zilog

An IXYS Company

eZ80Acclaim!® Flash Microcontrollers

eZ80F91 MCU

Product Specification

PS019215-0910

www.zilog.com

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Revision History

Each instance in the Revision History reflects a change to this document from its previous revision. For more details, refer to the corresponding pages or appropriate links given in the table below.

Date	Revision Level	Section	Description	Page Number
September 2010	15	All	Updated logos and copyright date.	All
August 2008	14	Ordering Information	Updated Part Number Description section.	360
May 2008	13	Introduction, Figure 48, ZDI-Supported Protocol, and Figure 49	Replaced ZPAK II with USB Smart Cable	231, 232, and 233
September 2007	12	General-Purpose Input/ Output, Flash Memory, Universal Asynchronous Receiver/Transmitter, Serial Peripheral Interface, Real-Time Clock Control Register, I^2C Serial I/O Interface, Pin Description, and Ordering Information.	Updated Table 1, Figure 6, Flash Program Control Register, UART Transmitter, Figure 40, Table 93, I2C Registers and Ordering Information.	4, 53, 112,174, 176, 201, 223, and 359
February 2007	11	Register Map, GPIO Mode 7—Alternate Functions, Register Map - Table 3. Low-Power Modes, Electrical Characteristics chapters. Updated Table 93.		27, 45, 54, 339

Date	Revision Level	Section	Description	Page Number
June 2006	10	Global modifications	Updated for new release.	All
		Pin Identification on the eZ80F91 Device	Table 3: The description of the following pins modified: pins 55, 61, 63 and 69	6
		General-Purpose Input/ Output	GPIO chapter totally rewritten	49
		Chip Selects and Wait States	Input/Output chip select operation modified	65
		Flash Memory	The following sections are modified in Flash memory chapter: Erasing Flash memory, Information page characteristics, Flash Write/Erase protection register, Flash program control registers, and Table 43.	97
		Real-Time Clock Overview	Added a note in real time clock overview section	159
		Universal Asynchronous Receiver/Transmitter	Table 102 and 109 modified	175
		Infrared Encoder/Decoder Control Registers	The field [7:4] modified in Table 111	199
		Zilog Debug Interface	Updated the Introduction section, Added two paragraphs to ZDI Read Memory Registers	231
		On-Chip Oscillators	On page 349, Figure 63: Recommended Crystal Oscillator Configuration, the value of inductance L is changed to 3.3 µH. On page 351, Table 232, changed serial resistance value from 40 kΩ to 50 kΩ	336
		POR and VBO Electrical Characteristics	In Table 235: Min, Typ, and Max values of VBO voltage threshold modified and added IS_{por_vbo} parameter	341
		Ordering Information	Ordering information modified	359

Table of Contents

Architectural Overview

Zilog's eZ80F91 device is a member of Zilog's family of eZ80Acclaim!® Flash microcontrollers. The eZ80F91 is a high-speed microcontroller with a maximum clock speed of 50 MHz and single-cycle instruction fetch. It operates in Z80®-compatible addressing mode (64 KB) or full 24-bit addressing mode (16 MB). The rich peripheral set of the eZ80F91 makes it suitable for a variety of applications, including industrial control, embedded communication, and point-of-sale terminals.

Features

Key features of eZ80F91 device include:

- Single-cycle instruction fetch, high-performance, pipelined eZ80® CPU core (referred as *The CPU* in this document)
- 10/100 BaseT ethernet media access controller with Media-Independent Interface (MII)
- 256 KB Flash memory
- 16 KB SRAM (8 KB user and 8 KB Ethernet)
- Low-power features including SLEEP mode, HALT mode, and selective peripheral power-down control
- Two Universal Asynchronous Receiver/Transmitter (UART) with independent Baud Rate Generators (BRG)
- Serial Peripheral Interface (SPI) with independent clock rate generator
- I^2C with independent clock rate generator
- IrDA-compliant infrared encoder/decoder
- Glueless external peripheral interface with 4 Chip Selects, individual Wait State generators, an external $\overline{WAIT}$ input pin—supports Z80-, Intel-, and Motorola-style buses
- Fixed-priority vectored interrupts (both internal and external) and interrupt controller
- Real-time clock with separate V_{DD} pin for battery backup and selectable on-chip 32 kHz oscillator or external 50/60 Hz input
- Four 16-bit Counter/Timers with prescalers and direct input/output drive
- Watchdog Timer with internal oscillator clocking option
- 32 bits of General-Purpose Input/Output (GPIO)
- On-Chip Instrumentation (OCI™) and Zilog Debug Interfaces (ZDI)

- IEEE 1149.1-compatible JTAG
- 144-pin LQFP and BGA packages
- 3.0 V to 3.6 V supply voltage with 5 V tolerant inputs
- Operating Temperature Range:
  - Standard: 0 ºC to +70 ºC
  - Extended: –40 ºC to +105 ºC

Note: *All signals with an overline are active Low. For example, the signal* $\overline{\text{DCD1}}$ *is active when it is a logical 0 (Low) state.*

The power connections conventions are provided in the table below.

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

Block Diagram

Figure 1 on page 3 displays a block diagram of the eZ80F91 microcontroller.

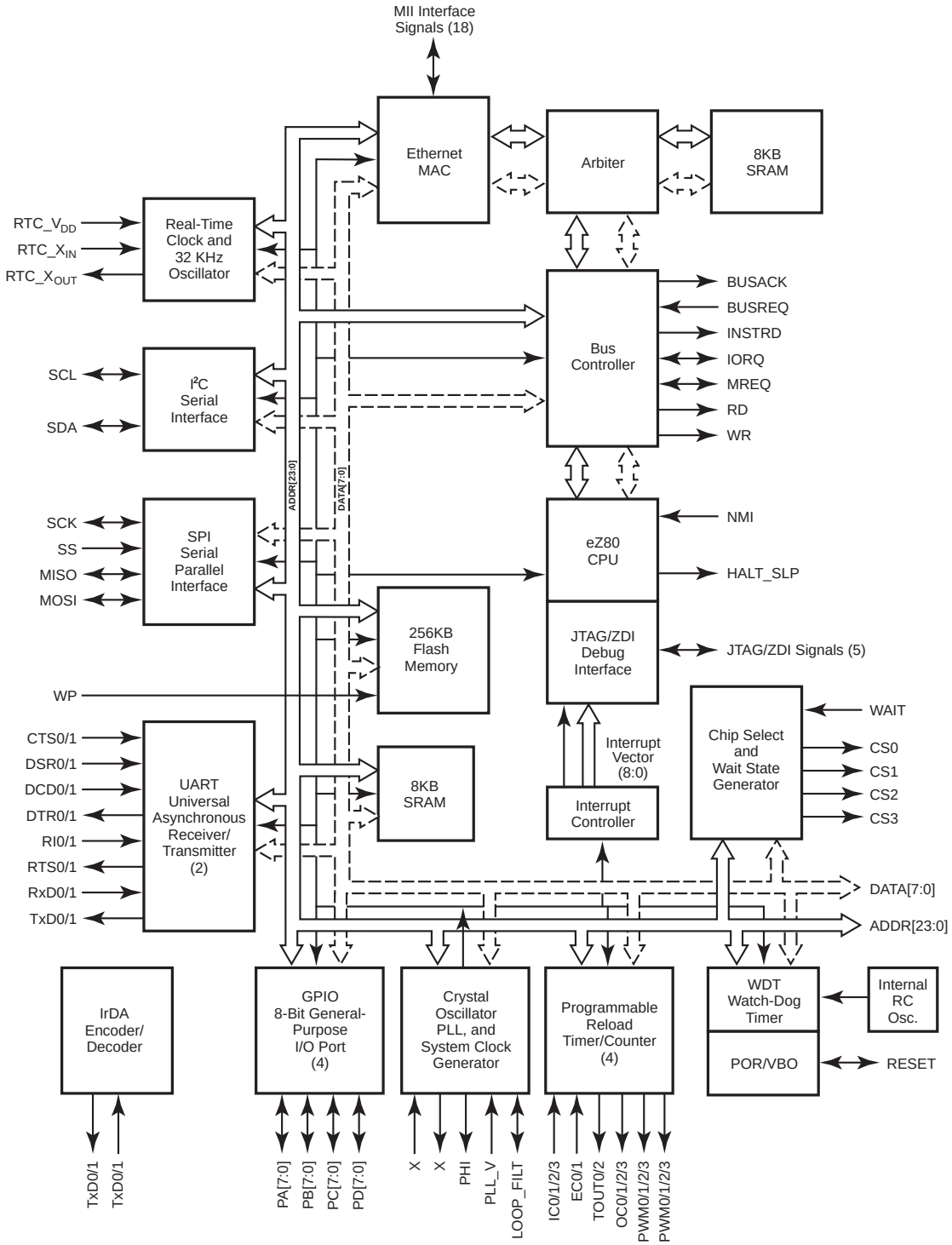


Figure 1. eZ80F91 Block Diagram

Pin Description

Table 1 lists the pin configuration of the eZ80F91 device in the 144-BGA package.

Table 1. eZ80F91 144-BGA Pin Configuration

	12	11	10	9	8	7	6	5	4	3	2	1
A	SDA	SCL	PA0	PA4	PA7	COL	TxD0	V_{DD}	Rx_DV	MDC	WPn	A0
B	V_{SS}	PHI	PA1	PA3	V_{DD}	TxD3	Tx_EN	V_{SS}	RxD1	MDIO	A2	A1
C	PB6	PB7	V_{DD}	PA5	V_{SS}	TxD2	Tx_CLK	Rx_ CLK	RxD3	A3	V_{SS}	V_{DD}
D	PB1	PB3	PB5	V_{SS}	CRS	TxD1	Rx_ER	RxD2	A4	A8	A6	A7
E	PC7	V_{DD}	PB0	PB4	PA2	Tx_ER	RxD0	A5	A11	V_{SS}	V_{DD}	A10
F	PC3	PC4	PC5	V_{SS}	PB2	PA6	A9	A17	A15	A14	A13	A12
G	V_{SS}	PC0	PC1	PC2	PC6	PLL_ V_{SS}	V_{SS}	A23	A20	V_{SS}	V_{DD}	A16
H	X_{OUT}	X_{IN}	PLL_ V_{DD}	V_{DD}	PD7	TMS	V_{SS}	D5	V_{SS}	A21	A19	A18
J	V_{SS}	V_{DD}	LOOP FILT_ OUT	PD4	TRIGOUT	RTC_ V_{DD}	NMIn	WRn	D2	CS0n	V_{DD}	A22
K	PD5	PD6	PD3	TDI	V_{SS}	V_{DD}	RESETn	RDn	V_{DD}	D1	CS2n	CS1n
L	PD1	PD2	TRSTn	TCK	RTC_ XOUT	BUSACKn	WAITn	MREQn	D6	D4	D0	CS3n
M	PD0	V_{SS}	TDO	HALT _ SLPn	RTC_ XIN	BUSREQn	INSTRDn	IORQn	D7	D3	V_{SS}	V_{DD}

Note: Lowercase n suffix indicates an active-low signal in this table only

Figure 2 displays the pin layout of the eZ80F91 device in the 144-pin LQFP package.

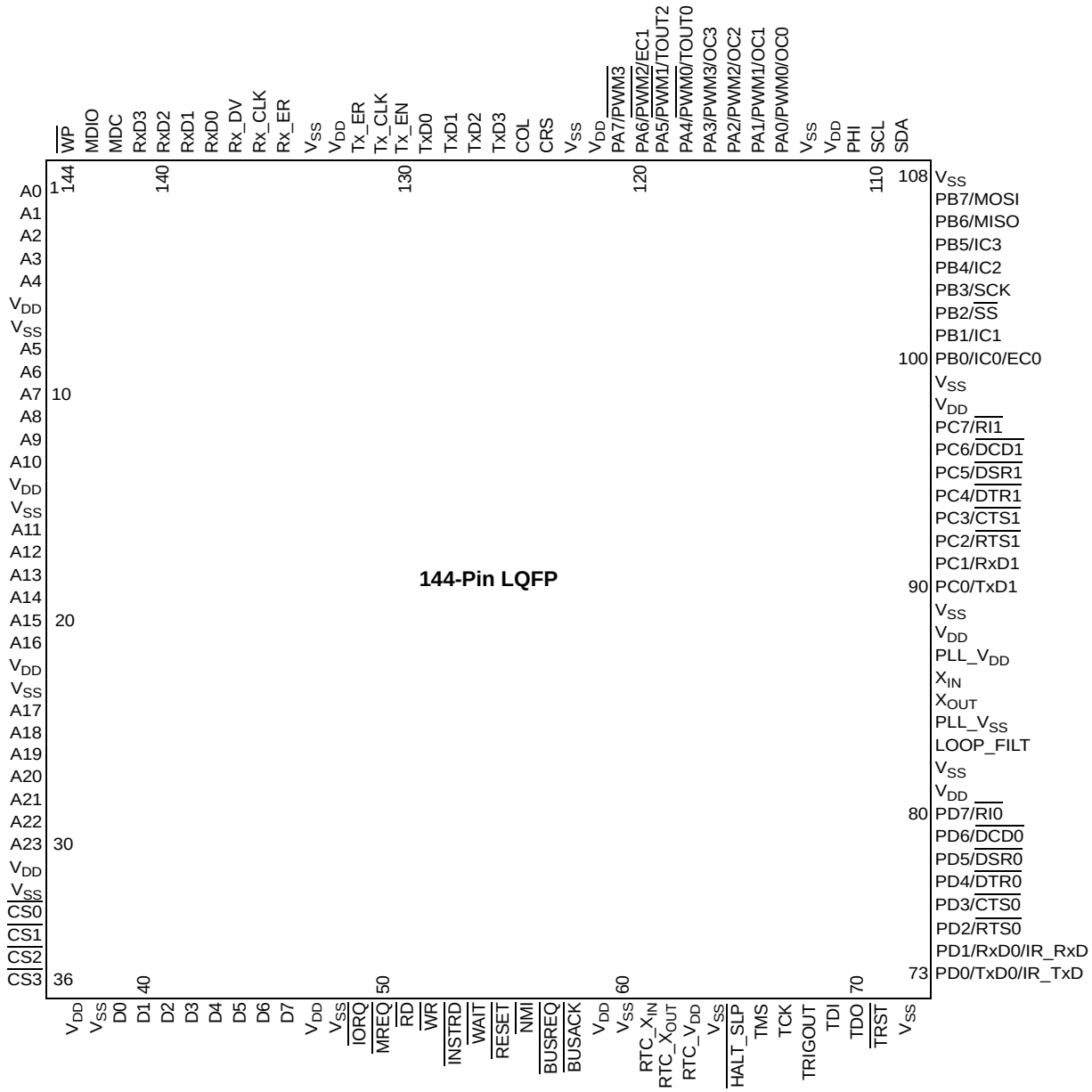


Figure 2. 144-Pin LQFP Configuration of the eZ80F91

Pin Characteristics

Table 2 lists the pins and functions of the eZ80F91 MCU's 144-pin LQFP package and 144-BGA package.

