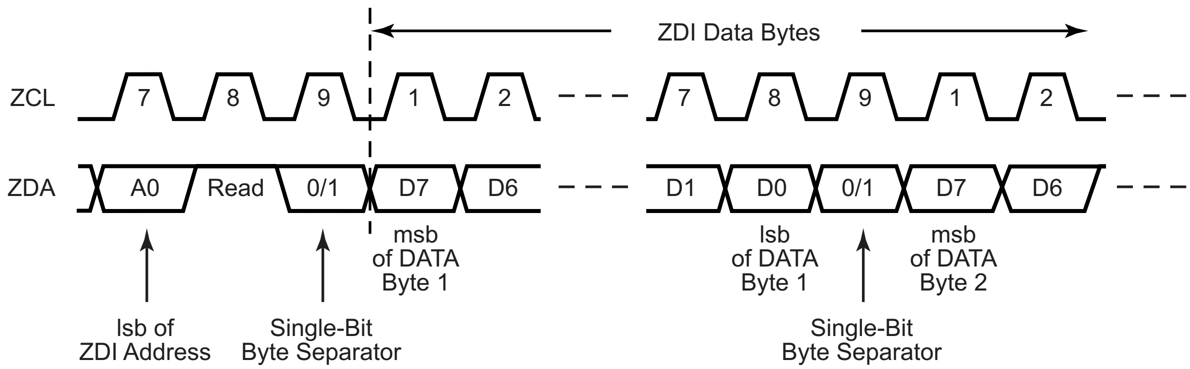


Figure 43. ZDI Block Data Read Timing

Operation of the eZ80L92 during ZDI BREAKpoints

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 Watch-Dog 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 (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 (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 BUSACK to allow additional data Read or Write operations. If the bit is a logical 1, indicating completion of the ZDI commands, 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 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 BUSACK assertion, may produce unpredictable results.

ZDI Write-Only Registers

Table 90 lists the ZDI Write-Only registers. Many of the ZDI Write-Only addresses are shared with ZDI Read-Only registers.

Table 90. ZDI Write-Only Registers

ZDI Address	ZDI Register Name	ZDI Register Function	Reset Value
00h	ZDI_ADDR0_L	Address Match 0 Low Byte	XXh
01h	ZDI_ADDR0_H	Address Match 0 High Byte	XXh
02h	ZDI_ADDR0_U	Address Match 0 Upper Byte	XXh
04h	ZDI_ADDR1_L	Address Match 1 Low Byte	XXh
05h	ZDI_ADDR1_H	Address Match 1 High Byte	XXh

Table 90. ZDI Write-Only Registers (Continued)

ZDI Address	ZDI Register Name	ZDI Register Function	Reset Value
06h	ZDI_ADDR1_U	Address Match 1 Upper Byte	XXh
08h	ZDI_ADDR2_L	Address Match 2 Low Byte	XXh
09h	ZDI_ADDR2_H	Address Match 2 High Byte	XXh
0Ah	ZDI_ADDR2_U	Address Match 2 Upper Byte	XXh
0Ch	ZDI_ADDR3_L	Address Match 3 Low Byte	XXh
0Dh	ZDI_ADDR3_H	Address Match 3 High Byte	XXh
0Eh	ZDI_ADDR3_U	Address Match 4 Upper Byte	XXh
10h	ZDI_BRK_CTL	BREAK Control register	00h
11h	ZDI_MASTER_CTL	Master Control register	00h
13h	ZDI_WR_DATA_L	Write Data Low Byte	XXh
14h	ZDI_WR_DATA_H	Write Data High Byte	XXh
15h	ZDI_WR_DATA_U	Write Data Upper Byte	XXh
16h	ZDI_RW_CTL	Read/Write Control register	00h
17h	ZDI_BUS_CTL	Bus Control register	00h
21h	ZDI_IS4	Instruction Store 4	XXh
22h	ZDI_IS3	Instruction Store 3	XXh
23h	ZDI_IS2	Instruction Store 2	XXh
24h	ZDI_IS1	Instruction Store 1	XXh
25h	ZDI_IS0	Instruction Store 0	XXh
30h	ZDI_WR_MEM	Write Memory register	XXh

ZDI Read-Only Registers

Table 91 lists the ZDI Read-Only registers. Many of the ZDI Read-Only addresses are shared with ZDI Write-Only registers.

Table 91. ZDI Read-Only Registers

ZDI Address	ZDI Register Name	ZDI Register Function	Reset Value
00h	ZDI_ID_L	eZ80® Product ID Low Byte register	06h
01h	ZDI_ID_H	eZ80® Product ID High Byte register	00h
02h	ZDI_ID_REV	eZ80® Product ID Revision register	XXh
03h	ZDI_STAT	Status register	00h
10h	ZDI_RD_L	Read Memory Address Low Byte register	XXh
11h	ZDI_RD_H	Read Memory Address High Byte register	XXh
12h	ZDI_RD_U	Read Memory Address Upper Byte register	XXh
17h	ZDI_BUS_STAT	Bus Status register	00h
20h	ZDI_RD_MEM	Read Memory Data Value	XXh

ZDI Register Definitions

ZDI Address Match Registers

The four sets of address match registers are used for setting the addresses for generating BREAK points. When the accompanying BRK_ADDRx bit is set in the ZDI BREAK Control register to enable the particular address match, the current eZ80L92 address is compared with the 3-byte address set, {ZDI_ADDRx_U, ZDI_ADDRx_H, ZDI_ADDR_x_L}. If the CPU is operating in ADL mode, the address is supplied by ADDR[23:0]. If the CPU is operating in Z80 mode, the address is supplied by {MBASE[7:0], ADDR[15:0]}. If a match is found, ZDI issues a BREAK to the eZ80L92 placing the processor in ZDI mode pending further instructions from the ZDI interface block. If the address is not the first op-code fetch, the ZDI BREAK is executed at the end of the instruction in which it is executed. There are four sets of address match registers. They can be used in conjunction with each other to BREAK on branching instructions. See Table 92.

Table 92. ZDI Address Match Registers
ZDI_ADDR0_L = 00h, ZDI_ADDR0_H = 01h, ZDI_ADDR0_U = 02h,
ZDI_ADDR1_L = 04h, ZDI_ADDR1_H = 05h, ZDI_ADDR1_U = 06h,
ZDI_ADDR2_L = 08h, ZDI_ADDR2_H = 09h, ZDI_ADDR2_U = 0Ah,
ZDI_ADDR3_L = 0Ch, ZDI_ADDR3_H = 0Dh, and ZDI_ADDR3_U = 0Eh
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: W = Write-only.

Bit Position	Value	Description
[7:0] ZDI_ADDR*x*_L, ZDI_ADDR*x*_H, or ZDI_ADDR*x*_U	00h–FFh	The four sets of ZDI address match registers are used for setting the addresses for generating BREAK points. The 24-bit addresses are supplied by {ZDI_ADDRx_U, ZDI_ADDRx_H, ZDI_ADDRx_L, where *x* is 0, 1, 2, or 3.

ZDI BREAK Control Register

The ZDI BREAK Control register is used to enable BREAK points. ZDI asserts a BREAK when the CPU instruction address, ADDR[23:0], matches the value in the ZDI Address Match 3 registers, {ZDI_ADDR3_U, ZDI_ADDR3_H, ZDI_ADDR3_L}. BREAKs can only occur on an instruction boundary. If the instruction address is not the beginning of an instruction (that is, for multibyte instructions), then the BREAK occurs at the end of the current instruction. The BRK_NEXT bit is set to 1. The BRK_NEXT bit must be reset to 0 to release the BREAK. See Table 93.

Table 93. ZDI BREAK Control Register (ZDI_BRK_CTL = 10h in the ZDI Write-Only Register Address Space)

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 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.

Bit Position	Value	Description
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.
1 IGN_LOW_0	0	The *Ignore the Low Byte* function of the ZDI Address Match 1 registers is disabled. If BRK_ADDR0 is set to 1, ZDI initiates a BREAK when the entire 24-bit address, ADDR[23:0], matches the 3-byte value {ZDI_ADDR0_U, ZDI_ADDR0_H, ZDI_ADDR0_L}.
	1	The *Ignore the Low Byte* function of the ZDI Address Match 1 registers is enabled. If the BRK_ADDR1 is set to 0, ZDI initiates a BREAK when only the upper 2 bytes of the 24-bit address, ADDR[23:8], match the 2 bytes value {ZDI_ADDR0_U, ZDI_ADDR0_H}. As a result, a BREAK can occur anywhere within a 256-byte page.
0 SINGLE_STEP	0	ZDI SINGLE STEP mode is disabled.
	1	ZDI SINGLE STEP mode is enabled. ZDI asserts a BREAK following execution of each instruction.

ZDI Master Control Register

The ZDI Master Control register provides control of the eZ80L92. It is capable of forcing a RESET and waking up the eZ80L92 from the low-power modes (HALT or SLEEP). See Table 94.

Table 94. ZDI Master Control Register (ZDI_MASTER_CTL = 11h in ZDI Register Write Address Spaces)

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 ZDI_RESET	0	No action.
	1	Initiate a RESET of the eZ80L92. This bit is automatically cleared at the end of the RESET event.
[6:0]	0000000	Reserved.

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. Refer to the eZ80 CPU User Manual (UM0077) for information regarding the CPU registers.

Table 96. ZDI Read/Write Control Register Functions (ZDI_RW_CTL = 16h in the ZDI Register Write-Only Address Space)

Hex Value	Command	Hex Value	Command
00	Read {MBASE, A, F} ZDI_RD_U ← MBASE ZDI_RD_H ← F ZDI_RD_L ← A	80	Write {MBASE, A, F} MBASE ← ZDI_WR_U F ← ZDI_WR_H A ← ZDI_WR_L
01	Read BC ZDI_RD_U ← BCU ZDI_RD_H ← B ZDI_RD_L ← C	81	Write BC BCU ← ZDI_WR_U B ← ZDI_WR_H C ← ZDI_WR_L
02	Read DE ZDI_RD_U ← DEU ZDI_RD_H ← D ZDI_RD_L ← E	82	Write DE DEU ← ZDI_WR_U D ← ZDI_WR_H E ← ZDI_WR_L
03	Read HL ZDI_RD_U ← HLU ZDI_RD_H ← H ZDI_RD_L ← L	83	Write HL HLU ← ZDI_WR_U H ← ZDI_WR_H L ← ZDI_WR_L
04	Read IX ZDI_RD_U ← IXU ZDI_RD_H ← IXH ZDI_RD_L ← IXL	84	Write IX IXU ← ZDI_WR_U IXH ← ZDI_WR_H IXL ← ZDI_WR_L
05	Read IY ZDI_RD_U ← IYU ZDI_RD_H ← IYH ZDI_RD_L ← IYL	85	Write IY IYU ← ZDI_WR_U IYH ← ZDI_WR_H IYL ← ZDI_WR_L
06	Read SP In ADL mode, SP = SPL. In Z80 mode, SP = SPS.	86	Write SP In ADL mode, SP = SPL. In Z80 mode, SP = SPS.
07	Read PC ZDI_RD_U ← PC[23:16] ZDI_RD_H ← PC[15:8] ZDI_RD_L ← PC[7:0]	87	Write PC PC[23:16] ← ZDI_WR_U PC[15:8] ← ZDI_WR_H PC[7:0] ← ZDI_WR_L
08	Set ADL ADL ← 1	88	Reserved

Note: The eZ80® CPU's alternate register set (A', F', B', C', D', E', HL') cannot be read directly. The ZDI programmer must execute the exchange instruction (EXX) to gain access to the alternate eZ80® CPU register set.

Table 96. ZDI Read/Write Control Register Functions (ZDI_RW_CTL = 16h in the ZDI Register Write-Only Address Space) (Continued)

Hex Value	Command	Hex Value	Command
09	Reset ADL ADL ← 0	89	Reserved
0A	Exchange CPU register sets AF ← AF' BC ← BC' DE ← DE' HL ← HL'	8A	Reserved
0B	Read memory from current PC value, increment PC	8B	Write memory from current PC value, increment PC

Note: The eZ80® CPU's alternate register set (A', F', B', C', D', E', HL') cannot be read directly. The ZDI programmer must execute the exchange instruction (EXX) to gain access to the alternate eZ80® CPU register set.