Table 2. Pin Identification on the eZ80F91 Device

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
1	A1	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	B1	ADDR1	Address Bus	Bidirectional	
3	B2	ADDR2	Address Bus	Bidirectional	
4	C3	ADDR3	Address Bus	Bidirectional	
5	D4	ADDR4	Address Bus	Bidirectional	
6	C1	V_{DD}	Power Supply		Power Supply.
7	C2	V_{SS}	Ground		Ground.
8	E5	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.
9	D2	ADDR6	Address Bus	Bidirectional	
10	D1	ADDR7	Address Bus	Bidirectional	
11	D3	ADDR8	Address Bus	Bidirectional	
12	F6	ADDR9	Address Bus	Bidirectional	
13	E1	ADDR10	Address Bus	Bidirectional	
14	E2	V_{DD}	Power Supply		Power Supply.
15	E3	V_{SS}	Ground		Ground.
16	E4	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.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
17	F1	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.
18	F2	ADDR13	Address Bus	Bidirectional	
19	F3	ADDR14	Address Bus	Bidirectional	
20	F4	ADDR15	Address Bus	Bidirectional	
21	G1	ADDR16	Address Bus	Bidirectional	
22	G2	V_{DD}	Power Supply		Power Supply.
23	G3	V_{SS}	Ground		Ground.
24	F5	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.
25	H1	ADDR18	Address Bus	Bidirectional	
26	H2	ADDR19	Address Bus	Bidirectional	
27	G4	ADDR20	Address Bus	Bidirectional	
28	H3	ADDR21	Address Bus	Bidirectional	
29	J1	ADDR22	Address Bus	Bidirectional	
30	G5	ADDR23	Address Bus	Bidirectional	
31	J2	V_{DD}	Power Supply		Power Supply.
32	H4	V_{SS}	Ground		Ground.
33	J3	CS0	Chip Select 0	Output, Active Low	$\overline{CS0}$ Low indicates that an access is occurring in the defined $\overline{CS0}$ memory or I/O address space.
34	K1	CS1	Chip Select 1	Output, Active Low	$\overline{CS1}$ Low indicates that an access is occurring in the defined $\overline{CS1}$ memory or I/O address space.
35	K2	CS2	Chip Select 2	Output, Active Low	$\overline{CS2}$ Low indicates that an access is occurring in the defined $\overline{CS2}$ memory or I/O address space.
36	L1	CS3	Chip Select 3	Output, Active Low	$\overline{CS3}$ Low indicates that an access is occurring in the defined $\overline{CS3}$ memory or I/O address space.
37	M1	V_{DD}	Power Supply		Power Supply.
38	M2	V_{SS}	Ground		Ground.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
39	L2	DATA0	Data Bus	Bidirectional	The data bus transfers data to and from I/O and memory devices. The eZ80F91 drives these lines only during Write cycles when the eZ80F91 is the bus master.
40	K3	DATA1	Data Bus	Bidirectional	
41	J4	DATA2	Data Bus	Bidirectional	
42	M3	DATA3	Data Bus	Bidirectional	
43	L3	DATA4	Data Bus	Bidirectional	
44	H5	DATA5	Data Bus	Bidirectional	
45	L4	DATA6	Data Bus	Bidirectional	
46	M4	DATA7	Data Bus	Bidirectional	
47	K4	V_{DD}	Power Supply		Power Supply.
48	G6	V_{SS}	Ground		Ground.
49	M5	$\overline{IORQ}$	Input/Output Request	Bidirectional, Active Low	$\overline{IORQ}$ indicates that the CPU is accessing a location in I/O space. $\overline{RD}$ and $\overline{WR}$ indicate the type of access. The eZ80F91 device does not drive this line during RESET. It is an input during bus acknowledge cycles.
50	L5	$\overline{MREQ}$	Memory Request	Bidirectional, Active Low	$\overline{MREQ}$ Low indicates that the CPU is accessing a location in memory. The $\overline{RD}$, $\overline{WR}$, and $\overline{INSTRD}$ signals indicate the type of access. The eZ80F91 device does not drive this line during RESET. It is an input during bus acknowledge cycles.
51	K5	$\overline{RD}$	Read	Output, Active Low	$\overline{RD}$ Low indicates that the eZ80F91 device is reading from the current address location. This pin is in a high-impedance state during bus acknowledge cycles.
52	J5	$\overline{WR}$	Write	Output, Active Low	$\overline{WR}$ indicates that the CPU is writing to the current address location. This pin is in a high-impedance state during bus acknowledge cycles.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
53	M6	$\overline{\text{INSTRD}}$	Instruction Read Indicator	Output, Active Low	$\overline{\text{INSTRD}}$ (with $\overline{\text{MREQ}}$ and $\overline{\text{RD}}$) indicates the eZ80F91 device is fetching an instruction from memory. This pin is in a high-impedance state during bus acknowledge cycles.
54	L6	$\overline{\text{WAIT}}$	WAIT Request	Schmitt-trigger 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.
55	K6	RESET	Reset	Bidirectional, Active Low Schmitt-trigger input or open drain output	This signal is used to initialize the eZ80F91, and/or allow the ez80F91 to signal when it resets. See reset section for the timing details. This Schmitt-trigger input allows for RC rise times.
56	J6	$\overline{\text{NMI}}$	Nonmaskable Interrupt	Schmitt-trigger input, Active Low, edge-triggered interrupt	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 for RC rise times.
57	M7	BUSREQ	Bus Request	Schmitt-trigger input, Active Low	External devices request the eZ80F91 device to release the memory interface bus for their use by driving this pin Low.
58	L7	$\overline{\text{BUSACK}}$	Bus Acknowledge	Output, Active Low	The eZ80F91 device responds to a Low on $\overline{\text{BUSREQ}}$ making the address, data, and control signals high impedance, 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.
59	K7	V_{DD}	Power Supply		Power Supply.
60	H6	V_{SS}	Ground		Ground.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
61	M8	RTC_X_{IN}	Real-Time Clock Crystal Input	Input	This pin is the input to the low-power 32 kHz crystal oscillator for the Real-time clock. If the Real-time clock is disabled or not used, this input must be left floating or tied to VSS to minimize any input current leakage.
62	L8	RTC_X_{OUT}	Real-Time Clock Crystal Output	Bidirectional	This pin is the output from the low-power 32 kHz 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.
63	J7	RTC_V_{DD}	Real-Time Clock Power Supply		Power supply for the Real-Time Clock and associated 32 kHz oscillator. Isolated from the power supply to the remainder of the chip. A battery is connected to this pin to supply constant power to the Real-Time Clock and 32 kHz oscillator. If the Real-time clock is disabled or not used this output must be tied to Vdd.
64	K8	V_{SS}	Ground		Ground.
65	M9	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.
66	H7	TMS	JTAG Test Mode Select	Input	JTAG Mode Select Input.
67	L9	TCK	JTAG Test Clock	Input	JTAG and ZDI clock input.
68	J8	TRIGOUT	JTAG Test Trigger Output	Output	Active High trigger event indicator.
69	K9	TDI	JTAG Test Data In	Bidirectional	JTAG data input pin. Functions as ZDI data I/O pin when JTAG is disabled. This pin has an internal pull-up resistor in the pad.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
70	M10	TDO	JTAG Test Data Out	Output	JTAG data output pin.
71	L10	TRST	JTAG Reset	Schmitt-trigger input, Active Low	JTAG reset input pin.
72	M11	V_{SS}	Ground		Ground.
73	M12	PD0	GPIO Port D	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port D pin, when programmed as output is 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.
74	L12	PD1	GPIO Port D	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port D pin, when programmed as output is 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.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
75	L11	PD2	GPIO Port D	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port D pin, when programmed as output is selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
		RTS0	Request to Send	Output, Active Low	Modem control signal from UART. This signal is multiplexed with PD2.
76	K10	PD3	GPIO Port D	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port D pin, when programmed as output is selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
		CTS0	Clear to Send	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PD3.
77	J9	PD4	GPIO Port D	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port D pin, when programmed as output is selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
		DTR0	Data Terminal Ready	Output, Active Low	Modem control signal to the UART. This signal is multiplexed with PD4.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
78	K12	PD5	GPIO Port D	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port D pin, when programmed as output is selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
		DSR0	Data Set Ready	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PD5.
79	K11	PD6	GPIO Port D	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port D pin, when programmed as output is selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
		DCD0	Data Carrier Detect	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PD6.
80	H8	PD7	GPIO Port D	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port D pin, when programmed as output is selected to be an open-drain or open-source output. Port D is multiplexed with one UART.
		RI0	Ring Indicator	Input, Active Low	Modem status signal to the UART. This signal is multiplexed with PD7.
81	J11	V_{DD}	Power Supply		Power Supply.
82	J12	V_{SS}	Ground		Ground.
83	J10	LOOP_FILT	PLL Loop Filter	Analog	Loop Filter pin for the Analog PLL.
84	G7	PLL_V_{SS}	Ground		Ground for Analog PLL.
85	H12	X_{OUT}	System Clock Oscillator Output	Output	This pin is the output of the onboard crystal oscillator. When used, a crystal must be connected between X_{IN} and X_{OUT}.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
86	H11	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 must be connected to this pin. When a crystal is used, it must be connected between X_{IN} and X_{OUT}.
87	H10	PLL_V_{DD}	Power Supply		Power Supply for Analog PLL.
88	H9	V_{DD}	Power Supply		Power Supply.
89	G12	V_{SS}	Ground		Ground.
90	G11	PC0	GPIO Port C	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port C pin, when programmed as output is 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.
91	G10	PC1	GPIO Port C	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port C pin, when programmed as output is selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
		RxD1	Receive Data	Schmitt-trigger input	This pin is used by the UART to receive asynchronous serial data. This signal is multiplexed with PC1.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
92	G9	PC2	GPIO Port C	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port C pin, when programmed as output is selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
		RTS1	Request to Send	Output, Active Low	Modem control signal from UART. This signal is multiplexed with PC2.
93	F12	PC3	GPIO Port C	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port C pin, when programmed as output is selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
		CTS1	Clear to Send	Schmitt-trigger input, Active Low	Modem status signal to the UART. This signal is multiplexed with PC3.
94	F11	PC4	GPIO Port C	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port C pin, when programmed as output is selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
		DTR1	Data Terminal Ready	Output, Active Low	Modem control signal to the UART. This signal is multiplexed with PC4.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
95	F10	PC5	GPIO Port C	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port C pin, when programmed as output is selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
		DSR1	Data Set Ready	Schmitt-trigger input, Active Low	Modem status signal to the UART. This signal is multiplexed with PC5.
96	G8	PC6	GPIO Port C	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port C pin, when programmed as output is selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
		DCD1	Data Carrier Detect	Schmitt-trigger input, Active Low	Modem status signal to the UART. This signal is multiplexed with PC6.
97	E12	PC7	GPIO Port C	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port C pin, when programmed as output is selected to be an open-drain or open-source output. Port C is multiplexed with one UART.
		RI1	Ring Indicator	Schmitt-trigger input, Active Low	Modem status signal to the UART. This signal is multiplexed with PC7.
98	E11	V_{DD}	Power Supply		Power Supply.
99	F9	V_{SS}	Ground		Ground.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
100	E10	PB0	GPIO Port B	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port B pin, when programmed as output is selected to be an open-drain or open-source output.
		IC0	Input Capture	Schmitt-trigger input	Input Capture A Signal to Timer 1. This signal is multiplexed with PB0.
		EC0	Event Counter	Schmitt-trigger input	Event Counter Signal to Timer 1. This signal is multiplexed with PB0.
101	D12	PB1	GPIO Port B	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port B pin, when programmed as output is selected to be an open-drain or open-source output.
		IC1	Input Capture	Schmitt-trigger input	Input Capture B Signal to Timer 1. This signal is multiplexed with PB1.
102	F8	PB2	GPIO Port B	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port B pin, when programmed as output is selected to be an open-drain or open-source output.
		SS	SPI Slave Select	Schmitt-trigger input, Active Low	The slave select input line is used to select a slave device in SPI mode. This signal is multiplexed with PB2.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
103	D11	PB3	GPIO Port B	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port B pin, when programmed as output is selected to be an open-drain or open-source output.
		SCK	SPI Serial Clock	Bidirectional with Schmitt-trigger input	SPI serial clock. This signal is multiplexed with PB3.
104	E9	PB4	GPIO Port B	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port B pin, when programmed as output is selected to be an open-drain or open-source output.
		IC2	Input Capture	Schmitt-trigger input	Input Capture A Signal to Timer 3. This signal is multiplexed with PB4.
105	D10	PB5	GPIO Port B	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port B pin, when programmed as output is selected to be an open-drain or open-source output.
		IC3	Input Capture	Schmitt-trigger input	Input Capture B Signal to Timer 3. This signal is multiplexed with PB5.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
106	C12	PB6	GPIO Port B	Bidirectional with Schmitt-trigger input	This pin is be used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port B pin, when programmed as output is selected to be an open-drain or open-source output.
		MISO	SPI Master-In/ Slave-Out	Bidirectional with Schmitt-trigger input	The MISO line is configured as an input when the eZ80F91 device is an SPI master device and as an output when eZ80F91 is an SPI slave device. This signal is multiplexed with PB6.
107	C11	PB7	GPIO Port B	Bidirectional with Schmitt-trigger input	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port B pin, when programmed as output is selected to be an open-drain or open-source output.
		MOSI	SPI Master Out Slave In	Bidirectional with Schmitt-trigger input	The MOSI line is configured as an output when the eZ80F91 device is an SPI master device and as an input when the eZ80F91 device is an SPI slave device. This signal is multiplexed with PB7.
108	B12	V_{SS}	Ground		Ground.
109	A12	SDA	I^2C Serial Data	Bidirectional	This pin carries the I^2C data signal.
110	A11	SCL	I^2C Serial Clock	Bidirectional	This pin is used to receive and transmit the I^2C clock.
111	B11	PHI	System Clock	Output	This pin is an output driven by the internal system clock. It is used by the system for synchronization with the eZ80F91 device.
112	C10	V_{DD}	Power Supply		Power Supply.
113	D9	V_{SS}	Ground		Ground.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
114	A10	PA0	GPIO Port A	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port A pin, when programmed as output is selected to be an open-drain or open-source output.
		PWM0	PWM Output 0	Output	This pin is used by Timer 3 for PWM 0. This signal is multiplexed with PA0.
		OC0	Output Compare 0	Output	This pin is used by Timer 3 for Output Compare 0. This signal is multiplexed with PA0.
115	B10	PA1	GPIO Port A	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port A pin, when programmed as output is selected to be an open-drain or open-source output.
		PWM1	PWM Output 1	Output	This pin is used by Timer 3 for PWM 1. This signal is multiplexed with PA1.
		OC1	Output Compare 1	Output	This pin is used by Timer 3 for Output Compare 1. This signal is multiplexed with PA1.
116	E8	PA2	GPIO Port A	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port A pin, when programmed as output is selected to be an open-drain or open-source output.
		PWM2	PWM Output 2	Output	This pin is used by Timer 3 for PWM 2. This signal is multiplexed with PA2.
		OC2	Output Compare 2	Output	This pin is used by Timer 3 for Output Compare 2. This signal is multiplexed with PA2.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
117	B9	PA3	GPIO Port A	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port A pin, when programmed as output is selected to be an open-drain or open-source output.
		PWM3	PWM Output 3	Output	This pin is used by Timer 3 for PWM 3. This signal is multiplexed with PA3.
		OC3	Output Compare 3	Output	This pin is used by Timer 3 for Output Compare 3 This signal is multiplexed with PA3.
118	A9	PA4	GPIO Port A	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port A pin, when programmed as output is selected to be an open-drain or open-source output.
		PWM0	PWM Output 0 Inverted	Output	This pin is used by Timer 3 for negative PWM 0. This signal is multiplexed with PA4.
		TOUT0	Timer Out	Output	This pin is used by Timer 0 timer-out signal. This signal is multiplexed with PA4.
119	C9	PA5	GPIO Port A	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port A pin, when programmed as output is selected to be an open-drain or open-source output.
		PWM1	PWM Output 1 Inverted	Output	This pin is used by Timer 3 for negative PWM 1. This signal is multiplexed with PA5.
		TOUT2	Timer Out	Output	This pin is used by the Timer 2 timer-out signal. This signal is multiplexed with PA5.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
120	F7	PA6	GPIO Port A	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port A pin, when programmed as output is selected to be an open-drain or open-source output.
		PWM2	PWM Output 2 Inverted	Output	This pin is used by Timer 3 for negative PWM 2. This signal is multiplexed with PA6.
		EC1	Event Counter	Input	Event Counter Signal to Timer 2. This signal is multiplexed with PA6.
121	A8	PA7	GPIO Port A	Bidirectional	This pin is used for GPIO. It is individually programmed as input or output and is also used individually as an interrupt input. Each Port A pin, when programmed as output is selected to be an open-drain or open-source output.
		PWM3	PWM Output 3 Inverted	Output	This pin is used by Timer 3 for negative PWM 3. This signal is multiplexed with PA7.
122	B8	V_{DD}	Power Supply		Power Supply.
123	C8	V_{SS}	Ground		Ground.
124	D8	CRS	MII Carrier Sense	Input	This pin is used by the EMAC for the MII Interface to the PHY (physical layer). Carrier Sense is an asynchronous signal.
125	A7	COL	MII Collision Detect	Input	This pin is used by the EMAC for the MII Interface to the PHY. Collision Detect is an asynchronous signal.
126	B7	TxD3	MII Transmit Data	Output	This pin is used by the EMAC for the MII Interface to the PHY. Transmit Data is synchronous to the rising-edge of Tx_CLK.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
127	C7	TxD2	MII Transmit Data	Output	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Transmit Data is synchronous to the rising-edge of Tx_CLK.
128	D7	TxD1	MII Transmit Data	Output	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Transmit Data is synchronous to the rising-edge of Tx_CLK.
129	A6	TxD0	MII Transmit Data	Output	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Transmit Data is synchronous to the rising-edge of Tx_CLK.
130	B6	Tx_EN	MII Transmit Enable	Output	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Transmit Enable is synchronous to the rising-edge of Tx_CLK.
131	C6	Tx_CLK	MII Transmit Clock	Input	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Transmit Clock is the Nibble or Symbol Clock provided by the MII PHY interface.
132	E7	Tx_ER	MII Transmit Error	Output	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Transmit Error is synchronous to the rising-edge of Tx_CLK.
133	A5	V_{DD}	Power Supply		Power Supply.
134	B5	V_{SS}	Ground		Ground.
135	D6	Rx_ER	MII Receive Error	Input	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Receive Error is provided by the MII PHY interface synchronous to the rising-edge of Rx_CLK.
136	C5	Rx_CLK	MII Receive Clock	Input	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Receive Clock is the Nibble or Symbol Clock provided by the MII PHY interface.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
137	A4	Rx_DV	MII Receive Data Valid	Input	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Receive Data Valid is provided by the MII PHY interface synchronous to the rising-edge of Rx_CLK.
138	E6	RxD0	MII Receive Data	Input	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Receive Data is provided by the MII PHY interface synchronous to the rising-edge of Rx_CLK.
139	B4	RxD1	MII Receive Data	Input	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Receive Data is provided by the MII PHY interface synchronous to the rising-edge of Rx_CLK.
140	D5	RxD2	MII Receive Data	Input	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Receive Data is provided by the MII PHY interface synchronous to the rising-edge of Rx_CLK.
141	C4	RxD3	MII Receive Data	Input	This pin is used by the Ethernet MAC for the MII Interface to the PHY. Receive Data is provided by the MII PHY interface synchronous to the rising-edge of Rx_CLK.
142	A3	MDC	MII Management Data Clock	Output	This pin is used by the Ethernet MAC for the MII Management Interface to the PHY. The Ethernet MAC provides the MII Management Data Clock to the MII PHY interface.

Table 2. Pin Identification on the eZ80F91 Device (Continued)

LQFP Pin No	BGA Pin No	Symbol	Function	Signal Direction	Description
143	B3	MDIO	MII Management Data	Bidirectional	This pin is used by the Ethernet MAC for the MII Management Interface to the PHY. The Ethernet MAC sends and receives the MII Management Data to and from the MII PHY interface.
144	A2	WP	Write Protect	Schmitt-trigger input, Active Low	The Write Protect input is used by the Flash Controller to protect the Boot Block from Write and ERASE operations.

System Clock Source Options

The following section describes the system clock source options.

System Clock—The eZ80F91 device's internal clock, SCLK, is responsible for clocking all internal logic. The SCLK source can be an external crystal oscillator, an internal PLL, or an internal 32 kHz RTC oscillator. The SCLK source is selected by PLL Control Register 0. RESET default is provided by the external crystal oscillator. For more details on CLK_MUX values in the PLL Control Register 0, see Table 154 on page 269.

PHI—PHI is a device output driven by SCLK that is used for system synchronization to the eZ80F91 device. PHI is used as the reference clock for all AC characteristics, see page 344.

External Crystal Oscillator—An externally-driven oscillator operates in two modes. In one mode, the X_{IN} pin is driven by a oscillator from DC up to 50 MHz when the X_{OUT} pin is not connected. In the other mode, the X_{IN} and X_{OUT} pins are driven by a crystal circuit.

Crystals recommended by Zilog® are defined to be a 50 MHz–3 overtone circuit or 1–10 MHz range fundamental for PLL operation. For details, see On-Chip Oscillators on page 335.

Real Time Clock—An internal 32 kHz real-time clock crystal oscillator driven by either the on-chip 32768 Hz crystal oscillator or a 50/60 Hz power-line frequency input. While intended for timekeeping, the RTC 32 kHz oscillator is selected as an SCLK. RTC_V_{DD} and RTC_V_{SS} provides an isolated power supply to ensure RTC operation in the event of loss of line power when a battery is provided. For more details, see On-Chip Oscillators on page 335.

PLL Clock—The eZ80F91 internal PLL driven by external crystals or external crystal oscillators in the range of 1 MHz to 10 MHz generates an SCLK up to 50 MHz. For more-details, see Phase-Locked Loop on page 265.

SCLK Source Selection Example

For additional SCLK source selection examples, refer to *Crystal Oscillator/Resonator Guidelines for eZ80® and eZ80Acclaim!® Devices Technical Note (TN0013)* available on www.zilog.com.

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] = XX). All I/O operations using 16-bit addresses within the 0000h–00FFh range 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 overlap the 0000h–00FFh address range.

Registers at unused addresses within the 0000h–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 lists the register map for the eZ80F91 device.

Table 3. Register Map

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
Product ID					
0000	ZDI_ID_L	eZ80® Product ID Low Byte Register	08	R	252
0001	ZDI_ID_H	eZ80 Product ID High Byte Register	00	R	252
0002	ZDI_ID_REV	eZ80 Product ID Revision Register	XX	R	252
Interrupt Priority					
0010	INT_P0	Interrupt Priority Register—Byte 0	00	R/W	61
0011	INT_P1	Interrupt Priority Register—Byte 1	00	R/W	61
0012	INT_P2	Interrupt Priority Register—Byte 2	00	R/W	61
0013	INT_P3	Interrupt Priority Register—Byte 3	00	R/W	61
0014	INT_P4	Interrupt Priority Register—Byte 4	00	R/W	61
0015	INT_P5	Interrupt Priority Register—Byte 5	00	R/W	61
Ethernet Media Access Controller					
0020	EMAC_TEST	EMAC Test Register	00	R/W	298
0021	EMAC_CFG1	EMAC Configuration Register	00	R/W	299
0022	EMAC_CFG2	EMAC Configuration Register	37	R/W	301
0023	EMAC_CFG3	EMAC Configuration Register	0F	R/W	302
0024	EMAC_CFG4	EMAC Configuration Register	00	R/W	303
0025	EMAC_STAD_0	EMAC Station Address—Byte 0	00	R/W	304

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
0026	EMAC_STAD_1	EMAC Station Address—Byte 1	00	R/W	304
0027	EMAC_STAD_2	EMAC Station Address—Byte 2	00	R/W	304
0028	EMAC_STAD_3	EMAC Station Address—Byte 3	00	R/W	304
0029	EMAC_STAD_4	EMAC Station Address—Byte 4	00	R/W	304
002A	EMAC_STAD_5	EMAC Station Address—Byte 5	00	R/W	304
002B	EMAC_TPTV_L	EMAC Transmit Pause Timer Value—Low Byte	00	R/W	305
002C	EMAC_TPTV_H	EMAC Transmit Pause Timer Value—High Byte	00	R/W	305
002D	EMAC_IPGT	EMAC Inter-Packet Gap	15	R/W	306
002E	EMAC_IPGR1	EMAC Non-Back-Back IPG	0C	R/W	308
002F	EMAC_IPGR2	EMAC Non-Back-Back IPG	12	R/W	308
0030	EMAC_MAXF_L	EMAC Maximum Frame Length—Low Byte	00	R/W	309
0031	EMAC_MAXF_H	EMAC Maximum Frame Length—High Byte	06	R/W	310
0032	EMAC_AFR	EMAC Address Filter Register	00	R/W	311
0033	EMAC_HTBL_0	EMAC Hash Table—Byte 0	00	R/W	312
0034	EMAC_HTBL_1	EMAC Hash Table—Byte 1	00	R/W	312
0035	EMAC_HTBL_2	EMAC Hash Table—Byte 2	00	R/W	312
0036	EMAC_HTBL_3	EMAC Hash Table—Byte 3	00	R/W	312
0037	EMAC_HTBL_4	EMAC Hash Table—Byte 4	00	R/W	312
0038	EMAC_HTBL_5	EMAC Hash Table—Byte 5	00	R/W	312
0039	EMAC_HTBL_6	EMAC Hash Table—Byte 6	00	R/W	312
003A	EMAC_HTBL_7	EMAC Hash Table—Byte 7	00	R/W	312
003B	EMAC_MIIMGT	EMAC MII Management Register	00	R/W	313
003C	EMAC_CTLD_L	EMAC PHY Configuration Data—Low Byte	00	R/W	314
003D	EMAC_CTLD_H	EMAC PHY Configuration Data—High Byte	00	R/W	315
003E	EMAC_RGAD	EMAC PHY Register Address Register	00	R/W	315

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
003F	EMAC_FIAD	EMAC PHY Unit Select Address Register	00	R/W	316
0040	EMAC_PTMR	EMAC Transmit Polling Timer Register	00	R/W	316
0041	EMAC_RST	EMAC Reset Control Register	20	R/W	317
0042	EMAC_TLBP_L	EMAC Transmit Lower Boundary Pointer—Low Byte	00	R/W	318
0043	EMAC_TLBP_H	EMAC Transmit Lower Boundary Pointer—High Byte	00	R/W	318
0044	EMAC_BP_L	EMAC Boundary Pointer—Low Byte	00	R/W	319
0045	EMAC_BP_H	EMAC Boundary Pointer—High Byte	C0	R/W	319
0046	EMAC_BP_U	EMAC Boundary Pointer—Upper Byte	FF	R/W	319
0047	EMAC_RHBP_L	EMAC Receive High Boundary Pointer—Low Byte	00	R/W	320
0048	EMAC_RHBP_H	EMAC Receive High Boundary Pointer—High Byte	00	R/W	321
0049	EMAC_RRP_L	EMAC Receive Read Pointer—Low Byte	00	R/W	321
004A	EMAC_RRP_H	EMAC Receive Read Pointer—High Byte	00	R/W	322
004B	EMAC_BUFSZ	EMAC Buffer Size Register	00	R/W	322
004C	EMAC_IEN	EMAC Interrupt Enable Register	00	R/W	323
004D	EMAC_ISTAT	EMAC Interrupt Status Register	00	R/W	325
004E	EMAC_PRSD_L	EMAC PHY Read Status Data—Low Byte	00	R/W	326
004F	EMAC_PRSD_H	EMAC PHY Read Status Data—High Byte	00	R/W	327
0050	EMAC_MIISTAT	EMAC MII Status Register	00	R/W	327
0051	EMAC_RWP_L	EMAC Receive Write Pointer—Low Byte	00	R/W	328
0052	EMAC_RWP_H	EMAC Receive Write Pointer—High Byte	00	R/W	329
0053	EMAC_TRP_L	EMAC Transmit Read Pointer—Low Byte	00	R/W	329

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
0054	EMAC_TRP_H	EMAC Transmit Read Pointer—High Byte	00	R/W	330
0055	EMAC_BLKSLFT_L	EMAC Receive Blocks Left Register—Low Byte	20	R/W	330
0056	EMAC_BLKSLFT_H	EMAC Receive Blocks Left Register—High Byte	00	R/W	331
0057	EMAC_FDATA_L	EMAC FIFO Data—Low Byte	XX	R/W	332
0058	EMAC_FDATA_H	EMAC FIFO Data—High Byte	0X	R/W	332
0059	EMAC_FFLAGS	EMAC FIFO Flags Register	33	R/W	333
PLL					
005C	PLL_DIV_L	PLL Divider Register—Low Byte	00	W	268
005D	PLL_DIV_H	PLL Divider Register—High Byte	00	W	269
005E	PLL_CTL0	PLL Control Register 0	00	R/W	269
005F	PLL_CTL1	PLL Control Register 1	00	R/W	271
Timers and PWM					
0060	TMR0_CTL	Timer 0 Control Register	00	R/W	132
0061	TMR0_IER	Timer 0 Interrupt Enable Register	00	R/W	133
0062	TMR0_IIR	Timer 0 Interrupt Identification Register	00	R/W	135
0063	TMR0_DR_L	Timer 0 Data Register—Low Byte	XX	R	136
	TMR0_RR_L	Timer 0 Reload Register—Low Byte	XX	W	138
0064	TMR0_DR_H	Timer 0 Data Register—High Byte	XX	R	137
	TMR0_RR_H	Timer 0 Reload Register—High Byte	XX	W	139
0065	TMR1_CTL	Timer 1 Control Register	00	R/W	132
0066	TMR1_IER	Timer 1 Interrupt Enable Register	00	R/W	133
0067	TMR1_IIR	Timer 1 Interrupt Identification Register	00	R/W	135
0068	TMR1_DR_L	Timer 1 Data Register—Low Byte	XX	R	136
	TMR1_RR_L	Timer 1 Reload Register—Low Byte	XX	W	138
0069	TMR1_DR_H	Timer 1 Data Register—High Byte	XX	R	137
	TMR1_RR_H	Timer 1 Reload Register—High Byte	XX	W	139