ZDI Bus Control Register

The ZDI Bus Control register controls bus requests during DEBUG mode. It enables or disables bus acknowledge in ZDI DEBUG mode and allows ZDI to force assertion of the BUSACK signal. This register should only be written during ZDI Debug mode (that is, following a BREAK). See Table 97.

Table 97. ZDI Bus Control Register (ZDI_BUS_CTL = 17h in the ZDI Register Write-Only Address Space)

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 ZDI_BUSAK_EN	0	Bus requests by external peripherals using the $\overline{BUSREQ}$ pin are ignored. The bus acknowledge signal, BUSACK, is not asserted in response to any bus requests.
	1	Bus requests by external peripherals using the $\overline{BUSREQ}$ pin are accepted. A bus acknowledge occurs at the end of the current ZDI operation. The bus acknowledge is indicated by asserting the BUSACK pin in response to a bus request.

Bit Position	Value	Description
6 ZDI_BUSAK	0	Deassert the bus acknowledge pin ($\overline{BUSACK}$) to return control of the address and data buses back to ZDI.
	1	Assert the bus acknowledge pin ($\overline{BUSACK}$) to pass control of the address and data buses to an external peripheral.
[5:0]	000000	Reserved.

Instruction Store 4:0 Registers

The ZDI Instruction Store registers are located in the ZDI Register Write-Only address space. They can be written with instruction data for direct execution by the CPU. When the ZDI_IS0 register is written, the eZ80L92 exits the ZDI BREAK state and executes a single instruction. The Op Codes and operands for the instruction come from these Instruction Store registers. The Instruction Store Register 0 is the first byte fetched, followed by Instruction Store registers 1, 2, 3, and 4, as necessary. Only the bytes the processor requires to execute the instruction must be stored in these registers. Some eZ80® instructions, when combined with the MEMORY mode suffixes (.SIS, .SIL, .LIS, or .LIL), require 6 bytes to operate. These 6-byte instructions cannot be executed directly using the ZDI Instruction Store registers. See Table 98.

Note: The Instruction Store 0 register is located at a higher ZDI address than the other Instruction Store registers. This feature allows the use of the ZDI auto-address increment function to load and execute a multibyte instruction with a single data stream from the ZDI master. Execution of the instruction commences with writing the final byte to ZDI_IS0.

Table 98. Instruction Store 4:0 Registers (ZDI_IS4 = 21h, ZDI_IS3 = 22h, ZDI_IS2 = 23h, ZDI_IS1 = 24h, and ZDI_IS0 = 25h 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_IS4, ZDI_IS3, ZDI_IS2, ZDI_IS1, or ZDI_IS0	00h–FFh	These registers contain the Op Codes and operands for immediate execution by the CPU following a Write to ZDI_IS0. The ZDI_IS0 register contains the first Op Code of the instruction. The remaining ZDI_ISx registers contain any additional Op Codes or operand dates required for execution of the required instruction.

ZDI Write Memory Register

A Write to the ZDI Write Memory register causes the eZ80L92 to write the 8-bit data to the memory location specified by the current address in the program counter. In Z80 MEMORY mode, this address is {MBASE, PC[15:0]}. In ADL MEMORY mode, this address is PC[23:0]. The program counter, PC, increments after each data Write. However, the ZDI register address does not increment automatically when this register is accessed. As a result, the ZDI master is allowed to write any number of data bytes by writing to this address one time followed by any number of data bytes. See Table 99.

Table 99. ZDI Write Memory Register (ZDI_WR_MEM = 30h 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_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.

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

eZ80® Product ID Revision Register

The eZ80® Product ID Revision register identifies the current revision of the eZ80L92 product. See Table 102.

Table 102. eZ80® Product ID Revision Register (ZDI_ID_REV = 02h in the ZDI Register Read-Only Address Space)

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: X = Undetermined; R = Read-only.

Bit Position	Value	Description
[7:0] ZDI_ID_REV	00h–FFh	Identifies the current revision of the eZ80L92 product.

ZDI Status Register

The ZDI Status register provides current information on the eZ80L92 and the eZ80® CPU. See Table 103.

Table 103. ZDI Status Register
(ZDI_STAT = 03h 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.								

Bit Position	Value	Description
7 ZDI_ACTIVE	0	The CPU is not functioning in ZDI mode.
	1	The CPU is currently functioning in ZDI mode.
6	0	Reserved.
5 HALT_SLP	0	eZ80L92 is not currently in HALT or SLEEP mode.
	1	eZ80L92 is currently in HALT or SLEEP mode.
4 ADL	0	The CPU is operating in Z80 MEMORY mode. (ADL bit = 0).
	1	The CPU is operating in ADL MEMORY mode. (ADL bit = 1).
3 MADL	0	The CPU’s Mixed-Memory mode (MADL) bit is reset to 0.
	1	The CPU’s Mixed-Memory mode (MADL) bit is set to 1.
2 IEF1	0	The CPU’s Interrupt Enable Flag 1 is reset to 0. Maskable interrupts are disabled.
	1	The CPU’s Interrupt Enable Flag 1 is set to 1. Maskable interrupts are enabled.
[1:0] RESERVED	00	Reserved.

ZDI Read Registers—Low, High, and Upper

The ZDI register Read-Only address space offers Low, High, and Upper functions, which contain the value read by a Read operation from the ZDI Read/Write Control register (ZDI_RW_CTL). This data is valid only while in ZDI BREAK mode and only if the instruction is read by a request from the ZDI Read/Write Control register. See Table 104.

Table 104. ZDI Read Registers—Low, High and Upper (ZDI_RD_L = 10h, ZDI_RD_H = 11h, and ZDI_RD_U = 12h 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.

Bit Position	Value	Description
[7:0] ZDI_RD_L, ZDI_RD_H, or ZDI_RD_U	00h–FFh	Values read from the memory location as requested by the ZDI Read Control register during a ZDI Read operation. The 24-bit value is supplied by {ZDI_RD_U, ZDI_RD_H, ZDI_RD_L}.

ZDI Bus Status Register

The ZDI Bus Status register monitors BUSACKs during DEBUG mode. See Table 105.

Table 105. ZDI Bus Control Register (ZDI_BUS_STAT = 17h 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.

Bit Position	Value	Description
7 ZDI_BUSACK_EN	0	Bus requests by external peripherals using the BUSREQ pin are ignored. The bus acknowledge signal, BUSACK, is not asserted.
	1	Bus requests by external peripherals using the BUSREQ pin are accepted. A bus acknowledge occurs at the end of the current ZDI operation. The bus acknowledge is indicated by asserting the BUSACK pin.
6 ZDI_BUS_STAT	0	Address and data buses are not relinquished to an external peripheral. bus acknowledge is deasserted (BUSACK pin is High).
	1	Address and data buses are relinquished to an external peripheral. bus acknowledge is asserted (BUSACK pin is Low).
[5:0]	000000	Reserved.

ZDI Read Memory Register

When a Read is executed from the ZDI Read Memory register, the eZ80L92 fetches the data from the memory address currently pointed to by the program counter, PC; the program counter is then incremented. In Z80 MEMORY mode, the memory address is {MBASE, PC[15:0]}. In ADL MEMORY mode, the memory address is PC[23:0]. Refer to the eZ80 CPU User Manual (UM0077) for more information regarding Z80 and ADL MEMORY modes. The program counter, PC, increments after each data Read. However, the ZDI register address does not increment automatically when this register is accessed. As a result, the ZDI master can read any number of data bytes out of memory through the ZDI Read Memory register. See Table 106.

Table 106. ZDI Read Memory Register (ZDI_RD_MEM = 20h 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.

Bit Position	Value	Description
[7:0] ZDI_RD_MEM	00h–FFh	8-bit data read from the memory address indicated by the CPU's program counter. In Z80 MEMORY mode, 8-bit data is transferred out from address {MBASE, PC[15:0]}. In ADL Memory mode, 8-bit data is transferred out from address PC[23:0].

On-Chip Instrumentation

Introduction to On-Chip Instrumentation

On-Chip Instrumentation[1] (OCI™) for the ZiLOG eZ80® CPU core enables powerful debugging features. The OCI provides run control, memory and register visibility, complex breakpoints, and trace history features.

The OCI employs all of the functions of the ZiLOG Debug Interface (ZDI) as described in the ZDI section. It also adds the following debug features:

- Control via a 4-pin JTAG port that conforms to IEEE Standard 1149.1 (Test Access Port and Boundary-Scan Architecture)[2]
- Complex breakpoint trigger functions
- Breakpoint enhancements, such as the ability to:
  - Define two breakpoint addresses that form a range
  - Break on masked data values
  - Start or stop trace
  - Assert a trigger output signal
- Trace history buffer
- Software breakpoint instruction

There are four sections to the OCI:

1. JTAG interface
2. ZDI debug control
3. Trace buffer memory
4. Complex triggers

OCI Activation

OCI features clock initialization circuitry so that external debug hardware can be detected during power up. The external debugger must drive the OCI clock pin (TCK) Low at least two system clock cycles prior to the end of the RESET to activate the OCI block. If TCK is High at the end of the RESET, the OCI block shuts down so that it does not draw power in normal product operation. When the OCI is

1. On-Chip Instrumentation and OCI are trademarks of First Silicon Solutions, Inc.
2. The eZ80L92 does not contain the boundary scan register required for 1149.1 compliance.

shut down, ZDI is enabled directly and can be accessed through the clock (TCK) and data (TDI) pins. See the ZiLOG Debug Interface section on page 156 for more information on ZDI.

OCI Interface

There are five dedicated pins on the eZ80L92 for the OCI interface. Four (TCK, TMS, TDI, and TDO) are required for IEEE Standard 1149.1-compatible JTAG ports. The TRIGOUT pin provides additional testability features. These five OCI pins are described in Table 107.

Table 107. OCI Pins

Symbol	Name	Type	Description
TCK	Clock.	Input	Asynchronous to the primary eZ80L92 system clock. The TCK period but must be at least twice the system clock period. During RESET, this pin is sampled to select either OCI or ZDI DEBUG modes. If Low during RESET, the OCI is enabled. If High during RESET, the OCI is powered down and ZDI DEBUG mode is enabled. When ZDI DEBUG mode is active, this pin is the ZDI clock. On-chip pull-up ensures a default value of 1 (High).
TMS	Test Mode Select	Input	This serial test mode input controls JTAG mode selection. On-chip pull-up ensures a default value of 1 (High). The TMS signal is sampled on the rising edge of the TCK signal.
TDI	Data In	Input (OCI enabled)	Serial test data input. On-chip pull-up ensures a default value of 1 (High). This pin is input-only when the OCI is enabled. The input data is sampled on the rising edge of the TCK signal.
		I/O (OCI disabled)	When the OCI is disabled, this pin functions as the ZDA (ZDI Data) I/O pin.
TDO	Data Out	Output	The output data changes on the falling edge of the TCK signal.
TRIGOUT	Trigger Output	Output	Generates an active High trigger pulse when valid OCI trigger events occur. Output is tristate when no data is being driven out.

OCI Information Requests

For additional information regarding On-Chip Instrumentation, or to order OCI debug tools, please contact:

First Silicon Solutions, Inc.
5440 SW Westgate Drive, Suite 240
Portland, OR 97221
Phone: (503) 292-6730
Fax: (503) 292-5840
www.fs2.com

eZ80® CPU Instruction Set

Tables 108 through 117 indicate the eZ80® CPU instructions available for use with the eZ80L92. The instructions are grouped by class. More detailed information is available in the eZ80® CPU User Manual.

Table 108. Arithmetic Instructions

Mnemonic	Instruction
ADC	Add with Carry
ADD	Add without Carry
CP	Compare with Accumulator
DAA	Decimal Adjust Accumulator
DEC	Decrement
INC	Increment
MLT	Multiply
NEG	Negate Accumulator
SBC	Subtract with Carry
SUB	Subtract without Carry

Table 109. Bit Manipulation Instructions

Mnemonic	Instruction
BIT	Bit Test
RES	Reset Bit
SET	Set Bit

Table 110. Block Transfer and Compare Instructions

Mnemonic	Instruction
CPD (CPDR)	Compare and Decrement (with Repeat)
CPI (CPIR)	Compare and Increment (with Repeat)
LDD (LDDR)	Load and Decrement (with Repeat)
LDI (LDIR)	Load and Increment (with Repeat)

Table 111. Exchange Instructions

Mnemonic	Instruction
EX	Exchange registers
EXX	Exchange CPU Multibyte register banks

Table 112. Input/Output Instructions

Mnemonic	Instruction
IN	Input from I/O
IN0	Input from I/O on Page 0
IND (INDR)	Input from I/O and Decrement (with Repeat)
INDRX	Input from I/O and Decrement Memory Address with Stationary I/O Address
IND2 (IND2R)	Input from I/O and Decrement (with Repeat)
INDM (INDMR)	Input from I/O and Decrement (with Repeat)
INI (INIR)	Input from I/O and Increment (with Repeat)
INIRX	Input from I/O and Increment Memory Address with Stationary I/O Address
INI2 (INI2R)	Input from I/O and Increment (with Repeat)
INIM (INIMR)	Input from I/O and Increment (with Repeat)
OTDM (OTDMR)	Output to I/O and Decrement (with Repeat)
OTDRX	Output to I/O and Decrement Memory Address with Stationary I/O Address
OTIM (OTIMR)	Output to I/O and Increment (with Repeat)
OTIRX	Output to I/O and Increment Memory Address with Stationary I/O Address
OUT	Output to I/O
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 113. Load Instructions

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

Table 114. Logical Instructions

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

Table 115. Processor Control Instructions

Mnemonic	Instruction
CCF	Complement Carry Flag
DI	Disable Interrupts
EI	Enable Interrupts
HALT	Halt
IM	Interrupt Mode
NOP	No Operation
RSMIX	Reset Mixed-Memory Mode Flag
SCF	Set Carry Flag
SLP	Sleep
STMIX	Set Mixed-Memory Mode Flag

Table 116. Program Control Instructions

Mnemonic	Instruction
CALL	Call Subroutine
CALL cc	Conditional Call Subroutine
DJNZ	Decrement and Jump if Nonzero
JP	Jump
JP cc	Conditional Jump
JR	Jump Relative
JR cc	Conditional Jump Relative
RET	Return
RET cc	Conditional Return
RETI	Return from Interrupt
RETN	Return from Nonmaskable interrupt
RST	Restart

Table 117. Rotate and Shift Instructions

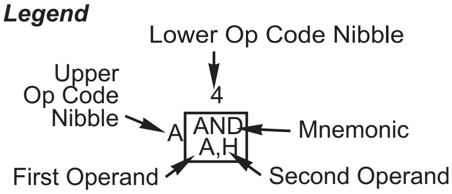
Mnemonic	Instruction
RL	Rotate Left
RLA	Rotate Left–Accumulator
RLC	Rotate Left Circular
RLCA	Rotate Left Circular–Accumulator
RLD	Rotate Left Decimal
RR	Rotate Right
RRA	Rotate Right–Accumulator
RRC	Rotate Right Circular
RRCA	Rotate Right Circular–Accumulator
RRD	Rotate Right Decimal
SLA	Shift Left Arithmetic
SRA	Shift Right Arithmetic
SRL	Shift Right Logical

Op-Code Map

Tables 118 through 124 indicate the hex values for each of the eZ80® instructions.

Table 118.Op Code Map—First Op Code

Legend



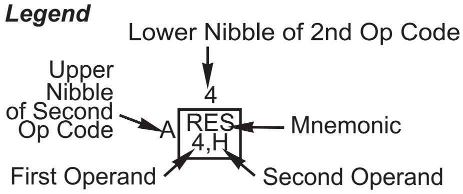
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LD BC, Mmn	LD (BC),A	INC BC	INC B	DEC B	LD B,n	RLCA	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	RRCA
1	DJNZ d	LD DE, Mmn	LD (DE),A	INC DE	INC D	DEC D	LD D,n	RLA	JR d	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	RRA
2	JR NZ,d	LD HL, Mmn	LD (Mmn), HL	INC HL	INC H	DEC H	LD H,n	DAA	JR Z,d	ADD HL,HL	LD HL, (Mmn)	DEC HL	INC L	DEC L	LD L,n	CPL
3	JR NC,d	LD SP, Mmn	LD (Mmn), A	INC SP	INC (HL)	DEC (HL)	LD (HL),n	SCF	JR CF,d	ADD HL,SP	LD A, (Mmn)	DEC SP	INC A	DEC A	LD A,n	CCF
4	.SIS suffix	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A	LD C,B	.LIS suffix	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
5	LD D,B	LD D,C	.SIL suffix	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A	LD E,B	LD E,C	LD E,D	.LIL suffix	LD E,H	LD E,L	LD E,(HL)	LD E,A
6	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
7	LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),A	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
8	ADD A,B	ADD A,C	ADD A,D	ADD A,E	ADD A,H	ADD A,L	ADD A,(HL)	ADD A,A	ADC A,B	ADC A,C	ADC A,D	ADC A,E	ADC A,H	ADC A,L	ADC A,(HL)	ADC A,A
9	SUB A,B	SUB A,C	SUB A,D	SUB A,E	SUB A,H	SUB A,L	SUB A,(HL)	SUB A,A	SBC A,B	SBC A,C	SBC A,D	SBC A,E	SBC A,H	SBC A,L	SBC A,(HL)	SBC A,A
A	AND A,B	AND A,C	AND A,D	AND A,E	AND A,H	AND A,L	AND A,(HL)	AND A,A	XOR A,B	XOR A,C	XOR A,D	XOR A,E	XOR A,H	XOR A,L	XOR A,(HL)	XOR A,A
B	OR A,B	OR A,C	OR A,D	OR A,E	OR A,H	OR A,L	OR A,(HL)	OR A,A	CP A,B	CP A,C	CP A,D	CP A,E	CP A,H	CP A,L	CP A,(HL)	CP A,A
C	RET NZ	POP BC	JP NZ, Mmn	JP Mmn	CALL NZ, Mmn	PUSH BC	ADD A,n	RST 00h	RET Z	RET	JP Z, Mmn	Table 119	CALL Z, Mmn	CALL Mmn	ADC A,n	RST 08h
D	RET NC	POP DE	JP NC, Mmn	OUT (n),A	CALL NC, Mmn	PUSH DE	SUB A,n	RST 10h	RET CF	EXX	JP CF, Mmn	IN A,(n)	CALL CF, Mmn	Table 120	SBC A,n	RST 18h
E	RET PO	POP HL	JP PO, Mmn	EX (SP),HL	CALL PO, Mmn	PUSH HL	AND A,n	RST 20h	RET PE	JP (HL)	JP PE, Mmn	EX DE,HL	CALL PE, Mmn	Table 121	XOR A,n	RST 28h
F	RET P	POP AF	JP P, Mmn	DI	CALL P, Mmn	PUSH AF	OR A,n	RST 30h	RET M	LD SP,HL	JP M, Mmn	EI	CALL M, Mmn	Table 122	CP A,n	RST 38h

Notes: n = 8-bit data; Mmn = 16- or 24-bit addr or data; d = 8-bit two's-complement displacement.

Table 119.Op Code Map—Second Op Code after 0CBh

Legend



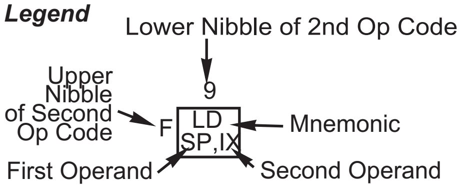
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	RLC B	RLC C	RLC D	RLC E	RLC H	RLC L	RLC (HL)	RLC A	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC (HL)	RRC A
1	RL B	RL C	RL D	RL E	RL H	RL L	RL (HL)	RL A	RR B	RR C	RR D	RR E	RR H	RR L	RR (HL)	RR A
2	SLA B	SLA C	SLA D	SLA E	SLA H	SLA L	SLA (HL)	SLA A	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA (HL)	SRA A
3									SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL (HL)	SRL A
4	BIT 0,B	BIT 0,C	BIT 0,D	BIT 0,E	BIT 0,H	BIT 0,L	BIT 0,(HL)	BIT 0,A	BIT 1,B	BIT 1,C	BIT 1,D	BIT 1,E	BIT 1,H	BIT 1,L	BIT 1,(HL)	BIT 1,A
5	BIT 2,B	BIT 2,C	BIT 2,D	BIT 2,E	BIT 2,H	BIT 2,L	BIT 2,(HL)	BIT 2,A	BIT 3,B	BIT 3,C	BIT 3,D	BIT 3,E	BIT 3,H	BIT 3,L	BIT 3,(HL)	BIT 3,A
6	BIT 4,B	BIT 4,C	BIT 4,D	BIT 4,E	BIT 4,H	BIT 4,L	BIT 4,(HL)	BIT 4,A	BIT 5,B	BIT 5,C	BIT 5,D	BIT 5,E	BIT 5,H	BIT 5,L	BIT 5,(HL)	BIT 5,A
7	BIT 6,B	BIT 6,C	BIT 6,D	BIT 6,E	BIT 6,H	BIT 6,L	BIT 6,(HL)	BIT 6,A	BIT 7,B	BIT 7,C	BIT 7,D	BIT 7,E	BIT 7,H	BIT 7,L	BIT 7,(HL)	BIT 7,A
8	RES 0,B	RES 0,C	RES 0,D	RES 0,E	RES 0,H	RES 0,L	RES 0,(HL)	RES 0,A	RES 1,B	RES 1,C	RES 1,D	RES 1,E	RES 1,H	RES 1,L	RES 1,(HL)	RES 1,A
9	RES 2,B	RES 2,C	RES 2,D	RES 2,E	RES 2,H	RES 2,L	RES 2,(HL)	RES 2,A	RES 3,B	RES 3,C	RES 3,D	RES 3,E	RES 3,H	RES 3,L	RES 3,(HL)	RES 3,A
A	RES 4,B	RES 4,C	RES 4,D	RES 4,E	RES 4,H	RES 4,L	RES 4,(HL)	RES 4,A	RES 5,B	RES 5,C	RES 5,D	RES 5,E	RES 5,H	RES 5,L	RES 5,(HL)	RES 5,A
B	RES 6,B	RES 6,C	RES 6,D	RES 6,E	RES 6,H	RES 6,L	RES 6,(HL)	RES 6,A	RES 7,B	RES 7,C	RES 7,D	RES 7,E	RES 7,H	RES 7,L	RES 7,(HL)	RES 7,A
C	SET 0,B	SET 0,C	SET 0,D	SET 0,E	SET 0,H	SET 0,L	SET 0,(HL)	SET 0,A	SET 1,B	SET 1,C	SET 1,D	SET 1,E	SET 1,H	SET 1,L	SET 1,(HL)	SET 1,A
D	SET 2,B	SET 2,C	SET 2,D	SET 2,E	SET 2,H	SET 2,L	SET 2,(HL)	SET 2,A	SET 3,B	SET 3,C	SET 3,D	SET 3,E	SET 3,H	SET 3,L	SET 3,(HL)	SET 3,A
E	SET 4,B	SET 4,C	SET 4,D	SET 4,E	SET 4,H	SET 4,L	SET 4,(HL)	SET 4,A	SET 5,B	SET 5,C	SET 5,D	SET 5,E	SET 5,H	SET 5,L	SET 5,(HL)	SET 5,A
F	SET 6,B	SET 6,C	SET 6,D	SET 6,E	SET 6,H	SET 6,L	SET 6,(HL)	SET 6,A	SET 7,B	SET 7,C	SET 7,D	SET 7,E	SET 7,H	SET 7,L	SET 7,(HL)	SET 7,A

Notes: n = 8-bit data; Mmn = 16- or 24-bit addr or data; d = 8-bit two's-complement displacement.

Table 120.Op Code Map—Second Op Code After 0DDh

Legend



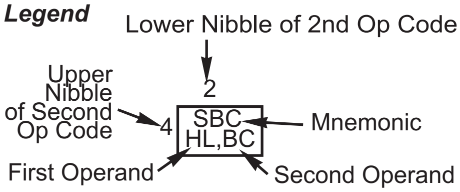
Lower Nibble (Hex) across the top; Upper Nibble (Hex) down the side.