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
006A	TMR1_CAP_CTL	Timer 1 Input Capture Control Register	XX	R/W	139
006B	TMR1_CAPA_L	Timer 1 Capture Value A Register—Low Byte	XX	R/W	140
006C	TMR1_CAPA_H	Timer 1 Capture Value A Register—High Byte	XX	R/W	141
006D	TMR1_CAPB_L	Timer 1 Capture Value B Register—Low Byte	XX	R/W	141
006E	TMR1_CAPB_H	Timer 1 Capture Value B Register—High Byte	XX	R/W	142
006F	TMR2_CTL	Timer 2 Control Register	00	R/W	132
0070	TMR2_IER	Timer 2 Interrupt Enable Register	00	R/W	133
0071	TMR2_IIR	Timer 2 Interrupt Identification Register	00	R/W	135
0072	TMR2_DR_L	Timer 2 Data Register—Low Byte	XX	R	136
	TMR2_RR_L	Timer 2 Reload Register—Low Byte	XX	W	138
0073	TMR2_DR_H	Timer 2 Data Register—High Byte	XX	R	137
	TMR2_RR_H	Timer 2 Reload Register—High Byte	XX	W	139
0074	TMR3_CTL	Timer 3 Control Register	00	R/W	132
0075	TMR3_IER	Timer 3 Interrupt Enable Register	00	R/W	133
0076	TMR3_IIR	Timer 3 Interrupt Identification Register	00	R/W	135
0077	TMR3_DR_L	Timer 3 Data Register—Low Byte	XX	R	136
	TMR3_RR_L	Timer 3 Reload Register—Low Byte	XX	W	138
0078	TMR3_DR_H	Timer 3 Data Register—High Byte	XX	R	137
	TMR3_RR_H	Timer 3 Reload Register—High Byte	XX	W	139
0079	PWM_CTL1	PWM Control Register 1	00	R/W	153
007A	PWM_CTL2	PWM Control Register 2	00	R/W	154
007B	PWM_CTL3	PWM Control Register 3	00	R/W	156
	TMR3_CAP_CTL	Timer 3 Input Capture Control Register	00	R/W	139
007C	PWM0R_L	PWM 0 Rising-Edge Register—Low Byte	XX	R/W	157
	TMR3_CAPA_L	Timer 3 Capture Value A Register—Low Byte	XX	R/W	140

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
007D	PWM0R_H	PWM 0 Rising-Edge Register—High Byte	XX	R/W	157
	TMR3_CAPA_H	Timer 3 Capture Value A Register—High Byte	XX	R/W	141
007E	PWM1R_L	PWM 1 Rising-Edge Register—Low Byte	XX	R/W	157
	TMR3_CAPB_L	Timer 3 Capture Value B Register—Low Byte	XX	R/W	141
007F	PWM1R_H	PWM 1 Rising-Edge Register—High Byte	XX	R/W	157
	TMR3_CAPB_H	Timer 3 Capture Value B Register—High Byte	XX	R/W	142
0080	PWM2R_L	PWM 2 Rising-Edge Register—Low Byte	XX	R/W	157
	TMR3_OC_CTL1	Timer 3 Output Compare Control Register 1	00	R/W	132
0081	PWM2R_H	PWM 2 Rising-Edge Register—High Byte	XX	R/W	157
	TMR3_OC_CTL2	Timer 3 Output Compare Control Register 2	00	R/W	132
0082	PWM3R_L	PWM 3 Rising-Edge Register—Low Byte	XX	R/W	157
	TMR3_OC0_L	Timer 3 Output Compare 0 Value Register—Low Byte	XX	R/W	144
0083	PWM3R_H	PWM 3 Rising-Edge Register—High Byte	XX	R/W	157
	TMR3_OC0_H	Timer 3 Output Compare 0 Value Register—High Byte	XX	R/W	145
0084	PWM0F_L	PWM 0 Falling-Edge Register—Low Byte	XX	R/W	158
	TMR3_OC1_L	Timer 3 Output Compare 1 Value Register—Low Byte	XX	R/W	144

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
0085	PWM0F_H	PWM 0 Falling-Edge Register—High Byte	XX	R/W	158
	TMR3_OC1_H	Timer 3 Output Compare 1 Value Register—High Byte	XX	R/W	145
0086	PWM1F_L	PWM 1 Falling-Edge Register—Low Byte	XX	R/W	158
	TMR3_OC2_L	Timer 3 Output Compare 2 Value Register—Low Byte	XX	R/W	144
0087	PWM1F_H	PWM 1 Falling-Edge Register—High Byte	XX	R/W	158
	TMR3_OC2_H	Timer 3 Output Compare 2 Value Register—High Byte	XX	R/W	145
0088	PWM2F_L	PWM 2 Falling-Edge Register—Low Byte	XX	R/W	158
	TMR3_OC3_L	Timer 3 Output Compare 3 Value Register—Low Byte	XX	R/W	144
0089	PWM2F_H	PWM 2 Falling-Edge Register—High Byte	XX	R/W	158
	TMR3_OC3_H	Timer 3 Output Compare 3 Value Register—High Byte	XX	R/W	145
008A	PWM3F_L	PWM 3 Falling-Edge Register—Low Byte	XX	R/W	158
008B	PWM3F_H	PWM 3 Falling-Edge Register—High Byte	XX	R/W	158
Watchdog Timer					
0093	WDT_CTL	Watchdog Timer Control Register	08/28	R/W	117
0094	WDT_RR	Watchdog Timer Reset Register	XX	W	119
General-Purpose Input/Output Ports					
0096	PA_DR	Port A Data Register	XX	R/W	55
0097	PA_DDR	Port A Data Direction Register	FF	R/W	55
0098	PA_ALT1	Port A Alternate Register 1	00	R/W	56
0099	PA_ALT2	Port A Alternate Register 2	00	R/W	56

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
009A	PB_DR	Port B Data Register	XX	R/W	55
009B	PB_DDR	Port B Data Direction Register	FF	R/W	55
009C	PB_ALT1	Port B Alternate Register 1	00	R/W	56
009D	PB_ALT2	Port B Alternate Register 2	00	R/W	56
009E	PC_DR	Port C Data Register	XX	R/W	55
009F	PC_DDR	Port C Data Direction Register	FF	R/W	55
00A0	PC_ALT1	Port C Alternate Register 1	00	R/W	56
00A1	PC_ALT2	Port C Alternate Register 2	00	R/W	56
00A2	PD_DR	Port D Data Register	XX	R/W	55
00A3	PD_DDR	Port D Data Direction Register	FF	R/W	55
00A4	PD_ALT1	Port D Alternate Register 1	00	R/W	56
00A5	PD_ALT2	Port D Alternate Register 2	00	R/W	56
00A6	PA_ALT0	Port A Alternate Register 0	00	W	56
00A7	PB_ALT0	Port B Alternate Register 0	00	W	56
Chip Select/Wait State Generator					
00A8	CS0_LBR	Chip Select 0 Lower Bound Register	00	R/W	85
00A9	CS0_UBR	Chip Select 0 Upper Bound Register	FF	R/W	86
00AA	CS0_CTL	Chip Select 0 Control Register	E8	R/W	87
00AB	CS1_LBR	Chip Select 1 Lower Bound Register	00	R/W	85
00AC	CS1_UBR	Chip Select 1 Upper Bound Register	00	R/W	86
00AD	CS1_CTL	Chip Select 1 Control Register	00	R/W	87
00AE	CS2_LBR	Chip Select 2 Lower Bound Register	00	R/W	85
00AF	CS2_UBR	Chip Select 2 Upper Bound Register	00	R/W	86
00B0	CS2_CTL	Chip Select 2 Control Register	00	R/W	87
00B1	CS3_LBR	Chip Select 3 Lower Bound Register	00	R/W	85
00B2	CS3_UBR	Chip Select 3 Upper Bound Register	00	R/W	86
00B3	CS3_CTL	Chip Select 3 Control Register	00	R/W	87
Random Access Memory Control					

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
00B4	RAM_CTL	RAM Control Register	C0	R/W	94
00B5	RAM_ADDR_U	RAM Address Upper Byte Register	FF	R/W	95
00B6	MBIST_GPR	General-Purpose RAM MBIST Control	00	R/W	96
00B7	MBIST_EMR	Ethernet MAC RAM MBIST Control	00	R/W	96
Serial Peripheral Interface					
00B8	SPI_BRG_L	SPI Baud Rate Generator Register—Low Byte	02	R/W	207
00B9	SPI_BRG_H	SPI Baud Rate Generator Register—High Byte	00	R/W	207
00BA	SPI_CTL	SPI Control Register	04	R/W	208
00BB	SPI_SR	SPI Status Register	00	R	209
00BC	SPI_TSR	SPI Transmit Shift Register	XX	W	210
	SPI_RBR	SPI Receive Buffer Register	XX	R	210
Infrared Encoder/Decoder					
00BF	IR_CTL	Infrared Encoder/Decoder Control	00	R/W	199
Universal Asynchronous Receiver/Transmitter 0 (UART0)					
00C0	UART0_RBR	UART 0 Receive Buffer Register	XX	R	184
	UART0_THR	UART 0 Transmit Holding Register	XX	W	184
	UART0_BRG_L	UART 0 Baud Rate Generator Register—Low Byte	02	R/W	182
00C1	UART0_IER	UART 0 Interrupt Enable Register	00	R/W	185
	UART0_BRG_H	UART 0 Baud Rate Generator Register—High Byte	00	R/W	183
00C2	UART0_IIR	UART 0 Interrupt Identification Register	01	R	186
	UART0_FCTL	UART 0 FIFO Control Register	00	W	187
Universal Asynchronous Receiver/Transmitter 0 (UART0)					
00C3	UART0_LCTL	UART 0 Line Control Register	00	R/W	188
00C4	UART0_MCTL	UART 0 Modem Control Register	00	R/W	190
00C5	UART0_LSR	UART 0 Line Status Register	60	R	191
00C6	UART0_MSR	UART 0 Modem Status Register	XX	R	193

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
00C7	UART0_SPR	UART 0 Scratch Pad Register	00	R/W	194
I²C					
00C8	I2C_SAR	I²C Slave Address Register	00	R/W	224
00C9	I2C_XSAR	I²C Extended Slave Address Register	00	R/W	224
00CA	I2C_DR	I²C Data Register	00	R/W	225
00CB	I2C_CTL	I²C Control Register	00	R/W	226
General-Purpose Input/Output Ports					
00CE	PC_ALT0	Port C Alternate Register 0	00	W	56
00CF	PD_ALT0	Port D Alternate Register 0	00	W	56
00CC	I2C_SR	I²C Status Register	F8	R	227
	I2C_CCR	I²C Clock Control Register	00	W	229
00CD	I2C_SRR	I²C Software Reset Register	XX	W	230
Universal Asynchronous Receiver/Transmitter 1 (UART1)					
00D0	UART1_RBR	UART 1 Receive Buffer Register	XX	R	184
	UART1_THR	UART 1 Transmit Holding Register	XX	W	184
	UART1_BRG_L	UART 1 Baud Rate Generator Register—Low Byte	02	R/W	182
00D1	UART1_IER	UART 1 Interrupt Enable Register	00	R/W	185
	UART1_BRG_H	UART 1 Baud Rate Generator Register—High Byte	00	R/W	183
00D2	UART1_IIR	UART 1 Interrupt Identification Register	01	R	186
	UART1_FCTL	UART 1 FIFO Control Register	00	W	187
00D3	UART1_LCTL	UART 1 Line Control Register	00	R/W	188
Universal Asynchronous Receiver/Transmitter 0 (UART0)					
00D4	UART1_MCTL	UART 1 Modem Control Register	00	R/W	190
00D5	UART1_LSR	UART 1 Line Status Register	60	R/W	191
00D6	UART1_MSR	UART 1 Modem Status Register	XX	R/W	193
00D7	UART1_SPR	UART 1 Scratch Pad Register	00	R/W	194
Low-Power Control					

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
00DB	CLK_PPD1	Clock Peripheral Power-Down Register 1	00	R/W	47
00DC	CLK_PPD2	Clock Peripheral Power-Down Register 2	00	R/W	48
Real-Time Clock					
00E0	RTC_SEC	RTC Seconds Register	XX	R/W	161
00E1	RTC_MIN	RTC Minutes Register	XX	R/W	162
00E2	RTC_HRS	RTC Hours Register	XX	R/W	163
00E3	RTC_DOW	RTC Day-of-the-Week Register	0X	R/W	164
00E4	RTC_DOM	RTC Day-of-the-Month Register	XX	R/W	165
00E5	RTC_MON	RTC Month Register	XX	R/W	166
00E6	RTC_YR	RTC Year Register	XX	R/W	167
00E7	RTC_CEN	RTC Century Register	XX	R/W	168
00E8	RTC_ASEC	RTC Alarm Seconds Register	XX	R/W	169
00E9	RTC_AMIN	RTC Alarm Minutes Register	XX	R/W	170
00EA	RTC_AHRS	RTC Alarm Hours Register	XX	R/W	171
00EB	RTC_ADOW	RTC Alarm Day-of-the-Week Register	0X	R/W	172
00EC	RTC_ACTRL	RTC Alarm Control Register	00	R/W	173
00ED	RTC_CTRL	RTC Control Register	x0xxxx00b/ x0xxxx10b[4]	R/W	174
Chip Select Bus Mode Control					
00F0	CS0_BMC	Chip Select 0 Bus Mode Control Register	02	R/W	88
00F1	CS1_BMC	Chip Select 1 Bus Mode Control Register	02	R/W	88
00F2	CS2_BMC	Chip Select 2 Bus Mode Control Register	02	R/W	88
00F3	CS3_BMC	Chip Select 3 Bus Mode Control Register	02	R/W	88

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
Flash Memory Control					
00F5	FLASH_KEY	Flash Key Register	00	W	102
00F6	FLASH_DATA	Flash Data Register	XX	R/W	103
00F7	FLASH_ADDR_U	Flash Address Upper Byte Register	00	R/W	104
00F8	FLASH_CTL	Flash Control Register	88	R/W	105
00F9	FLASH_FDIV	Flash Frequency Divider Register	01	R/W	106
00FA	FLASH_PROT	Flash Write/Erase Protection Register	FF	R/W	107
00FB	FLASH_IRQ	Flash Interrupt Control Register	00	R/W	108
00FC	FLASH_PAGE	Flash Page Select Register	00	R/W	109
00FD	FLASH_ROW	Flash Row Select Register	00	R/W	111
00FE	FLASH_COL	Flash Column Select Register	00	R/W	112
00FF	FLASH_PGCTL	Flash Program Control Register	00	R/W	112

eZ80® CPU Core

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

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

Features

The features of eZ80 CPU include:

- 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 Arithmetic Logic Unit (ALU)
- Debug support
- Nonmaskable Interrupt (NMI), plus support for 128 maskable vectored interrupts

New Instructions

The new instructions are listed below:

- Loads/unloads the I register with a 16-bit value. These new instructions are:
  - LD I,HL (ED C7)
  - LD HL,I (ED D7)

For more information on the CPU, its instruction set, and eZ80 programming, refer to eZ80 *CPU User Manual (UM0077)*, available on www.zilog.com.