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0								LD BC, (IX+d)		ADD IX,BC						LD (IX+d), BC
1								LD DE, (IX+d)		ADD IX,DE						LD (IX+d), DE
2		LD IX, Mmn	LD (Mmn), IX	INC IX	INC IXH	DEC IXH	LD IXH,n	LD HL, (IX+d)		ADD IX,IX	LD IX, (Mmn)	DEC IX	INC IXL	DEC IXL	LD IXL,n	LD (IX+d), HL
3		LD IY, (IX+d)			INC (IX+d)	DEC (IX+d)	LD (IX +d),n	LD IX, (IX+d)		ADD IX,SP					LD (IX+d), IY	LD (IX+d), IX
4					LD B,IXH	LD B,IXL	LD B, (IX+d)						LD C,IXH	LD C,IXL	LD C, (IX+d)	
5					LD D,IXH	LD D,IXL	LD D, (IX+d)						LD E,IXH	LD E,IXL	LD E, (IX+d)	
6	LD IXH,B	LD IXH,C	LD IXH,D	LD IXH,E	LD IXH,IXH	LD IXH,IXL	LD H, (IX+d)	LD IXH,A	LD IXL,B	LD IXL,C	LD IXL,D	LD IXL,E	LD IXL,IXH	LD IXL,IXL	LD L, (IX+d)	LD IXL,A
7	LD (IX+d), B	LD (IX+d), C	LD (IX+d), D	LD (IX+d), E	LD (IX+d),H	LD (IX+d), L		LD (IX+d), A					LD A,IXH	LD A,IXL	LD A, (IX+d)	
8					ADD A,IXH	ADD A,IXL	ADD A, (IX+d)						ADC A,IXH	ADC A,IXL	ADC A, (IX+d)	
9					SUB A,IXH	SUB A,IXL	SUB A, (IX+d)						SBC A,IXH	SBC A,IXL	SBC A, (IX+d)	
A					AND A,IXH	AND A,IXL	AND A, (IX+d)						XOR A,IXH	XOR A,IXL	XOR A, (IX+d)	
B					OR A,IXH	OR A,IXL	OR A, (IX+d)						CP A,IXH	CP A,IXL	CP A, (IX+d)	
C												Table 123				
D																
E		POP IX		EX (SP),IX		PUSH IX				JP (IX)						
F										LD SP,IX						

Notes: n = 8-bit data; Mmn = 16- or 24-bit addr or data; d = 8-bit two's-complement displacement.

Table 121.Op Code Map—Second Op Code After 0EDh

Legend



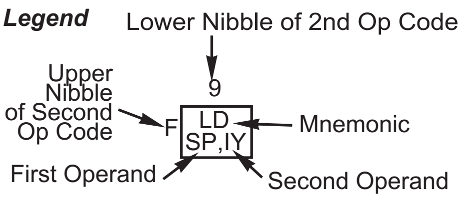
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	IN0 B,(n)	OUT0 (n),B	LEA BC, IX+d	LEA BC, IY+d	TST A,B			LD BC, (HL)	IN0 C,(n)	OUT0 (n),C			TST A,C			LD (HL), BC
1	IN0 D,(n)	OUT0 (n),D	LEA DE, IX+d	LEA DE, IY+d	TST A,D			LD DE, (HL)	IN0 E,(n)	OUT0 (n),E			TST A,E			LD(HL), DE
2	IN0 H,(n)	OUT0 (n),H	LEA HL ,IX+d	LEA HL ,IY+d	TST A,H			LD HL, (HL)	IN0 L,(n)	OUT0 (n),L			TST A,L			LD (HL), HL
3		LD IY, (HL)	LEA IX ,IX+d	LEA IY ,IY+d	TST A,(HL)			LD IX, (HL)	IN0 A,(n)	OUT0 (n),A			TST A,A		LD (HL),IY	LD (HL), IX
4	IN B,(BC)	OUT (BC),B	SBC HL,BC	LD (Mmn), BC	NEG	RETN	IM 0	LD I,A	IN C,(C)	OUT (C),C	ADC HL,BC	LD BC, (Mmn)	MLT BC	RETI		LD R,A
5	IN D,(BC)	OUT (BC),D	SBC HL,DE	LD (Mmn), DE	LEA IX, IY+d	LEA IY, IX+d	IM 1	LD A,I	IN E,(C)	OUT (C),E	ADC HL,DE	LD DE, (Mmn)	MLT DE		IM 2	LD A,R
6	IBN H,(C)	OUT (BC),H	SBC HL,HL	LD (Mmn), HL	TST A,n	PEA IX+d	PEA IY+d	RRD	IN L,(C)	OUT (C),L	ADC HL,HL	LD HL, (Mmn)	MLT HL	LD MB,A	LD A,MB	RLD
7			SBC HL,SP	LD (Mmn), SP	TSTIO n		SLP		IN A,(C)	OUT (C),A	ADC HL,SP	LD SP, (Mmn)	MLT SP	STMIX	RSMIX	
8			INIM	OTIM	INI2						INDM	OTDM	IND2			
9			INIMR	OTIMR	INI2R						INDMR	OTDMR	IND2R			
A	LDI	CPI	INI	OUTI	OUTI2				LDD	CPD	IND	OUTD	OUTD2			
B	LDIR	CPIR	INIR	OTIR	OTI2R				LDDR	CPDR	INDR	OTDR	OTD2R			
C			INIRX	OTIRX							INDRX	OTDRX				
D																
E																
F																

Notes: n = 8-bit data; Mmn = 16- or 24-bit addr or data; d = 8-bit two's-complement displacement.

Table 122.Op Code Map—Second Op Code After 0FDh

Legend

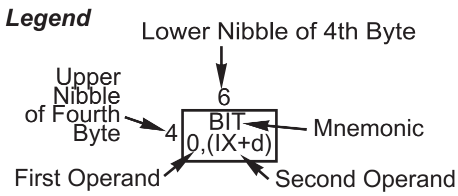


Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0								LD BC,(IY+d)		ADD IY,BC						LD (IY+d),BC
1								LD DE,(IY+d)		ADD IY,DE						LD (IY+d),DE
2		LD IY,Mmn	LD (Mmn),IY	INC IY	INC IYH	DEC IYH	LD IYH,n	LD HL,(IY+d)		ADD IY,IY	LD IY,(Mmn)	DEC IY	INC IYL	DEC IYL	LD IYL,n	LD (IY+d),HL
3		LD IX,(IY+d)			INC (IY+d)	DEC (IY+d)	LD (IY+d),n	LD IY,(IY+d)		ADD IY,SP					LD (IY+d),IX	LD (IY+d),IY
4					LD B,IYH	LD B,IYL	LD B,(IY+d)						LD C,IYH	LD C,IYL	LD C,(IY+d)	
5					LD D,IYH	LD D,IYL	LD D,(IY+d)						LD E,IYH	LD E,IYL	LD E,(IY+d)	
6	LD IYH,B	LD IYH,C	LD IYH,D	LD IYH,E	LD IYH,IYH	LD IYH,IYL	LD H,(IY+d)	LD IYH,A	LD IYL,B	LD IYL,C	LD IYL,D	LD IYL,E	LD IYL,IYH	LD IYL,IYL	LD L,(IY+d)	LD IYL,A
7	LD (IY+d),B	LD (IY+d),C	LD (IY+d),D	LD (IY+d),E	LD (IY+d),H	LD (IY+d),L		LD (IY+d),A					LD A,IYH	LD A,IYL	LD A,(IY+d)	
8					ADD A,IYH	ADD A,IYL	ADD A,(IY+d)						ADC A,IYH	ADC A,IYL	ADC A,(IY+d)	
9					SUB A,IYH	SUB A,IYL	SUB A,(IY+d)						SBC A,IYH	SBC A,IYL	SBC A,(IY+d)	
A					AND A,IYH	AND A,IYL	AND A,(IY+d)						XOR A,IYH	XOR A,IYL	XOR A,(IY+d)	
B					OR A,IYH	OR A,IYL	OR A,(IY+d)						CP A,IYH	CP A,IYL	CP A,(IY+d)	
C												Table 124				
D																
E		POP IY		EX (SP),IY		PUSH IY				JP (IY)						
F										LD SP,IY						

Notes: n = 8-bit data; Mmn = 16- or 24-bit addr or data; d = 8-bit two's-complement displacement.

Table 123.Op Code Map—Fourth Byte After 0DDh, 0CBh, and dd



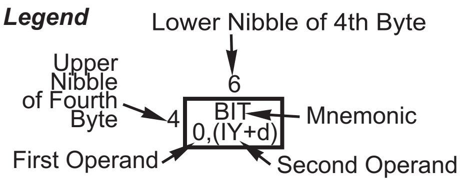
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0							RLC (IX+d)								RRC (IX+d)	
1							RL (IX+d)								RR (IX+d)	
2							SLA (IX+d)								SRA (IX+d)	
3															SRL (IX+d)	
4							BIT 0, (IX+d)								BIT 1, (IX+d)	
5							BIT 2, (IX+d)								BIT 3, (IX+d)	
6							BIT 4, (IX+d)								BIT 5, (IX+d)	
7							BIT 6, (IX+d)								BIT 7, (IX+d)	
8							RES 0, (IX+d)								RES 1, (IX+d)	
9							RES 2, (IX+d)								RES 3, (IX+d)	
A							RES 4, (IX+d)								RES 5, (IX+d)	
B							RES 6, (IX+d)								RES 7, (IX+d)	
C							SET 0, (IX+d)								SET 1, (IX+d)	
D							SET 2, (IX+d)								SET 3, (IX+d)	
E							SET 4, (IX+d)								SET 5, (IX+d)	
F							SET 6, (IX+d)								SET 7, (IX+d)	

Notes: d = 8-bit two's-complement displacement.

Table 124.Op Code Map—Fourth Byte After 0FDh, 0CBh, and dd*

Legend



Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0							RLC (IY+d)								RRC (IY+d)	
1							RL (IY+d)								RR (IY+d)	
2							SLA (IY+d)								SRA (IY+d)	
3															SRL (IY+d)	
4							BIT 0, (IY+d)								BIT 1, (IY+d)	
5							BIT 2, (IY+d)								BIT 3, (IY+d)	
6							BIT 4, (IY+d)								BIT 5, (IY+d)	
7							BIT 6, (IY+d)								BIT 7, (IY+d)	
8							RES 0, (IY+d)								RES 1, (IY+d)	
9							RES 2, (IY+d)								RES 3, (IY+d)	
A							RES 4, (IY+d)								RES 5, (IY+d)	
B							RES 6, (IY+d)								RES 7, (IY+d)	
C							SET 0, (IY+d)								SET 1, (IY+d)	
D							SET 2, (IY+d)								SET 3, (IY+d)	
E							SET 4, (IY+d)								SET 5, (IY+d)	
F							SET 6, (IY+d)								SET 7, (IY+d)	

Notes: d = 8-bit two's-complement displacement.

On-Chip Oscillators

The eZ80L92 features two on-chip oscillators for use with an external crystal. The primary oscillator generates the system clock for the internal CPU and the majority of the on-chip peripherals. Alternatively, the X_{IN} input pin can also accept a CMOS-level clock input signal. If an external clock generator is used, the X_{OUT} pin should be left unconnected. The secondary oscillator can drive a 32 KHz crystal to generate the time-base for the Real-Time Clock.

20 MHz Primary Crystal Oscillator Operation

Figure 44 illustrates a recommended configuration for connection with an external 20MHz, fundamental-mode, parallel-resonant crystal. Recommended crystal specifications are provided in Table 125. Resistor R_1 limits total power dissipation by the crystal. Printed circuit board layout should add no more than 4pF of stray capacitance to either the X_{IN} or X_{OUT} pins. If oscillation does not occur, reduce the values of capacitors C_1 and C_2 to decrease loading.