Reset

The Reset controller within the eZ80F91 device features a consistent reset function for all types of resets that affects the system. A system reset, referred in this document as RESET, returns the eZ80F91 to a defined state. All internal registers affected by a RESET return to their default conditions. RESET configures the GPIO port pins as inputs and clears the CPU's Program Counter to 000000h. Program code execution ceases during RESET.

The events that cause a RESET are:

- Power-on reset (POR).
- Low-Voltage Brownout (VBO).
- External $\overline{\text{RESET}}$ pin assertion.
- Watchdog Timer (WDT) time-out when configured to generate a RESET.
- Real-Time Clock alarm with the 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 1025 system clock (SCLK) cycles to allow sufficient time for the primary crystal oscillator to stabilize. For internal RESET sources, the RESET mode timer begins incrementing on the next rising edge of SCLK following deactivation of the signal that is initiating the RESET event. For external $\overline{\text{RESET}}$ pin assertion, the RESET mode timer begins on the next rising edge of SCLK following assertion of the $\overline{\text{RESET}}$ pin for three consecutive SCLK cycles.

Note: *The default clock source for SCLK on RESET is the crystal input (X_{IN}). See the CLK_MUX values in the PLL Control Register 0, (see Table 154 on page 269).*

External Reset Input and Indicator

The eZ80F91 $\overline{\text{RESET}}$ pin functions as both open-drain (active Low) RESET mode indicator and active Low $\overline{\text{RESET}}$ input. When a RESET event occurs, the internal circuitry begins driving the $\overline{\text{RESET}}$ pin Low. The $\overline{\text{RESET}}$ pin is held Low by the internal circuitry until the internal RESET mode timer times out. If the external reset signal is released prior to the end of the 1025 count time-out, program execution begins following the RESET mode time-out. If the external reset signal is released after the end of the 1025 count time-out, then program execution begins following release of the $\overline{\text{RESET}}$ input (the $\overline{\text{RESET}}$ pin is High for four consecutive SCLK cycles).

Power-On Reset

A POR occurs every time the supply voltage to the part rises from below the Voltage Brownout threshold (V_{VBO}) to above the POR voltage threshold (V_{POR}). The internal bandgap-referenced voltage detector sends a continuous RESET signal to the Reset controller until the supply voltage (V_{CC}) exceeds the POR voltage threshold. After V_{CC} rises above V_{POR}, an on-chip analog delay element briefly maintains the RESET signal to the Reset controller. After this analog delay element times out, the Reset controller holds the eZ80F91 in RESET until the RESET mode timer expires. POR operation is displayed in Figure 3. The signals in Figure 3 are not drawn to scale but for displaying purposes only.

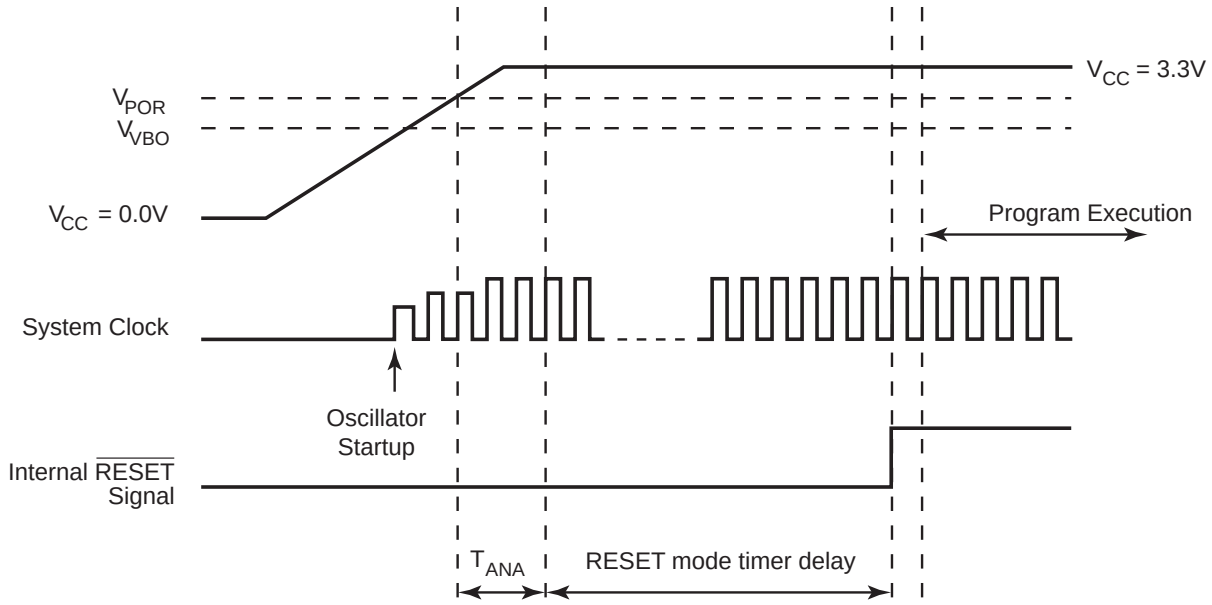


Figure 3. Power-On Reset Operation

Voltage Brownout Reset

If the supply voltage (V_{CC}) drops below the V_{VBO} after program execution begins, the eZ80F91 device resets. The VBO protection circuitry detects the low supply voltage and initiates a RESET via the Reset controller. The eZ80F91 remains in RESET until the supply voltage again returns above the POR voltage threshold (V_{POR}) and the Reset controller releases the internal RESET signal. The VBO circuitry rejects short negative brown-out pulses to prevent spurious RESET events.

VBO operation is displayed in Figure 4 on page 43. The signals in the figure are not drawn to scale but for illustration purposes only.

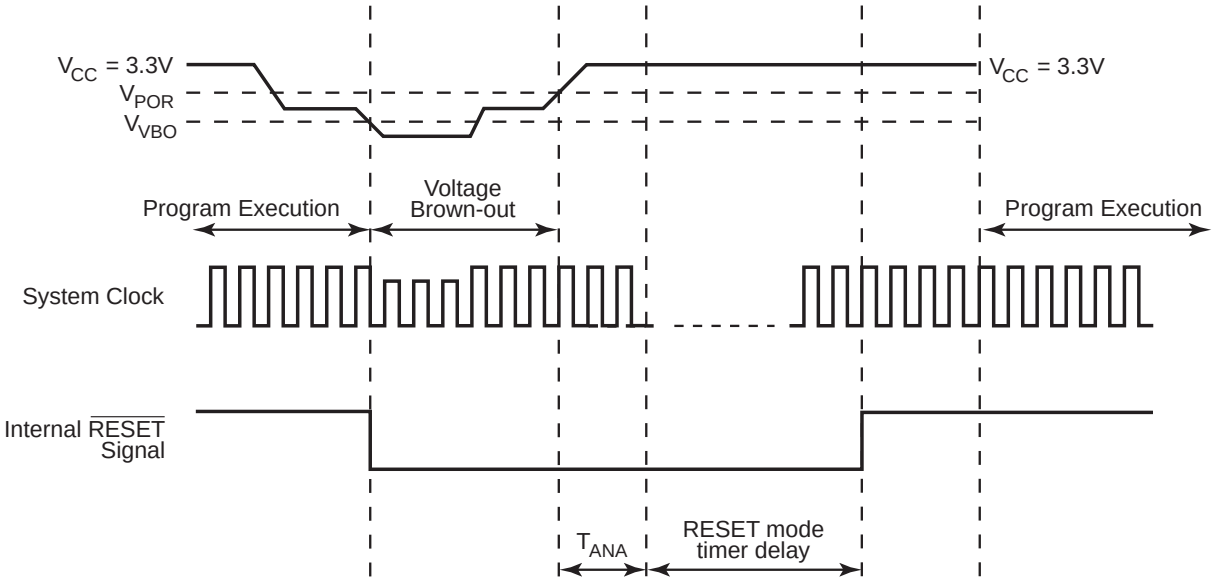


Figure 4. Voltage Brownout Reset Operation

Low-Power Modes

The eZ80F91 device provides a range of power-saving features. The highest level of power reduction is provided by SLEEP mode with all peripherals disabled, including VBO. The next level of power reduction is provided by the HALT instruction. The most basic level of power reduction is provided by the clock peripheral power-down registers.

SLEEP Mode

Execution of the CPU's SLP instruction puts the eZ80F91 device into SLEEP mode. In SLEEP mode, the operating characteristics are:

- The primary crystal oscillator is disabled.
- The system clock is disabled.
- The CPU is idle.
- The Program Counter (PC) stops incrementing.
- The 32 kHz crystal oscillator continues to operate and drives the real-time clock and WDT (if WDT is configured to operate from the 32 kHz oscillator).

The CPU is brought out of SLEEP mode by any of the following operations:

- A RESET via the external $\overline{\text{RESET}}$ pin driven Low.
- A RESET via a real-time clock alarm.
- A RESET via a WDT time-out (if running out of the 32 kHz oscillator and configured to generate a RESET on time-out).
- A RESET via execution of a Debug RESET command.
- A RESET via the Low-Voltage Brownout (VBO) detection circuit, if enabled.

After exiting SLEEP mode, the standard RESET delay occurs to allow the primary crystal oscillator to stabilize. For more information, see Figure 4 on page 43.

HALT Mode

Execution of the CPU's HALT instruction puts the eZ80F91 device into HALT mode. In HALT mode, the operating characteristics are:

- The primary crystal oscillator is enabled and continues to operate.
- The system clock is enabled and continues to operate.
- The CPU is idle.

- The PC stops incrementing.

The CPU is brought out of HALT mode by any of the following operations:

- A nonmaskable interrupt (NMI).
- A maskable interrupt.
- A RESET via the external $\overline{\text{RESET}}$ pin driven Low.
- A Watchdog Timer time-out (if, configured to generate either an NMI or RESET upon time-out).
- A RESET via execution of a Debug RESET command.
- A RESET via the Low-Voltage Brownout detection circuit, if enabled.

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

HALT Mode and the EMAC Function

When the CPU is in HALT mode, the eZ80F91 device's EMAC block cannot be disabled as other peripherals can. On receipt of an Ethernet packet, a maskable Receive interrupt is generated by the EMAC block, just as it would be in a non-halt mode. Accordingly, the processor wakes up and continues with the user-defined application.

Clock Peripheral Power-Down Registers

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

Additionally, the VBO_OFF bit of CLK_PPD2 is used to disable the VBO detection circuit and thereby significantly reduce DC current consumption (see Table 234 on page 341) when this function is not required.

Many peripherals features 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 individual peripheral control register is not accessible for Read or Write access, (see Table 4 on page 47 and Table 5 on page 48).

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

Note: R/W = Read/Write.

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 GPIO_A_OFF	1	System clock to GPIO Port A is powered down. Port A alternate functions do not operate correctly.
	0	System clock to GPIO Port A is powered up.
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/W	R	R	R/W	R/W	R/W	R/W

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

Bit Position	Value	Description
7 PHI_OFF	1	PHI Clock output is disabled (output is high-impedance).
	0	PHI Clock output is enabled.
6 VBO_OFF	1	Voltage Brownout detection circuit is disabled. This reduces DC current consumption in situations where VBO detection is not necessary. Power-On Reset functionality is not affected by this setting.
	0	VBO detection circuit is enabled.
[5:4]	000	Reserved.
3 TIMER3_OFF	1	System clock to TIMER3 is powered down.
	0	System clock to TIMER3 is powered up.
2 TIMER2_OFF	1	System clock to TIMER2 is powered down.
	0	System clock to TIMER2 is powered up.
1 TIMER1_OFF	1	System clock to TIMER1 is powered down.
	0	System clock to TIMER1 is powered up.
0 TIMER0_OFF	1	System clock to TIMER0 is powered down.
	0	System clock to TIMER0 is powered up.

General-Purpose Input/Output

The eZ80F91 device features 32 General-Purpose Input/Output (GPIO) pins. The GPIO pins are assembled as four 8-bit ports—Port A, Port B, Port C, and Port D. All port signals are configured as either inputs or outputs. In addition, all the port pins are used as vectored interrupt sources for the CPU.

The eZ80F91 microcontroller's GPIO ports are slightly different from its eZ80® predecessors. Specifically, Port A pins source 8 mA and sink 10 mA. In addition, the Port B and C inputs now feature Schmitt-trigger input buffers.

GPIO Operation

GPIO operation is the same for all four GPIO ports (Ports A, 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. The 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 A, B, C, or D representing any of the four GPIO ports. 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 on page 50 lists the function of each port signal based on these four register bits. After a RESET event, all GPIO port pins are configured as standard digital inputs with the interrupts disabled.

In addition to the four mode control registers, each port has an 8-bit register, which is used for clearing edge triggered interrupts. This register is the Port x Alternate register 0(Px_ALT0) where x can be A, B, C, or D representing the four GPIO ports. When a GPIO pin is configured as an edge triggered interrupt, writing 1 to the corresponding bit of the Px_ALT0 register clears the interrupt.

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	Alternate function controls port I/O.	
	1	0	1	1	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

Figure 5 on page 53 and Figure 6 on page 53 display the simplified block diagrams of the GPIO port pin for the various modes.

GPIO Mode 1—Output

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

GPIO Mode 2—Input

The port pin is configured as a standard digital input pin. The output is 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—Open Drain

The port pin is configured as open-drain Input/Output. 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 0 to the Port *x* Data register outputs a Low at the pin. Writing 1 to the Port *x* Data register results in high-impedance output.

GPIO Mode 4—Open Source

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 1 to the Port *x* Data register outputs a High at the pin. Writing 0 to the Port *x* Data register results in a high-impedance output.

GPIO Mode 5—Reserved

This mode produces a high-impedance output.

GPIO Mode 6—Dual Edge Triggered

The port pin is configured for dual edge-triggered interrupt mode. Both a rising and a falling edge on this pin cause an interrupt request to be sent to the CPU. To select this mode from the default mode (mode 2), you must:

1. Set Px_DR=1
2. Set Px_ALT2=1
3. Set Px_ALT1=0
4. Set Px_DDR=0

Writing a 1 to the Port x ALT0 register bit position corresponding to the interrupt request clears the interrupt.

GPIO Mode 7—Alternate Functions

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 PC5 is the $\overline{\text{DSR1}}$ input signal to UART1 and the alternate mode function for PB4 is the timer 3 input capture. When GPIO mode 7 is enabled, the pin output data and pin high-impedance control is obtained from the alternate function's data output and high-impedance control, respectively. The value in the Port *x* Data register produces no effect on operation. Input signals are sampled by the system clock before being passed to the alternate input function.

If the alternate function of a pin is an input and alternate function mode for that pin is not enabled, the input is driven to a default non-asserted value. For example, in alternate mode function, PC5 drives the $\overline{\text{DSR1}}$ signal to UART1. As this signal is Low level true, the $\overline{\text{DSR1}}$ signal to UART1 is driven to 1 when PC5 is not in alternate mode function.