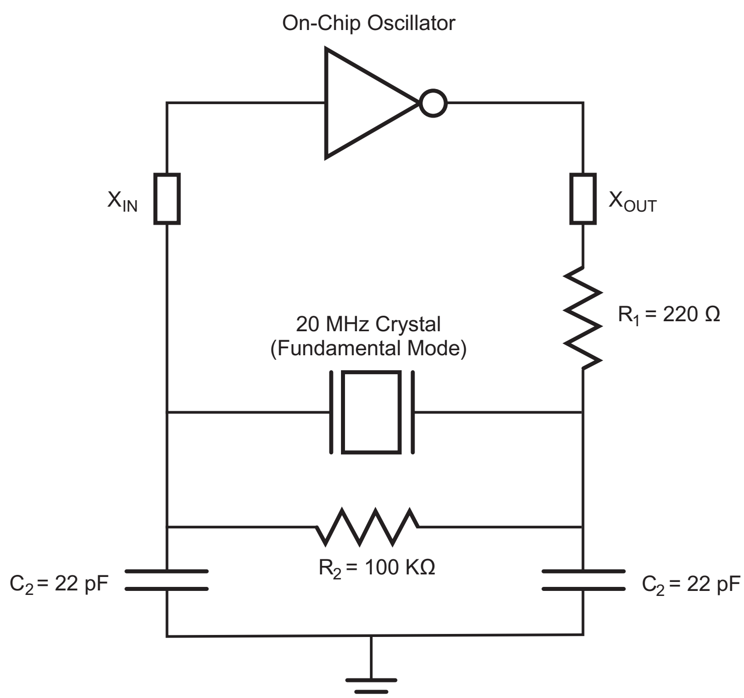


Figure 44.Recommended Crystal Oscillator Configuration (20MHz operation)

Table 125. Recommended Crystal Oscillator Specifications (20 MHz Operation)

Parameter	Value	Units	Comments
Frequency	20	MHz	
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	25	Ω	Maximum
Load Capacitance (C_L)	20	pF	Maximum
Shunt Capacitance (C_0)	7	pF	Maximum
Drive Level	1	mW	Maximum

50 MHz Primary Crystal Oscillator Operation

Figure 45 illustrates a recommended configuration for connection with an external 50MHz, third-overtone, parallel-resonant crystal. Recommended crystal specifications are provided in Table 126. Printed circuit board layout should add no more than 4 pF of stray capacitance to either the X_{IN} or X_{OUT} pins. If oscillation does not occur, the user should try removing C1 for testing and decreasing the value of C_2 by the estimated stray capacitance to decrease loading.

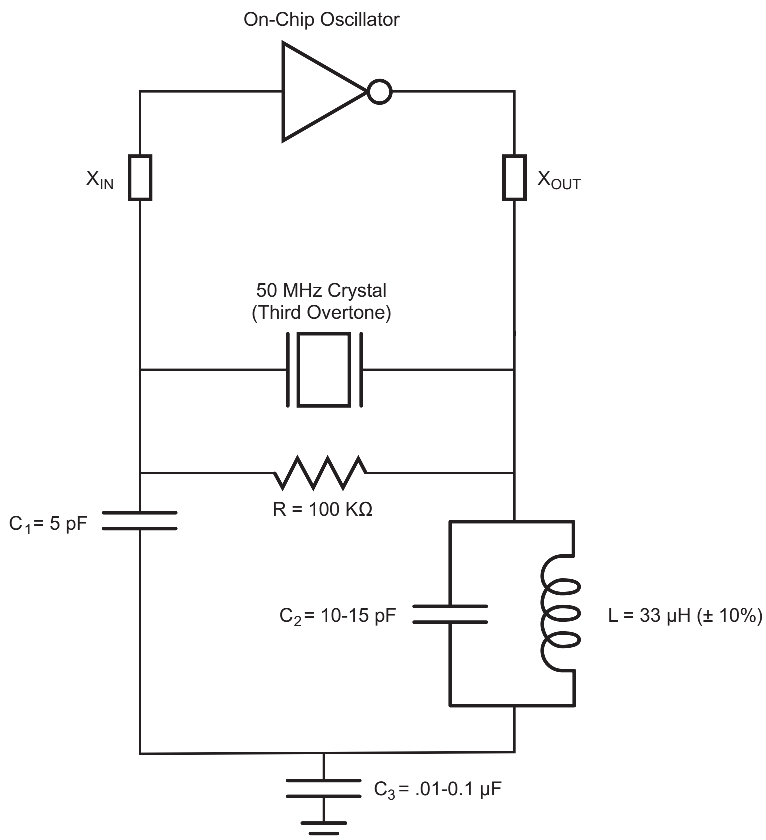


Figure 45.Recommended Crystal Oscillator Configuration (50MHz operation)

Table 126. Recommended Crystal Oscillator Specifications (50MHz Operation)

Parameter	Value	Units	Comments
Frequency	50	MHz	
Resonance	Parallel		
Mode	3rd Overtone		
Series Resistance (R_S)	65	Ω	Maximum
Load Capacitance (C_L)	20	pF	Maximum
Shunt Capacitance (C_0)	7	pF	Maximum
Drive Level	100	µW	Maximum

32 KHz Real-Time Clock Crystal Oscillator Operation

Figure 46 illustrates a recommended configuration for connecting the Real-Time Clock oscillator with an external 32 KHz, fundamental-mode, parallel-resonant crystal. The recommended crystal specifications are provided in Table 127. A printed circuit board layout should add no more than 4 pF of stray capacitance to either the RTC_X_{IN} or RTC_X_{OUT} pins. If oscillation does not occur, reduce the values of capacitors C_1 and C_2 to decrease loading.

An on-chip MOS resistor sets the crystal drive current limit. This configuration does not require an external bias resistor across the crystal. An on-chip MOS resistor provides the biasing.

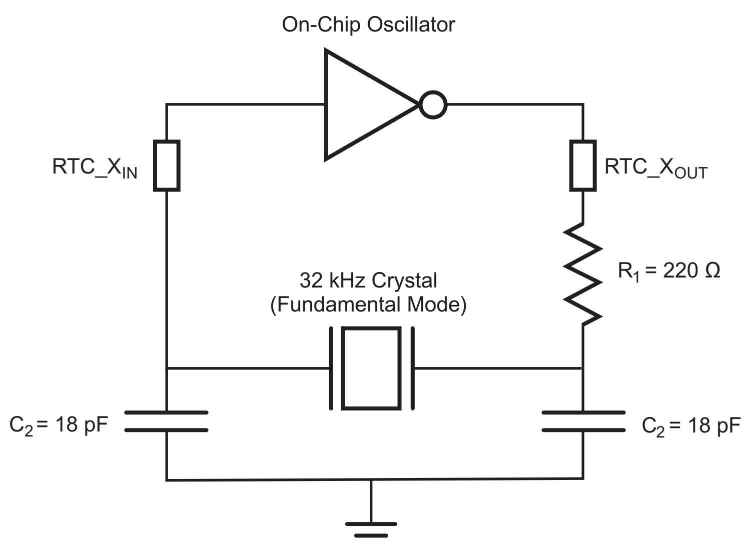


Figure 46.Recommended Crystal Oscillator Configuration (32KHz operation)

Table 127. Recommended Crystal Oscillator Specifications (32 KHz Operation)

Parameter	Value	Units	Comments
Frequency	32	KHz	32768 Hz
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	40	kΩ	Maximum

Table 127. Recommended Crystal Oscillator Specifications (32 KHz Operation) (Continued)

Parameter	Value	Units	Comments
Load Capacitance (C_L)	12.5	pF	Maximum
Shunt Capacitance (C_0)	3	pF	Maximum
Drive Level	1	µW	Maximum

Electrical Characteristics

Absolute Maximum Ratings

Stresses greater than those listed in Table 128 may cause permanent damage to the device. These ratings are stress ratings only. Operation of the device at any condition outside those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. For improved reliability, unused inputs should be tied to one of the supply voltages (V_{DD} or V_{SS}).

Table 128. Absolute Maximum Ratings

Parameter	Min	Max	Units	Notes
Ambient temperature under bias (ºC)	–40	+105	C	1
Storage temperature (ºC)	–65	+150	C	
Voltage on any pin with respect to V_{SS}	–0.3	+6.0	V	2
Voltage on V_{DD} pin with respect to V_{SS}	–0.3	+6.0	V	
Total power dissipation		520	mW	
Maximum current out of V_{SS}		145	mA	
Maximum current into V_{DD}		145	mA	
Maximum current on input and/or inactive output pin	–15	+15	µA	
Maximum output current from active output pin	–8	+8	mA	

Notes:
1. Operating temperature is specified in DC Characteristics.
2. This voltage applies to all pins except where noted otherwise.

DC Characteristics

Table 129 lists the DC characteristics of the eZ80L92. Figures 47 and 48 plot supply current values against CPU frequency and wait states.

Note: All data is preliminary and subject to change following completion of production characterization.

Table 129. DC Characteristics

Symbol	Parameter	T_A = 0ºC to 70ºC Min	T_A = 0ºC to 70ºC Max	T_A = –40ºC to 105ºC Min	T_A = –40ºC to 105ºC Max	Units	Conditions
V_{DD}	Supply Voltage	3.0	3.6	3.0	3.6	V	
V_{IL}	Low Level Input Voltage	–0.3	0.8	–0.3	0.8	V	
V_{IH}	High Level Input Voltage	$0.7 \times V_{DD}$	5.5	$0.7 \times V_{DD}$	5.5	V	
V_{OL}	Low Level Output Voltage		0.4		0.4	V	V_{DD} = 3.0V; I_{OL} = 1mA
V_{OH}	High Level Output Voltage	2.4		2.4		V	V_{DD} = 3.0V; I_{OH} = –1mA
I_{IL}	Input Leakage Current	–10	+10	–10	+10	µA	V_{DD} = 3.6V; V_{IN} = V_{DD} or V_{SS}[1]
I_{TL}	Tristate Leakage Current	–10	+10	–10	+10	µA	V_{DD} = 3.6V
I_{DD}	Power Dissipation (normal operation)		100		100	mA	F = 20MHz
			145		145	mA	F = 50MHz
	Power Dissipation (HALT mode)		10		10	mA	F = 20MHz
			20		20	mA	F = 50MHz
	Power Dissipation (SLEEP mode)		10		25	µA	Internal clocks stopped
RTC_V_{DD}	RTC Supply Voltage	3.0	3.6	3.0	3.6	V	
I_{RTC}	RTC Supply Current	2.5 typical	10	2.5 typical	10	µA	Supply current into RTC_V_{DD}

Note: [1]This condition excludes all pins with on-chip pull-ups when driven Low.

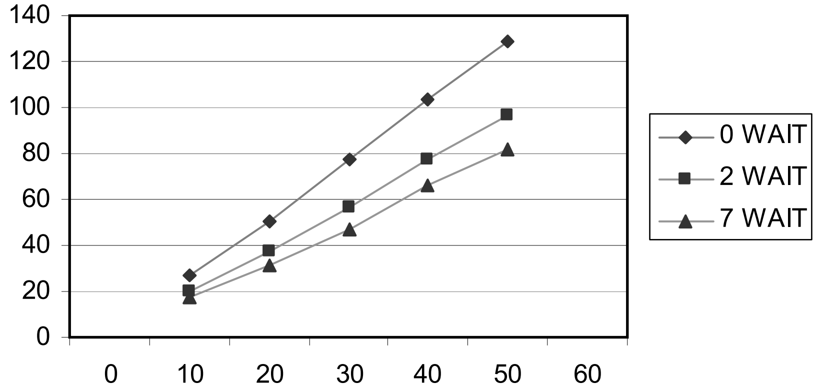


Figure 47. I_{CC} **vs. Frequency (Typical @ 3.3V, 25ºC)**

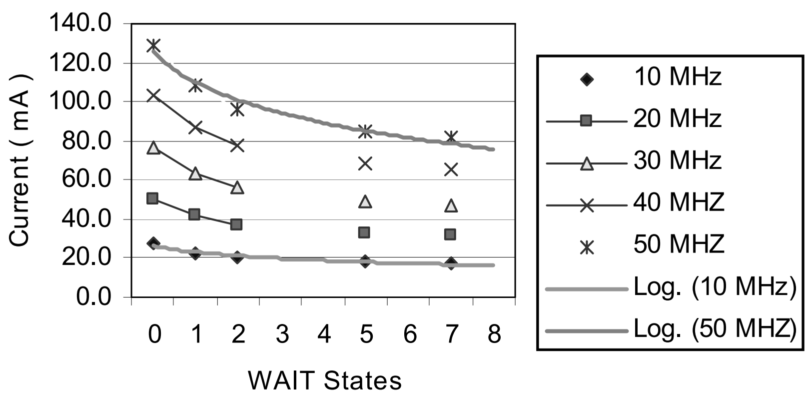


Figure 48. I_{CC} **vs. WAIT (Typical @ 3.3V, 25ºC)**

AC Characteristics

The section provides information on the AC characteristics and timing of the eZ80L92. All AC timing information assumes a standard load of 50 pF on all outputs. See Table 130.

Note: All data is preliminary and subject to change following completion of production characterization.

Table 130. AC Characteristics

		T_A = 0ºC to 70ºC		T_A = –40ºC to 105ºC			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Conditions**
T_{XIN}	System Clock Cycle Time	20		20		ns	V_{DD} = 3.0 – 3.6V
T_{XINH}	System Clock High Time		10		10	ns	V_{DD} = 3.0 – 3.6V; T_{CLK} = 50ns
T_{XINL}	System Clock Low Time		10		10	ns	V_{DD} = 3.0 – 3.6V; T_{CLK} = 50ns
T_{XINR}	System Clock Rise Time		3		3	ns	V_{DD} = 3.0 – 3.6V; T_{CLK} = 50ns
T_{XINF}	System Clock Fall Time		3		3	ns	V_{DD} = 3.0 – 3.6V; T_{CLK} = 50ns

External Memory Read Timing

Figure 49 and Table 131 diagram the timing for external READs.

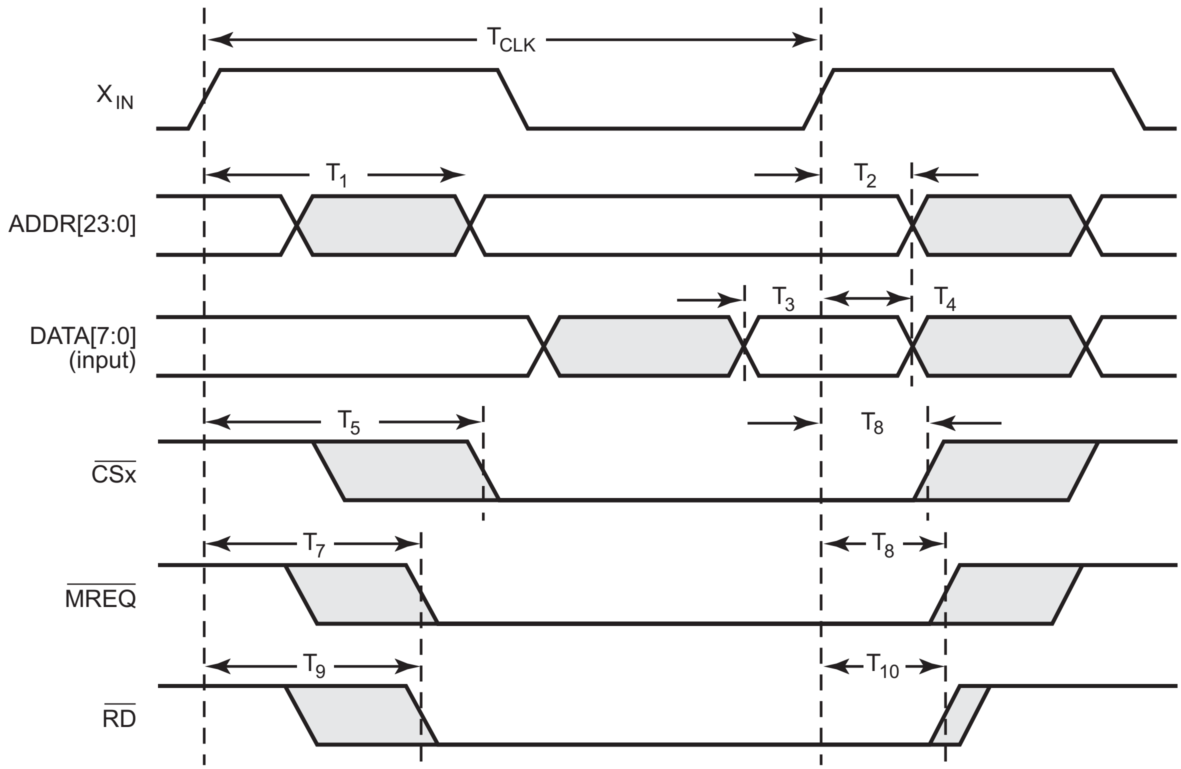


Figure 49. External Memory Read Timing

Table 131. 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
T_8	Clock Rise to $\overline{\text{MREQ}}$ Deassertion Delay	2.6	6.9	2.6	6.9

Table 131. External Read Timing (Continued)

Parameter	Description	20 MHz (ns) Min.	20 MHz (ns) Max.	50 MHz (ns) Min.	50 MHz (ns) Max.
T_9	Clock Rise to $\overline{RD}$ Assertion Delay	3.0	9.8	3.0	9.8
T_{10}	Clock Rise to $\overline{RD}$ Deassertion Delay	2.6	7.1	2.6	7.1

External Memory Write Timing

Figure 50 and Table 132 diagram the timing for external writes.

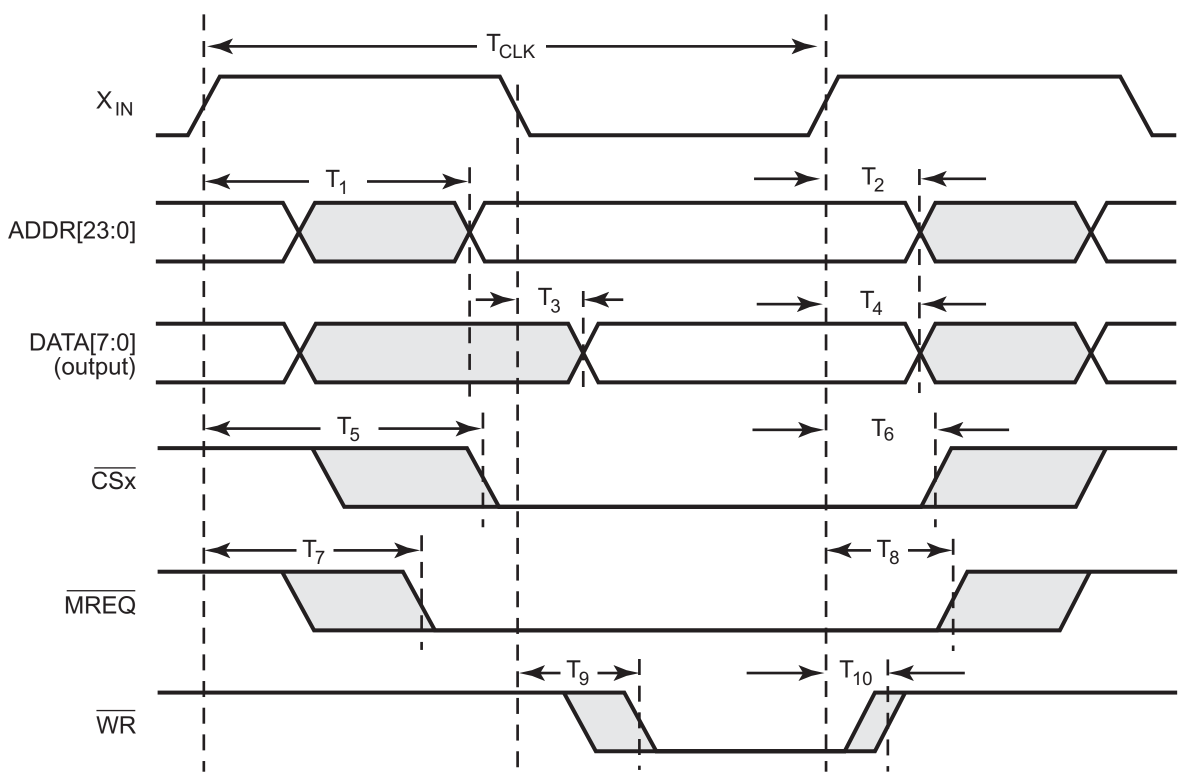


Figure 50. External Memory Write Timing

Table 132. External Write Timing

Parameter	Description	20 MHz (ns)		50 MHz (ns)	
		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, deassertion of $\overline{\text{WR}}$ always occurs before any change to ADDR, DATA, $\overline{\text{CSx}}$, or MREQ. In certain applications, the deassertion of WR can be concurrent with ADDR, DATA, CSx, or MREQ when buffering is used off-chip.

External I/O Read Timing

Figure 51 and Table 133 diagram the timing for external I/O Reads.

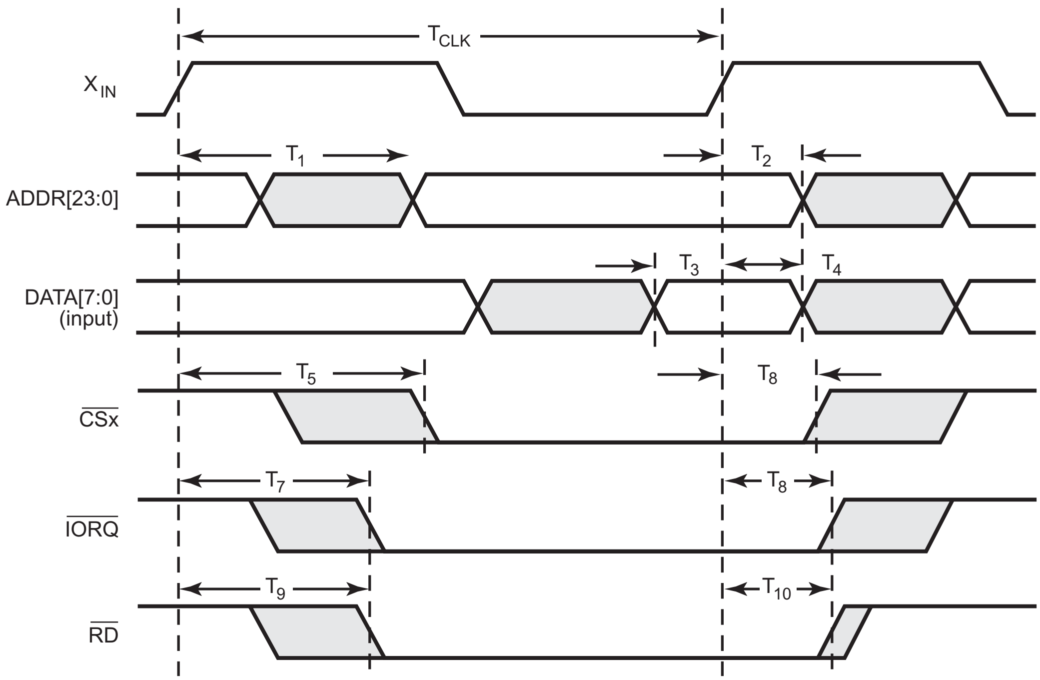


Figure 51.External I/O Read Timing

Table 133. External I/O Read Timing

		20 MHz (ns)		50 MHz (ns)	
Parameter	**Abbreviation**	**Min**	**Max**	**Min.**	**Max.**
T_1	Clock Rise to ADDR Valid Delay	—	6.9	—	6.9
T_2	Clock Rise to ADDR Hold Time	2.2	—	2.2	—
T_3	Input DATA Valid to Clock Rise Setup Time	0.4	—	0.4	—
T_4	Clock Rise to DATA Hold Time	1.3	—	1.3	—
T_5	Clock Rise to $\overline{\text{CSx}}$ Assertion Delay	2.6	10.8	2.6	10.8
T_6	Clock Rise to $\overline{\text{CSx}}$ Deassertion Delay	2.4	8.8	2.4	8.8
T_7	Clock Rise to $\overline{\text{IORQ}}$ Assertion Delay	2.6	7.0	2.6	7.0
T_8	Clock Rise to $\overline{\text{IORQ}}$ Deassertion Delay	2.3	6.3	2.3	6.3

Table 133. External I/O Read Timing (Continued)

		20 MHz (ns)		50 MHz (ns)	
Parameter	**Abbreviation**	**Min**	**Max**	**Min.**	**Max.**
T_9	Clock Rise to $\overline{RD}$ Assertion Delay	2.7	7.0	2.7	7.0
T_{10}	Clock Rise to $\overline{RD}$ Deassertion Delay	2.4	6.3	2.4	6.3

External I/O Write Timing

Figure 52 and Table 134 diagram the timing for external I/O writes.