GPIO Mode 8—Level Sensitive Interrupt

The port pin is configured for level-sensitive interrupt mode. The value in the Port x Data register determines if a low or high-level causes an interrupt request. 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 two system clock periods to initiate an interrupt. The interrupt request remains active as long as this condition is maintained at the external source. For example, if a port pin is configured as a low-level-sensitive interrupt, the interrupt request will be asserted when the pin has been low for two system clocks and remains active until the pin goes high.

Configuring a pin for mode 8 requires a transition through mode 9 (edge triggered mode). To avoid the possibility of an unwanted interrupt while transition through mode 9, the following steps must be taken to select mode 8 when starting from the default mode (mode 2):

1. Disable interrupts
2. Set Px_DR = 0 (low level interrupt) or 1 (high level interrupt)
3. Set Px_ALT2 = 1
4. Set Px_ALT1 =1 (mode 9)
5. Set Px_DDR = 0 (mode 8)
6. Set Px_ALT0 = 1 (to clear possible mode 9 interrupt)
7. Enable interrupts

GPIO Mode 9—Edge Triggered Interrupt

The port pin is configured for single edge triggered interrupt mode. The value in the Port x Data register determines whether a positive or negative edge causes an interrupt request. Writing 0 to the Port *x* Data register bit sets the selected pin to generate an interrupt request for falling edges. Writing 1 to the Port *x* Data register bit sets the selected pin to generate an interrupt request for rising edges. The interrupt request remains active until 1 is written to the corresponding bit of the Port x Alternate register 0. To select mode 9 from the default mode (mode 2), you must:

1. Set the Port x Data register
2. Set Px_ALT2 = 1
3. Set Px_ALT1 = 1
4. Set Px_DDR=1

Simplified GPIO Port Block Diagram for Modes 2, 6, 7(input), 8, and 9

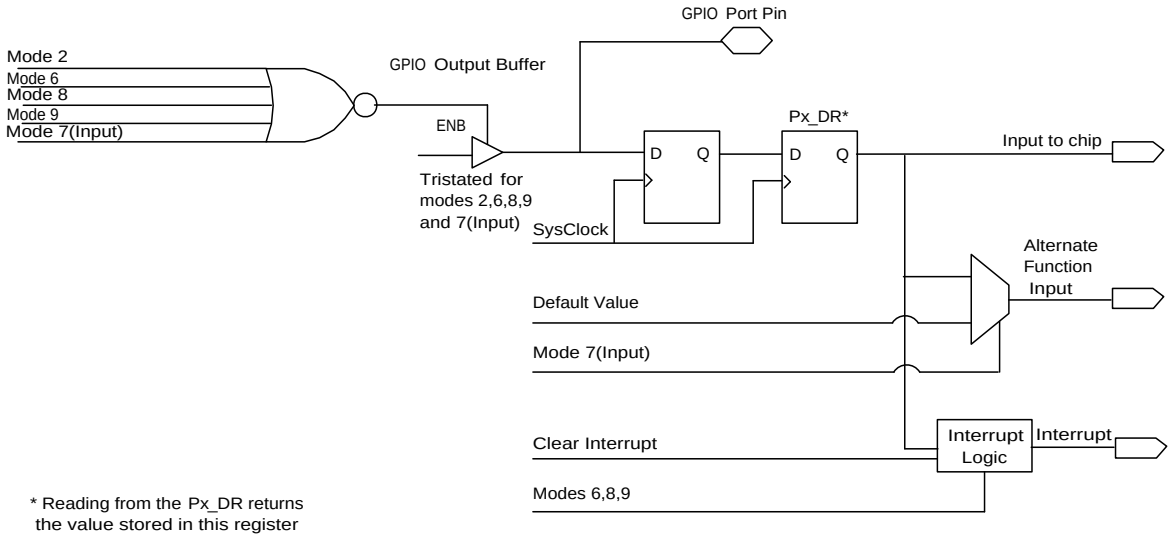


Figure 5. GPIO Port Pin Block Diagram for Input and Interrupt Modes

Simplified GPIO Port Block Diagram for Modes 1, 3, 4, and 7 (Output)

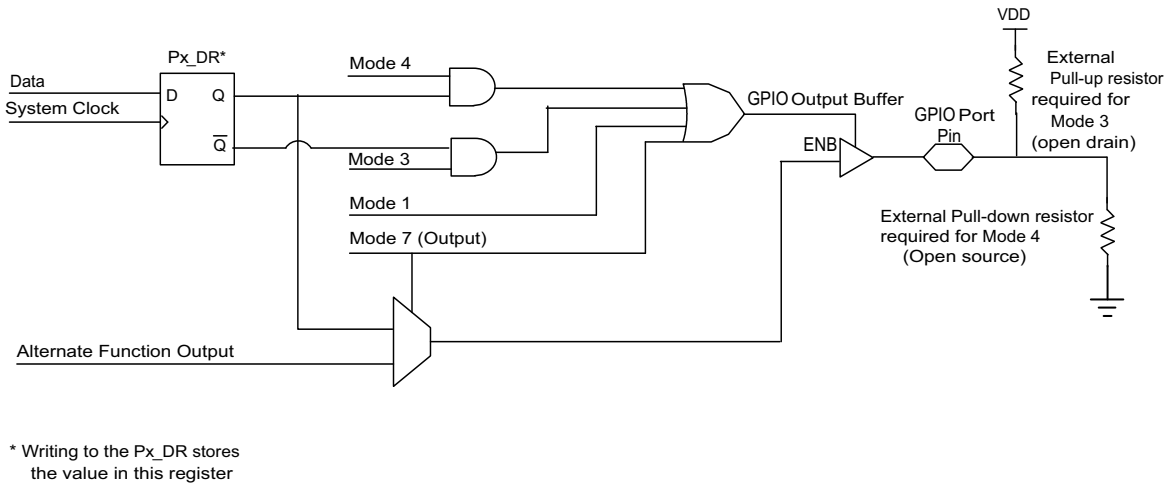


Figure 6. GPIO Port Pin Block Diagram for Output and Input/Output Mode

GPIO Interrupts

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

Level-Triggered Interrupts

When the port is configured for level-triggered interrupts (mode 8), the corresponding port pin is open-drain. 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 two clock periods to initiate an interrupt. The interrupt request remains active as long as this condition is maintained at the external source.

For example, if PA3 is programmed for low-level interrupt and the pin is forced Low for two clock cycles, an interrupt request signal is generated from that port pin and sent to the CPU. The interrupt request signal remains active until the external device driving PA3 forces the pin high. The CPU must be enabled to respond to interrupts for the interrupt request signal to be acted upon.

Edge Triggered Interrupts

When the port is configured for edge triggered interrupts, the corresponding port pin is open-drain. If the pin receives the correct edge from an external device, the port pin generates an interrupt request signal to the CPU.

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 CPU. To select mode 6 from the default mode (mode 2), you must:

1. Set Px_DR = 1
2. Set Px_ALT2 =1
3. Set Px_ALT1= 0
4. Set Px_DDR = 0

When configured for single-edge triggered interrupt mode (GPIO mode 9), the value in the Port x Data register determines whether a positive or negative edge causes an interrupt request. 0 in the Port x Data register bit sets the selected pin to generate an interrupt request for falling edges. 1 in the Port x Data register bit sets the selected pin to generate an interrupt request for rising edges. To select mode 9 from the default mode (mode 2), you must:

1. Set Px_DR = 1
2. Set Px_ALT2 = 1
3. Set Px_ALT = 1
4. Set Px_DDR = 1

Edge triggered interrupts are cleared by writing 1 to the corresponding bit of the Px_ALT0 register. For example, if PD4 has been set up to generate an edge triggered interrupt, the interrupt is cleared by writing a 1 to Px_ALT0[4].

GPIO Control Registers

Each GPIO port has four registers that controls its operation. The operating mode of each bit within a port is selected by writing to the corresponding bits of these four registers as listed in Table 6 on page 50. These four registers are Port Data register (Px_DR), Port Data Direction register (Px_DDR), Port Alternate register 1 (PX_ALT1), and Port Alternate register 2 (Px_ALT2). In addition to these four control registers, each port has a Port Alternate register 0 (Px_ALT0), which is used for clearing edge triggered interrupts.

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 (see Table 7) is driven on the corresponding pins. In all modes, reading from the Port x Data registers always returns the sampled current value of the corresponding pins. When the port pins are configured for edge triggered interrupts or level-sensitive interrupts, the value written to the Port *x* Data register bit selects the interrupt edge or interrupt level (for more details on GPIO mode selection, see Table 6 on page 50).

Table 7. Port x Data Registers
(PA_DR = 0096h, 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 (see Table 8) control the operating modes of the GPIO port pins. For more details on GPIO mode selection, see Table 6 on page 50.

Table 8. Port x Data Direction Registers
(PA_DDR = 0097h, 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 0

The Port x Alternate register 0 is used to clear edge triggered interrupts. If an edge triggered interrupt occurs, writing 1 to the corresponding bit of this register will clear it.

Table 9. Port x Alternate Registers 0
(PA_ALT0 = 00A6h, PB_ALT0 = 00A7h, PC_ALT0 = 00CEh, PD_ALT0 = 00CFh)

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								

Port x Alternate Register 1

In conjunction with the other GPIO Control registers, the Port *x* Alternate Register 1 (see Table 10) controls the operating modes of the GPIO port pins. For more details on GPIO mode selection, see Table 6 on page 50.

Table 10. Port *x* Alternate Registers 1
(PA_ALT1 = 0098h, 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 (see Table 11) controls the operating modes of the GPIO port pins. For more details on GPIO mode selection, see Table 6 on page 50.

Table 11. Port x Alternate Registers 2
(PA_ALT2 = 0099h, 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 eZ80F91 device routes the interrupt request signals from the internal peripherals, external devices (via the internal port I/O), and the nonmaskable interrupt (NMI) pin to the CPU.

Maskable Interrupts

On the eZ80F91 device, all maskable interrupts use the CPU's vectored interrupt function. The size of I register is modified to 16 bits in the eZ80F91 device differing from the previous versions of eZ80® CPU, to allow for a 16 MB range of interrupt vector table placement. Additionally, the size of the IVECT register is increased from 8 bits to 9 bits to provide an interrupt vector table that is expanded and more easily integrated with other interrupts.

The vectors are 4 bytes (32 bits) apart, even though only 3 bytes (24 bits) are required. A fourth byte is implemented for both programmability and expansion purposes.

Starting the interrupt vectors at 40h allows for easy implementation of the interrupt controller vectors with the RST vectors. Table 12 lists the interrupt vector sources by priority for each of the maskable interrupt sources. The maskable interrupt sources are listed in order of their priority, with vector 40h being the highest-priority interrupt. In ADL mode, the full 24-bit interrupt vector is located at starting address {I[15:1], IVECT[8:0]}, where I[15:0] is the CPU's Interrupt Page Address register.

Table 12. Interrupt Vector Sources by Priority

Priority	Vector	Source	Priority	Vector	Source
0	040h	EMAC Rx	24	0A0h	Port B 0
1	044h	EMAC Tx	25	0A4h	Port B 1
2	048h	EMAC SYS	26	0A8h	Port B 2
3	04Ch	PLL	27	0ACh	Port B 3
4	050h	Flash	28	0B0h	Port B 4
5	054h	Timer 0	29	0B4h	Port B 5
6	058h	Timer 1	30	0B8h	Port B 6
7	05Ch	Timer 2	31	0BCh	Port B 7
8	060h	Timer 3	32	0C0h	Port C 0
9	064h	Unused*	33	0C4h	Port C 1
10	068h	Unused*	34	0C8h	Port C 2

Table 12. Interrupt Vector Sources by Priority (Continued)

Priority	Vector	Source	Priority	Vector	Source
11	06Ch	RTC	35	0CCh	Port C 3
12	070h	UART 0	36	0D0h	Port C 4
13	074h	UART 1	37	0D4h	Port C 5
14	078h	I^2C	38	0D8h	Port C 6
15	07Ch	SPI	39	0DCh	Port C 7
16	080h	Port A 0	40	0E0h	Port D 0
17	084h	Port A 1	41	0E4h	Port D 1
18	088h	Port A 2	42	0E8h	Port D 2
19	08Ch	Port A 3	43	0ECh	Port D 3
20	090h	Port A 4	44	0F0h	Port D 4
21	094h	Port A 5	45	0F4h	Port D 5
22	098h	Port A 6	46	0F8h	Port D 6
23	09Ch	Port A 7	47	0FCh	Port D 7

Note: *The vector addresses 064h and 068h are left unused to avoid conflict with the nonmaskable interrupt (NMI) address 066h. The NMI is prioritized higher than all maskable interrupts.

The program must store the interrupt service routine starting address in the four-byte interrupt vector locations. For example in ADL mode, the three-byte address for the SPI interrupt service routine is stored at {I[15:1], 07Ch}, {I[15:1], 07Dh}, and {I[15:1], 07Eh}. In Z80® mode, the two-byte address for the SPI interrupt service routine is stored at {MBASE[7:0], I[7:1], 07Ch} and {MBASE, I[7:1], 07Dh}. The LSB is stored at the lower address.

When one or more interrupt requests (IRQs) become active, an interrupt request is generated by the interrupt controller and sent to the CPU. The corresponding 9-bit interrupt vector for the highest-priority interrupt is placed on the 9-bit interrupt vector bus, IVECT[8:0]. The interrupt vector bus is internal to the eZ80F91 device and is therefore externally not visible. The response time of the 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[15:1], IVECT[8:0]} is visible on the address bus (ADDR[23:0]), when the interrupt service routine begins. The response of the CPU to a vectored interrupt on the eZ80F91 device is listed in Table 13 on page 59. Interrupt sources are required to be active until the Interrupt Service Routine (ISR) starts.

Note: *The lower bit of the I register is replaced with the MSB of the IVECT from the interrupt controller. As a result, the interrupt vector table is required to be placed onto a 512-byte*

boundary. Setting the LSB of the I register produces no effect on the interrupt vector address.

Table 13. 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 [8:0], by the interrupting peripheral. IEF1 ← 0 IEF2 ← 0 The Starting Program Counter is effective {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[7:1], IVECT[8:0] }. PC[23:0] ← ({ MBASE, I[7:1], IVECT[8:0] }). 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 [8:0], 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 { I[15:1], IVECT[8:0] }. PC[23:0] ← ({ I[15:1], IVECT[8:0] }). The interrupt service routine must end with RETI.
Z80 Mode	0	1	**Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT[8:0], bus by the interrupting peripheral.** • **IEF1 ← 0** • **IEF2 ← 0** • The Starting Program Counter is effective {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 islocated at { I[15:1], IVECT[8:0] }. • PC[23:0]← **({ I[15:1], IVECT[8:0] })**. • The interrupt service routine must end with RETI.L

Table 13. Vectored Interrupt Operation (Continued)

Memory Mode	ADL Bit	MADL Bit	Operation
ADL Mode	1	1	**Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT [8:0], 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 {I[15:1], IVECT[8:0]}. • PC[23:0]← **({ I[15:1], IVECT[8:0] })**. • The interrupt service routine must end with RETI.L

Interrupt Priority Registers

The eZ80F91 provides two interrupt priority levels for the maskable interrupts. The default priority (or Level 0) is listed in Table 14 on page 61. The default priority of any maskable interrupt increases to Level 1 (a higher priority than any Level 0 interrupt) by setting the appropriate bit in the Interrupt Priority registers as listed in Table 14 on page 61.

Table 14. Interrupt Priority Registers (INT_P0 = 0010h, INT_P1 = 0011h, INT_P2 = 0012h, INT_P3 = 0013h, INT_P4 = 0014h, INT_P5 = 0015h)

Bit	7	6	5	4	3	2	1	0
INT_P0 Reset	0	0	0	0	0	0	0	0
INT_P1 Reset	0	0	0	0	0	0*	0*	0
INT_P2 Reset	0	0	0	0	0	0	0	0
INT_P3 Reset	0	0	0	0	0	0	0	0
INT_P4 Reset	0	0	0	0	0	0	0	0
INT_P5 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: X = Undefined; R/W = Read/Write, *Unused.

Bit Position	Value	Description
7 INT_PX	0	Default Interrupt Priority
	1	Level One Interrupt Priority
6 INT_PX	0	Default Interrupt Priority
	1	Level One Interrupt Priority
5 INT_PX	0	Default Interrupt Priority
	1	Level One Interrupt Priority
4 INT_PX	0	Default Interrupt Priority
	1	Level One Interrupt Priority
3 INT_PX	0	Default Interrupt Priority
	1	Level One Interrupt Priority
2 INT_PX	0	Default Interrupt Priority
	1	Level One Interrupt Priority
1 INT_PX	0	Default Interrupt Priority
	1	Level One Interrupt Priority
0 INT_PX	0	Default Interrupt Priority
	1	Level One Interrupt Priority

The Interrupt Vector Priority Control bits are listed in Table 15.

Table 15. Interrupt Vector Priority Control Bits

Priority Control Bit	Vector	Source	Priority Control Bit	Vector	Source
INT_P0[0]	040h	EMAC Rx	INT_P3[0]	0A0h	Port B 0
INT_P0[1]	044h	EMAC Tx	INT_P3[1]	0A4h	Port B 1
INT_P0[2]	048h	EMAC SYS	INT_P3[2]	0A8h	Port B 2
INT_P0[3]	04Ch	PLL	INT_P3[3]	0ACh	Port B 3
INT_P0[4]	050h	Flash	INT_P3[4]	0B0h	Port B 4
INT_P0[5]	054h	Timer 0	INT_P3[5]	0B4h	Port B 5
INT_P0[6]	058h	Timer 1	INT_P3[6]	0B8h	Port B 6
INT_P0[7]	05Ch	Timer 2	INT_P3[7]	0BCh	Port B 7
INT_P1[0]	060h	Timer 3	INT_P4[0]	0C0h	Port C 0
INT_P1[1]	064h	Unused*	INT_P4[1]	0C4h	Port C 1
INT_P1[2]	068h	Unused*	INT_P4[2]	0C8h	Port C 2
INT_P1[3]	06Ch	RTC	INT_P4[3]	0CCh	Port C 3
INT_P1[4]	070h	UART 0	INT_P4[4]	0D0h	Port C 4
INT_P1[5]	074h	UART 1	INT_P4[5]	0D4h	Port C 5
INT_P1[6]	078h	I^2C	INT_P4[6]	0D8h	Port C 6
INT_P1[7]	07Ch	SPI	INT_P4[7]	0DCh	Port C 7
INT_P2[0]	080h	Port A 0	INT_P5[0]	0E0h	Port D 0
INT_P2[1]	084h	Port A 1	INT_P5[1]	0E4h	Port D 1
INT_P2[2]	088h	Port A 2	INT_P5[2]	0E8h	Port D 2
INT_P2[3]	08Ch	Port A 3	INT_P5[3]	0ECh	Port D 3
INT_P2[4]	090h	Port A 4	INT_P5[4]	0F0h	Port D 4
INT_P2[5]	094h	Port A 5	INT_P5[5]	0F4h	Port D 5
INT_P2[6]	098h	Port A 6	INT_P5[6]	0F8h	Port D 6
INT_P2[7]	09Ch	Port A 7	INT_P5[7]	0FCh	Port D 7

Note: *The vector addresses 064h and 068h are left unused to avoid conflict with the NMI vector address 066h.

If more than one maskable interrupt is prioritized to a higher level (Level 1), the higher-priority interrupts follow the priority order as listed in Table 14 on page 61. For example,

Table 16 lists the maskable interrupts 044h (EMAC Tx), 084h (Port A 1), and 06Ch (RTC) as elevated to priority Level 1. Table 17 lists the new interrupt priority for the top ten maskable interrupts.

Table 16. Example: Maskable Interrupt Priority

Priority Register	Setting	Description
INT_P0	02h	Increase 044h (EMAC Tx) to Priority Level 1
INT_P1	08h	Increase 06Ch (RTC) to Priority Level 1
INT_P2	02h	Increase 084h (Port A1) to Priority Level 1
INT_P3	00h	Default priority
INT_P4	00h	Default priority
INT_P5	00h	Default priority

Table 17. Example: Priority Levels for Maskable Interrupts

Priority	Vector	Source
0	044h	EMAC Tx
1	06Ch	RTC
2	084h	Port A 1
3	040h	EMAC Rx
4	048h	EMAC SYS
5	04Ch	PLL
6	050h	Flash
7	054h	Timer 0
8	058h	Timer 1
9	05Ch	Timer 2

GPIO Port Interrupts

All interrupts are latched. In effect, an interrupt is held even if the interrupt occurs while another interrupt is being serviced and interrupts are disabled, or if the interrupt is of a lower priority. However, before the latched ISR completes its task or re-enables interrupts, the ISR must clear the interrupt. For on-chip peripherals, the interrupt is cleared when the data register is accessed. For GPIO-level interrupts, the interrupt signal must be removed before the ISR completes its task. For GPIO-edge interrupts (single and dual), the interrupt is cleared by writing a 1 to the corresponding bit position in the Px_ALT0 register. See Edge Triggered Interrupts on page 54.

Note: *For F91 devices with a ZDI or JTAG revision less than 2, care must be taken using a GPIO data register when it is configured for interrupts. For edge-interrupt modes (modes 6 and 9) as discussed earlier, writing 1 clears the interrupt. However, 1 in the data register also conveys a particular configuration. For example, when the data register Px_DR is set first followed by the Px_ALT2, Px_ALT1, and Px_DDR registers, then the configuration is performed correctly. Writing 1 to the register later to clear interrupts does not change the configuration. For F91 devices with a ZDI or JTAG revision 2 or later, the clearing of interrupts is accomplished through the new Px_ALT0 registers and the above problem does not exist.*

In mode 9 operation, if the GPIO is already configured for mode 9 and if the trigger edge must be changed (from falling to rising or from rising to falling), then the configuration must be changed to another mode, such as Mode 2, and then changed back to mode 9. For example, enter mode 2 by writing the registers in the sequence PxDR, Px_ALT2, Px_ALT1, and Px_DDR. Next, change back to mode 9 by writing the registers in the sequence PxDR, Px_ALT2, Px_ALT1, and Px_DDR.

In Mode 8 operation, if the GPIO is configured for level-sensitive interrupts, a Write value to Px_DR after configuration must be the same Write value used when configuring the GPIO.

Chip Selects and Wait States

The eZ80F91 generates four chip selects for external devices. Each chip select is programmed to access either the memory space or the I/O space. The memory chip selects are individually programmed on a 64 KB boundary. Each I/O chip selects choose a 256-byte section of I/O space. In addition, each chip select is programmed for up to 7 Wait states.

Memory and I/O Chip Selects

Each of the chip selects are enabled either for 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 select is controlled by three control registers. To enable a particular memory chip select, the following conditions must be satisfied:

- 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].
- On-chip Flash is not configured for the same address space, because on-chip Flash is prioritized higher than all memory chip selects.
- On-chip RAM is not configured for the same address space, because on-chip RAM is prioritized higher than Flash and all memory chip selects.
- No higher priority (lower number) chip select meets the above conditions.
- A memory access instruction must be executing.

If all the preceding conditions are satisfied to generate a memory chip select, then the following results occur:

- The appropriate chip select—$\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$, or $\overline{CS3}$ is asserted (driven Low).
- $\overline{MREQ}$ is asserted (driven Low).
- Depending on the instruction either $\overline{RD}$ or $\overline{WR}$ 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, then chip select 0 is active. If the address range programmed for any chip select signal overlaps with the address of internal memory, the internal memory is accorded higher priority. If the particular chip select(s) are configured with an address range that overlaps with an internal memory address and when the internal memory is accessed, the chip select signal is not asserted.

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 the other chip select Lower and Upper Bound registers reset to `00h`.

Memory Chip Select Example

The use of Memory chip selects is displayed in Figure 7 on page 67. The associated control register values are listed in Table 18 on page 67. In this example, all four 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.

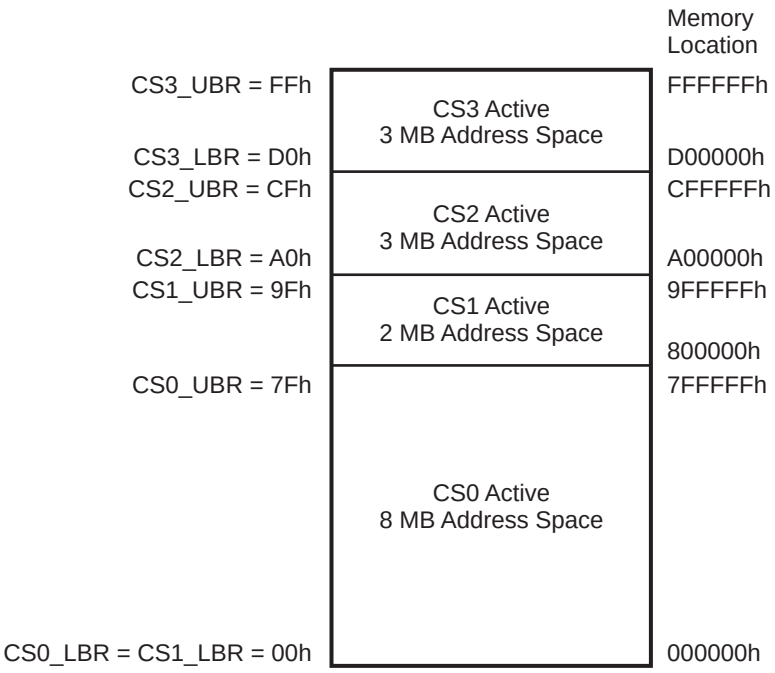


Figure 7. Example: Memory Chip Select

Table 18. Example: Register Values for Figure 7 Memory Chip Select

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.

Input/Output Chip Select Operation

I/O chip selects will be active only when the CPU is performing I/O instructions. Because the I/O space is separate from the memory space in the eZ80F91 device, a conflict between I/O and memory addresses never occurs.

The eZ80F91 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 are always accessed from 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 eZ80F91 device. To generate a particular I/O chip select, the following conditions must be satisfied:

- 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 `0000h–00FFh`. On-chip peripheral registers assume priority for all addresses where:

  `0000h` ≤ 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 results occur:

- The appropriate chip select—$\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$, or $\overline{CS3}$ is asserted (driven Low).
- $\overline{IORQ}$ is asserted (driven Low).
- Depending on the instruction, either $\overline{RD}$ or $\overline{WR}$ is asserted (driven Low).

Wait States

For each of the chip selects, programmable Wait states are 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 are 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.

$\overline{\text{WAIT}}$ Input Signal

Similar to the programmable wait states, an external peripheral drives the $\overline{\text{WAIT}}$ input pin to force the CPU to provide additional clock cycles to complete its Read or Write operation. Driving the $\overline{\text{WAIT}}$ pin Low stalls the CPU. The CPU resumes operation on the first rising edge of the internal system clock following deassertion of the $\overline{\text{WAIT}}$ pin.

Caution: *If the $\overline{\text{WAIT}}$ 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 $\overline{\text{WAIT}}$ input pin (see Figure 8), one programmable wait state is required to allow the external peripheral sufficient time to assert the $\overline{\text{WAIT}}$ pin. It is recommended that the corresponding chip select for the external device be programmed to provide the maximum number of wait states (seven).*

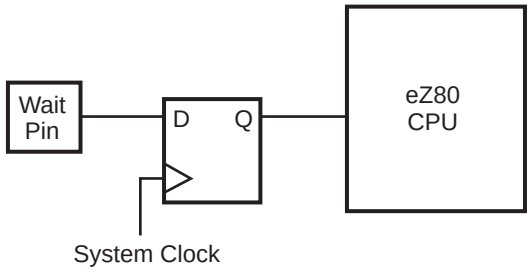


Figure 8. Wait Input Sampling Block Diagram

An example of wait state operation is illustrated in Figure 9 on page 70. In this example, the chip select is configured to provide a single wait state. The external peripheral accessed drives the $\overline{\text{WAIT}}$ pin Low to request assertion of an additional wait state. If the $\overline{\text{WAIT}}$ pin is asserted for additional system clock cycles, wait states are added until the $\overline{\text{WAIT}}$ pin is deasserted (active High).

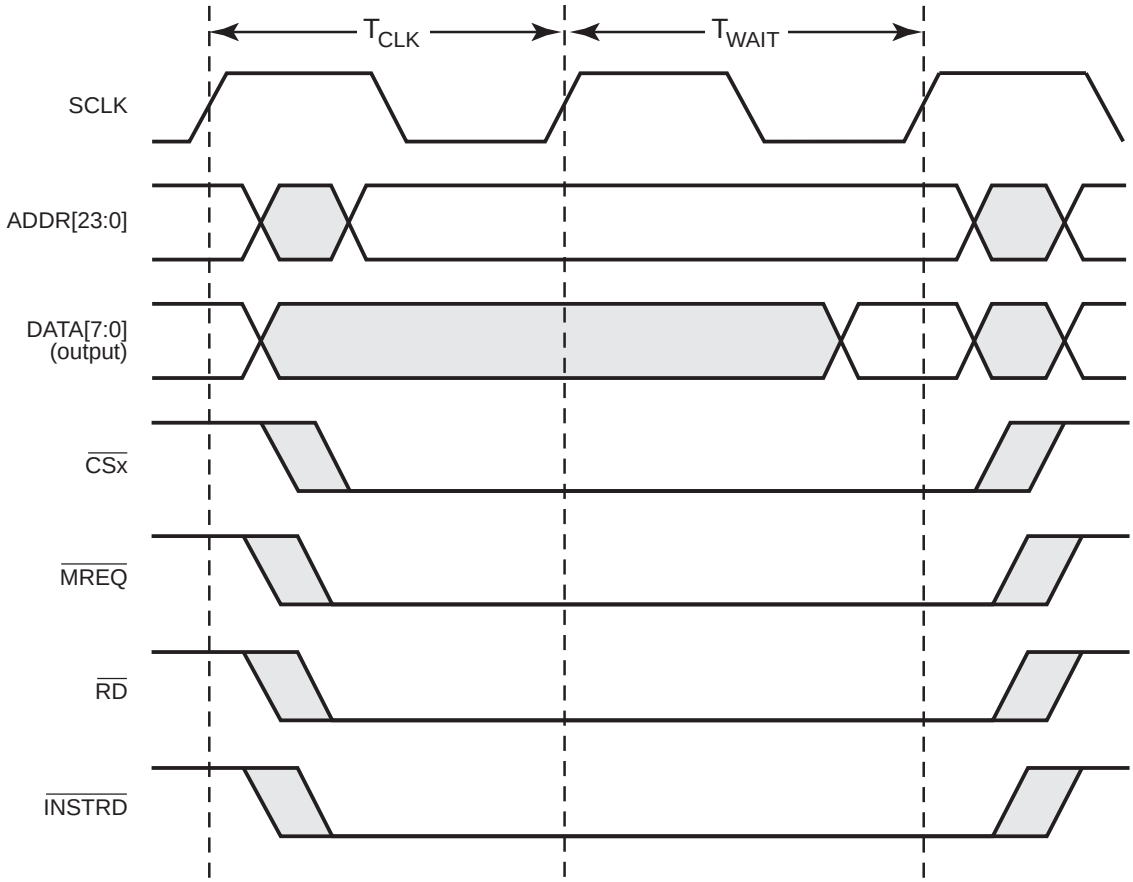


Figure 9. Example: Wait State Read Operation

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 ($\overline{\text{BUSREQ}}$), it drives the bus acknowledge pin ($\overline{\text{BUSACK}}$) Low. The external peripheral then drives 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 eZ80F91.