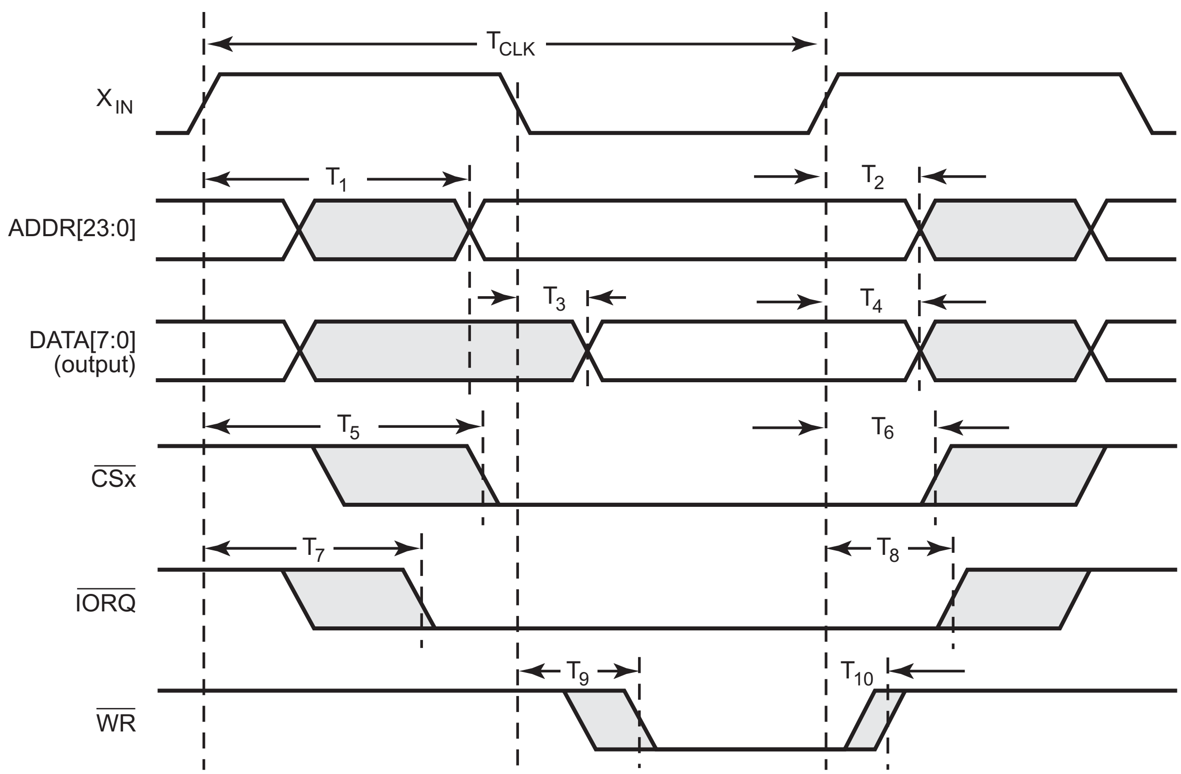


Figure 52. External I/O Write Timing

Table 134. External I/O Write Timing

Parameter	Abbreviation	20 MHz (ns)		50 MHz (ns)	
		Min	Max	Min.	Max.
T_1	Clock Rise to ADDR Valid Delay	—	7.7	—	7.7
T_2	Clock Rise to ADDR Hold Time	2.2	—	2.2	—
T_3	Clock Fall to Output DATA Valid Delay	—	6	—	6
T_4	Clock Rise to DATA Hold Time	2.3	—	2.3	—
T_5	Clock Rise to $\overline{\text{CSx}}$ Assertion Delay	2.6	10.8	2.6	10.8
T_6	Clock Rise to $\overline{\text{CSx}}$ Deassertion Delay	2.4	8.8	2.4	8.8
T_7	Clock Rise to $\overline{\text{IORQ}}$ Assertion Delay	2.6	7.0	2.6	7.0
T_8	Clock Rise to $\overline{\text{IORQ}}$ Deassertion Delay	2.3	6.3	2.3	6.3
T_9	Clock Fall to $\overline{\text{WR}}$ Assertion Delay	1.8	4.5	1.8	4.5
T_{10}	Clock Rise to $\overline{\text{WR}}$ Deassertion Delay*	1.6	4.4	1.6	4.4
	$\overline{\text{WR}}$ Deassertion to ADDR Hold Time	0.4	—	0.4	—
	$\overline{\text{WR}}$ Deassertion to DATA Hold Time	0.5	—	0.5	—
	$\overline{\text{WR}}$ Deassertion to $\overline{\text{CSx}}$ Hold Time	1.2	—	1.2	—
	$\overline{\text{WR}}$ Deassertion to $\overline{\text{IORQ}}$ Hold Time	0.5	—	0.5	—

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

Wait State Timing for Read Operations

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

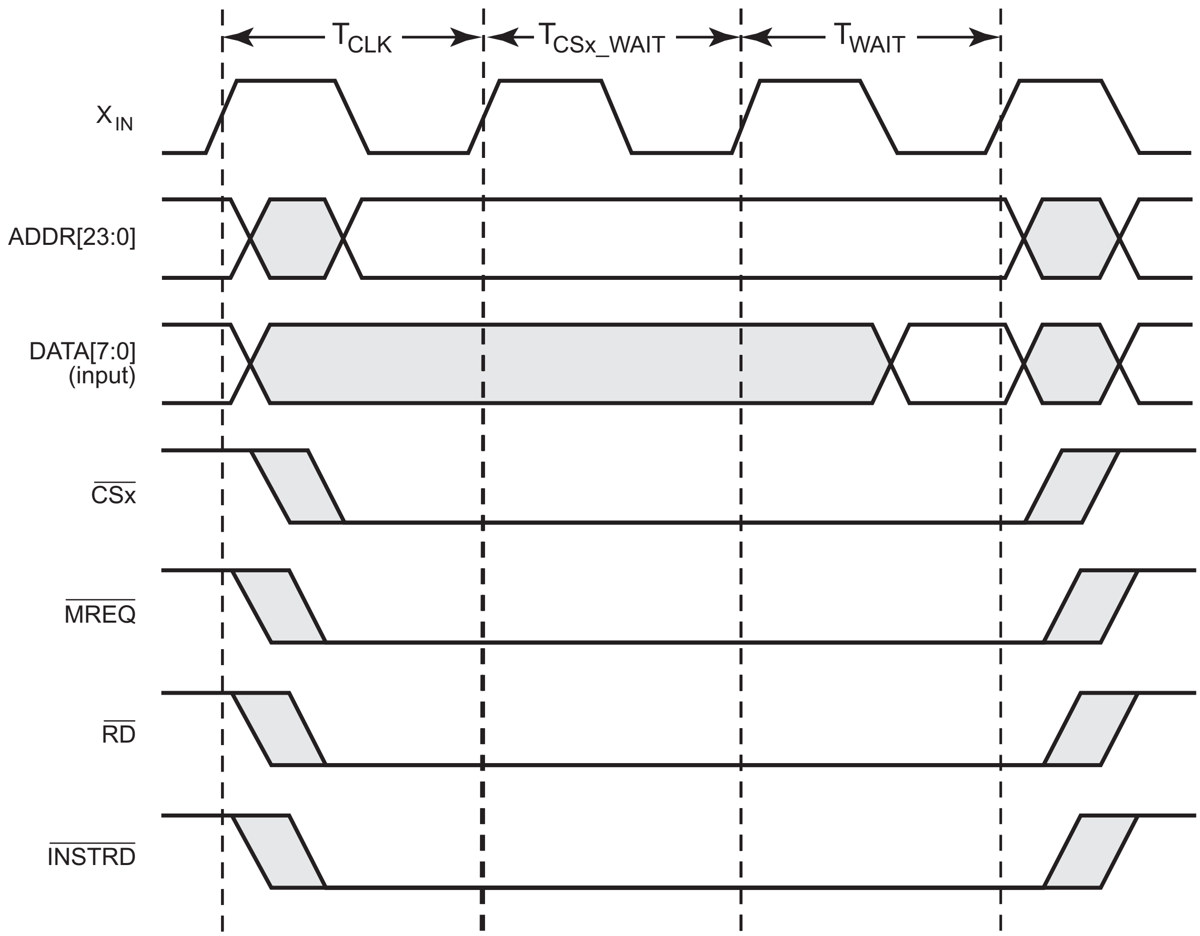


Figure 53. Wait State Timing for Read Operations

Wait State Timing for Write Operations

Figure 54 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.

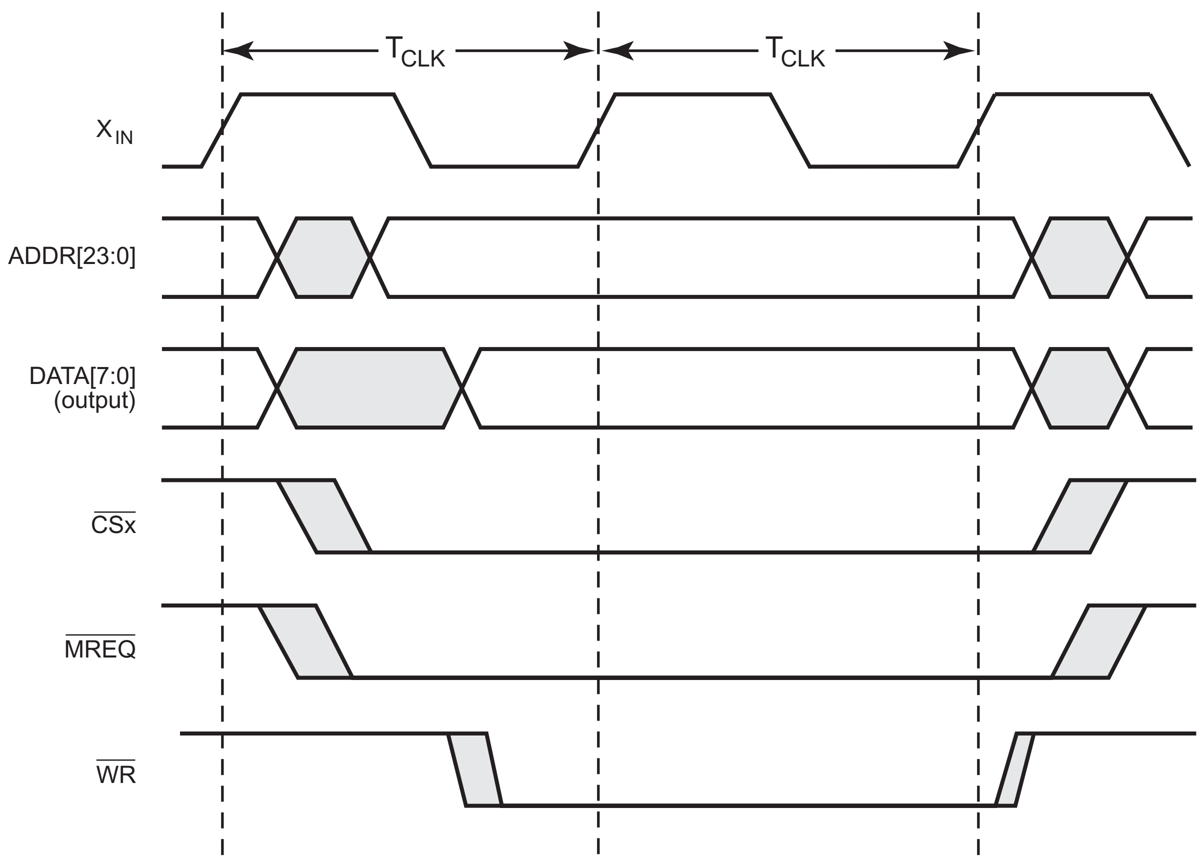


Figure 54. Wait State Timing for Write Operations

General Purpose I/O Port Input Sample Timing

Figure 55 illustrates timing of the GPIO input sampling. The input value on a GPIO port pin is sampled on the rising edge of the system clock. The port value is then available to the CPU on the second rising clock edge following the change of the port value.

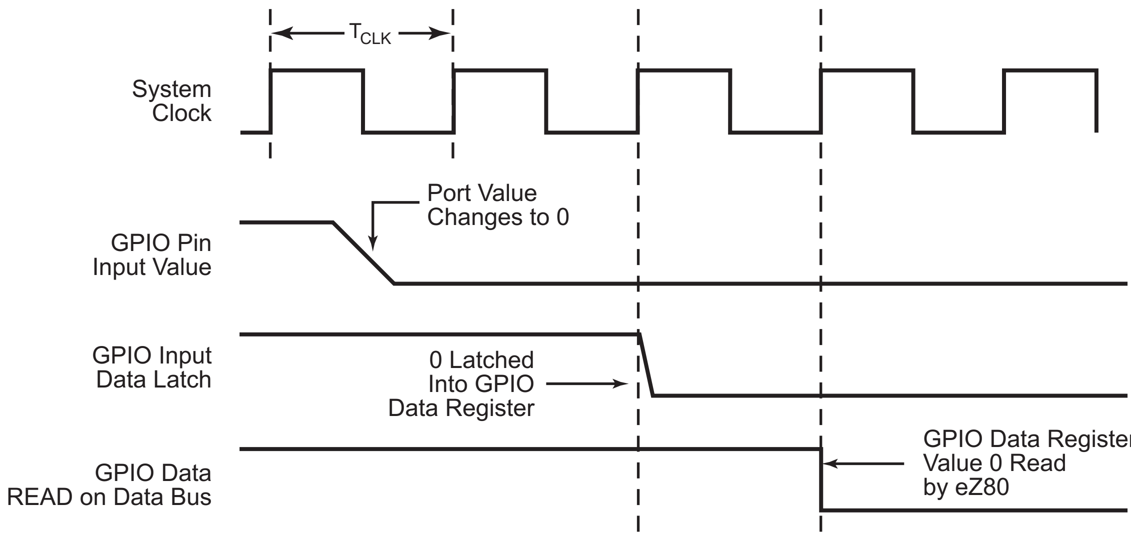


Figure 55. Port Input Sample Timing

General Purpose I/O Port Output Timing

Figure 56 and Table 135 provide timing information for GPIO port pins.

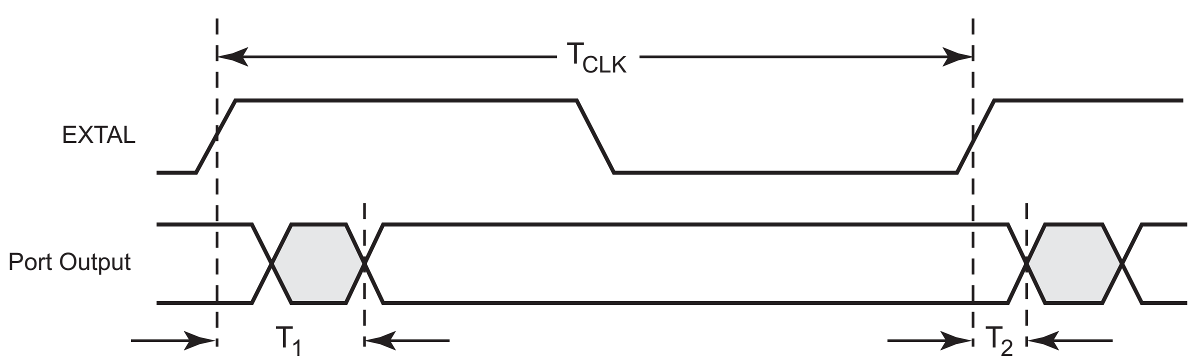


Figure 56. GPIO Port Output Timing

Table 135. GPIO Port Output Timing

		20 MHz (ns)		50 MHz (ns)	
Parameter	**Abbreviation**	**Min**	**Max**	**Min**	**Max**
T_1	Clock Rise to Port Output Valid Delay	—	9.3	—	9.3
T_2	Clock Rise to Port Output Hold Time	2.0	—	2.0	—

External Bus Acknowledge Timing

Table 136 provides information on the bus acknowledge timing.

Table 136. Bus Acknowledge Timing

		20 MHz (ns)		50 MHz (ns)	
Parameter	**Abbreviation**	**Min**	**Max**	**Min**	**Max**
T_1	Clock Rise to $\overline{\text{BUSACK}}$ Assertion Delay	2.8	9.3	2.8	9.3
T_2	Clock Rise to $\overline{\text{BUSACK}}$ Deassertion Delay	2.5	6.5	2.5	6.5

External System Clock Driver (PHI) Timing

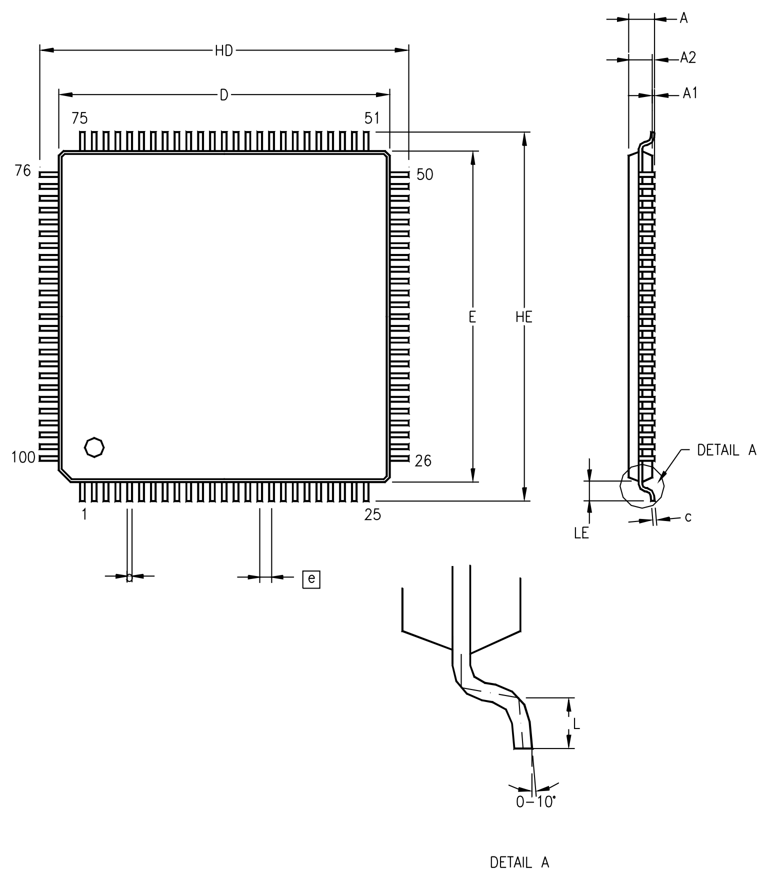
Table 137 provides timing information for the PHI pin. The PHI pin allows external peripherals to synchronize with the internal system clock driver on the eZ80L92.

Table 137. PHI System Clock Timing

		20 MHz (ns)		50 MHz (ns)	
Parameter	**Abbreviation**	**Min**	**Max**	**Min**	**Max**
T_1	Clock Rise to PHI Rise	1.6	4.6	1.6	4.6
T_2	Clock Fall to PHI Fall	1.8	4.3	1.8	4.3

Packaging

Figure 57 illustrates the 100-pin LQFP (low-profile quad flat pack) package for the eZ80L92 devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	1.35	1.60	.053	.063
A1	0.05	0.20	.002	.008
A2	1.30	1.50	.051	.059
b	0.15	0.26	.006	.010
c	0.10	0.20	.004	.008
HD	15.85	16.15	.624	.636
D	13.90	14.10	.547	.555
HE	15.85	16.15	.624	.636
E	13.90	14.10	.547	.555
e	0.50 BSC		.0197 BSC	
L	0.35	0.65	.014	.026
LE	0.90	1.10	.035	.043

1. CONTROLLING DIMENSIONS : MM
2. MAX COPLANARITY : .10mm / .004"

Figure 57. 100-Lead Plastic Low-Profile Quad Flat Package (LQFP)

Ordering Information

Table 138 provides a part number, a product specification index code, and a brief description of each eZ80L92 part.

Table 138. Ordering Information

Part	PSI	Description
eZ80L92	eZ80L92AZ020SC	100-pin LQFP, 20 MHz, Standard Temperature
eZ80L92	eZ80L92AZ020EC	100-pin LQFP, 20 MHz, Extended Temperature
eZ80L92	eZ80L92AZ050SC	100-pin LQFP, 50 MHz, Standard Temperature
eZ80L92	eZ80L92AZ050EC	100-pin LQFP, 50 MHz, Extended Temperature

Navigate your browser to ZiLOG's website to order the eZ80L92 MPU. Or, contact your local ZiLOG Sales Office to order these devices. ZiLOG provides additional assistance on its Customer Service page, and is also here to help with technical support issues.

For ZILOG's valuable software development tools and downloadable software, visit the ZiLOG website.

Part Number Description

ZiLOG part numbers consist of a number of components, as indicated in the following examples:

ZiLOG Base Products	
eZ80	ZiLOG eZ80® CPU
L92	Product Number
AZ	Package
050	Speed
S or E	Temperature
C	Environmental Flow

Package	AZ = LQFP (also called the VQFP)
Speed	050 = 50 MHz
Standard Temperature	S = 0ºC to +70ºC
Extended Temperature	E = –40ºC to +105ºC
Environmental Flow	C = Plastic Standard

Example: Part number eZ80L92AZ020SC is an eZ80® CPU product in a LQFP package, operating with a 20-MHz external clock frequency over a 0ºC to +70ºC temperature range and built using the Plastic Standard environmental flow.

Precharacterization Product

The product represented by this document is newly introduced and ZiLOG has not completed the full characterization of the product. The document states what ZiLOG knows about this product at this time, but additional features or nonconformance with some aspects of the document might be found, either by ZiLOG or its customers in the course of further application and characterization work. In addition, ZiLOG cautions that delivery might be uncertain at times, due to start-up yield issues.

ZiLOG, Inc.
532 Race Street
San Jose, CA 95126
Telephone (408) 558-8500
FAX 408 558-8300
Internet: www.zilog.com

Document Information

Document Number Description

The Document Control Number that appears in the footer on each page of this document contains unique identifying attributes, as indicated in the following table:

PS	Product Specification
0130	Unique Document Number
10	Revision Number
0903	Month and Year Published

Change Log

Rev	Date	Purpose	By
01	01/02	Original issue	D. Wilson, R. Beebe
02	06/02	Memory timing revisions	J. Eversmann, R. Beebe
03	06/02	Memory timing revisions	J. Eversmann, R. Beebe
04	10/02	Modifications to characteristics data	J. Eversmann, R. Beebe
05	11/02	Modifications to characteristics data	J. Eversmann, R. Beebe
06	01/03	Minor modifications to content	R. Beebe
07	02/03	Minor modifications to content	R. Johnson, R. Beebe
08	06/03	Modifications to characteristics data	C. Bender
09	08/03	Clarification to Tables 24 and 25	E. Aquino, R. Beebe
10	09/03	Minor modifications	R. Beebe

Index

D

E

R

S

T

U

V

W

Z

Customer Feedback Form

The eZ80L92 Product Specification

If you experience any problems while operating this product, or if you note any inaccuracies while reading this Product Specification, please copy and complete this form, then mail or fax it to ZiLOG (see *Return Information*, below). We also welcome your suggestions!

Customer Information

Name	Country
Company	Phone
Address	Fax
City/State/Zip	Email

Product Information

Serial # or Board Fab #/Rev. #
Software Version
Document Number
Host Computer Description/Type

Return Information

ZiLOG
System Test/Customer Support
532 Race Street
San Jose, CA 95126
Phone: (408) 558-8500
Fax: (408) 558-8536
ZiLOG Customer Support

Problem Description or Suggestion

Provide a complete description of the problem or your suggestion. If you are reporting a specific problem, include all steps leading up to the occurrence of the problem. Attach additional pages as necessary.