Bus Mode Controller

The bus mode controller allows the address and data bus timing and signal formats of the eZ80F91 to be configured to connect with external devices compatible with eZ80®, Z80®, Intel™ and Motorola microcontrollers. Bus modes for each of the chip selects are configured independently using the Chip Select Bus Mode Control Registers. The number of

CPU system clock cycles per bus mode state is also independently programmable. For Intel bus mode, multiplexed address and data are selected in which both the lower byte of the address and the data byte use the data bus, DATA[7:0]. Each of the bus modes are explained 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 on page 344. 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 listed in Table 19.

Table 19. 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 on the instruction, either the $\overline{MREQ}$ or $\overline{IORQ}$ signal is asserted. If the external $\overline{WAIT}$ pin is driven Low at least one CPU system clock cycle prior to the end of State T2, additional Wait states (T_{WAIT}) are asserted until the $\overline{WAIT}$ pin is driven High.
STATE T3	During State T3, no bus signals are altered. The data is latched by the eZ80F91 at the rising edge of the CPU system clock at the end of State T3.

During Write operations, Z80 Bus mode employs three states—T1, T2, and T3 as listed in Table 20.

Table 20. Z80 Bus Mode Write States

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

Z80® bus mode Read and Write timing is displayed in Figure 10 and Figure 11 on page 73. The Z80 bus mode states are configured for 1 to 15 CPU system clock cycles. In the figures, each Z80 bus mode state is two CPU system clock cycles in duration. The figures also display the assertion of 1 wait state (T_{WAIT}) by the external peripheral during each Z80 bus mode cycle.

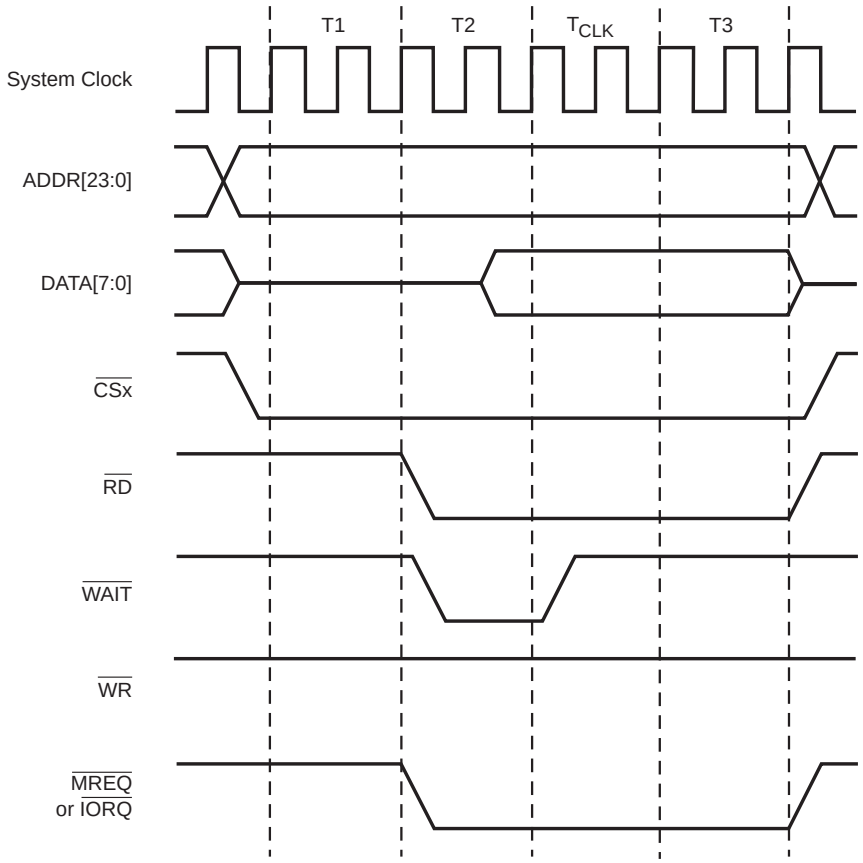


Figure 10. Example: Z80 Bus Mode Read Timing

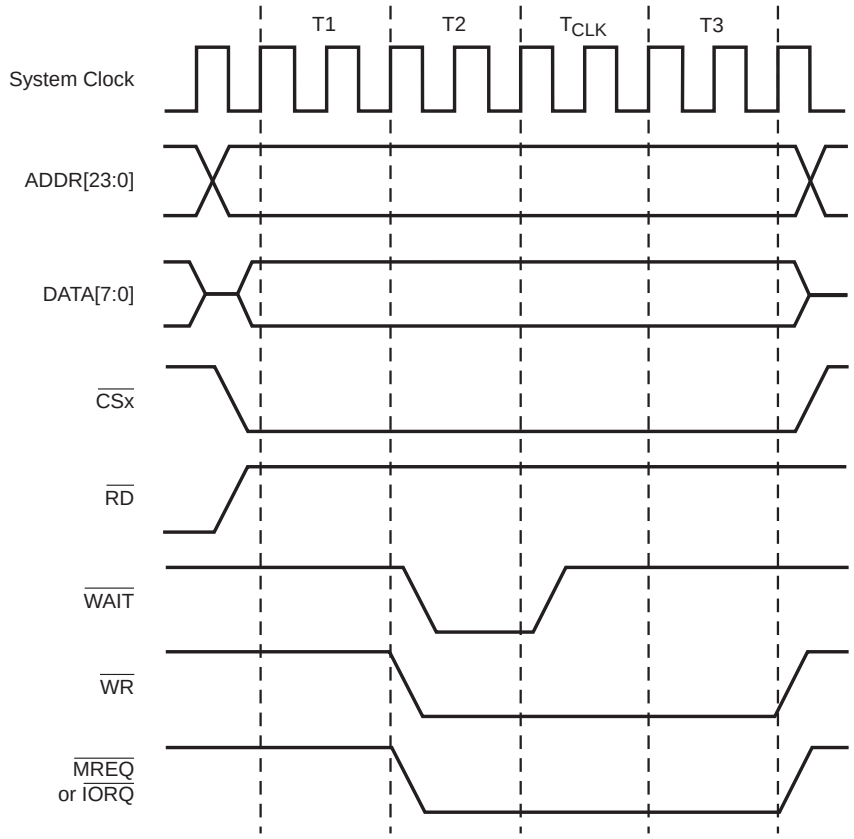


Figure 11. Example: Z80® Bus Mode Write Timing

Intel Bus Mode

Chip selects configured for Intel bus mode modify the CPU bus signals to duplicate a four-state memory transfer similar to that found on Intel-style microcontrollers. The bus signals and eZ80F91 pins are mapped as displayed in Figure 12 on page 74. In Intel bus mode, you 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.

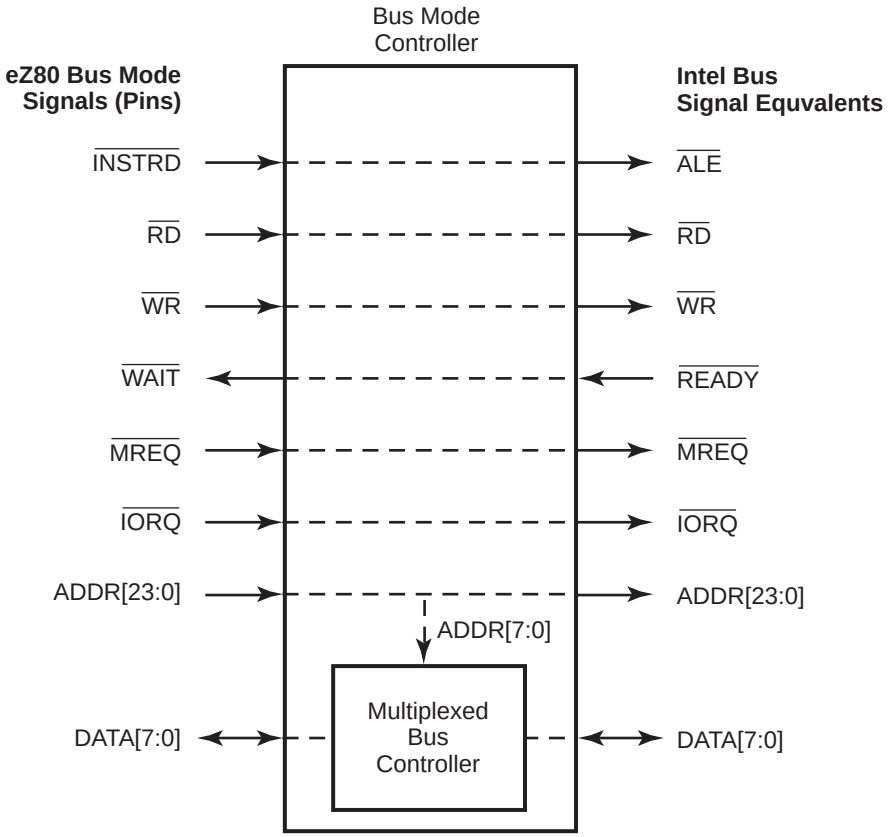


Figure 12. Intel Bus Mode Signal and Pin Mapping

Intel™ Bus Mode—Separate Address and Data Buses

During Read operations with separate address and data buses, the Intel bus mode employs four states—T1, T2, T3, and T4 as listed in Table 21.

Table 21. 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. In 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 $\overline{MREQ}$ or $\overline{IORQ}$ signal is asserted.

Table 21. 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 ($\overline{WAIT}$) pin is driven Low at least one CPU system clock cycle prior to the beginning of State T3, additional wait states (T_{WAIT}) are asserted until the READY pin is driven High.
STATE T4	The CPU latches the Read data at the beginning of State T4. The CPU deasserts the $\overline{RD}$ signal and completes the Intel bus mode cycle.

During Write operations with separate address and data buses, the Intel bus mode employs four states—T1, T2, T3, and T4 as listed in Table 22.

Table 22. 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 $\overline{WR}$ signal. Depending on the instruction, either the $\overline{MREQ}$ or $\overline{IORQ}$ signal is asserted.
STATE T3	During State T3, no bus signals are altered. If the external READY ($\overline{WAIT}$) pin is driven Low at least one CPU 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 $\overline{WR}$ signal at the beginning of State T4. The CPU holds the data and address buses till the end of T4. The bus cycle is completed at the end of T4.

Intel bus mode timing is displayed for a Read operation in Figure 13 on page 76 and for a Write operation in Figure 14 on page 77. If the READY signal (external $\overline{WAIT}$ 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 are configured for 2 to 15 CPU system clock cycles. In the Figure 13 on page 76 and Figure 14 on page 77, each Intel bus mode state is 2 CPU system clock cycles in duration. Figure 13 on page 76 and Figure 14 on page 77 also display the assertion of one Wait state (T_{WAIT}) by the selected peripheral.

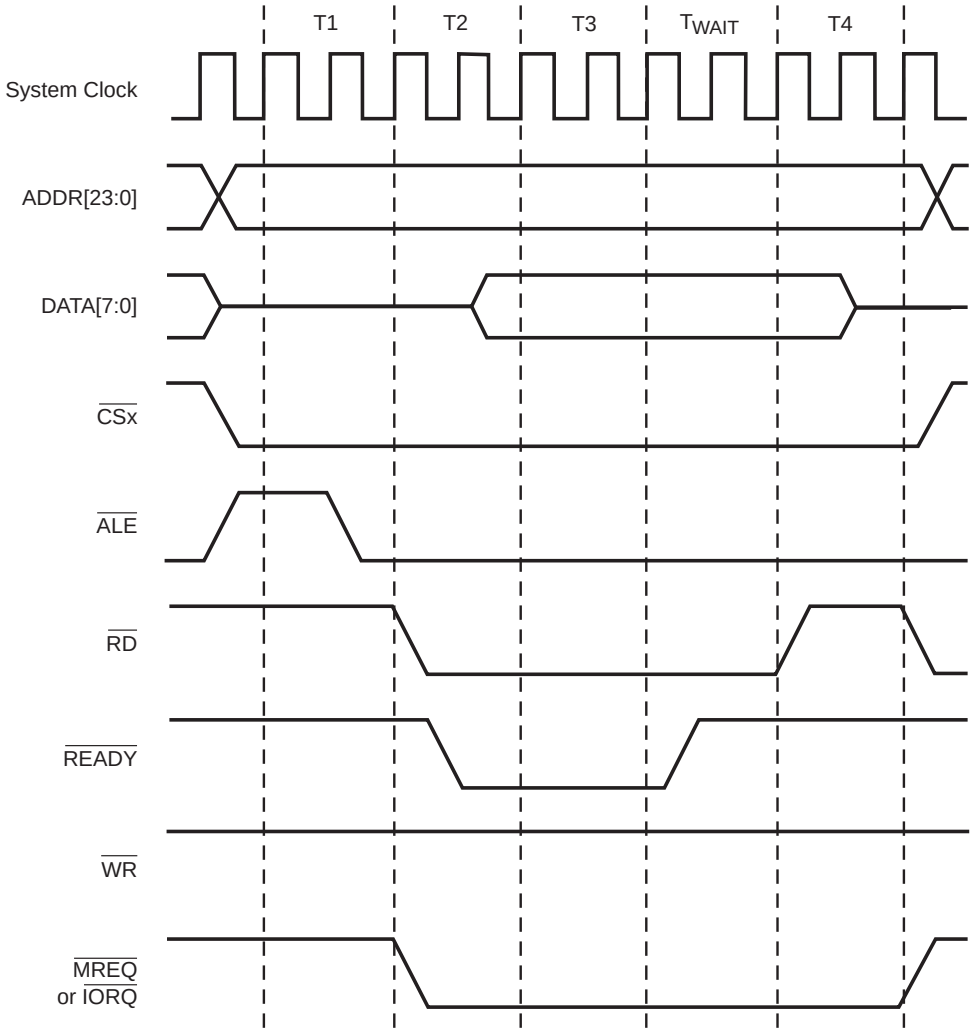


Figure 13. Example: Intel Bus Mode Read Timing—Separate Address and Data Buses

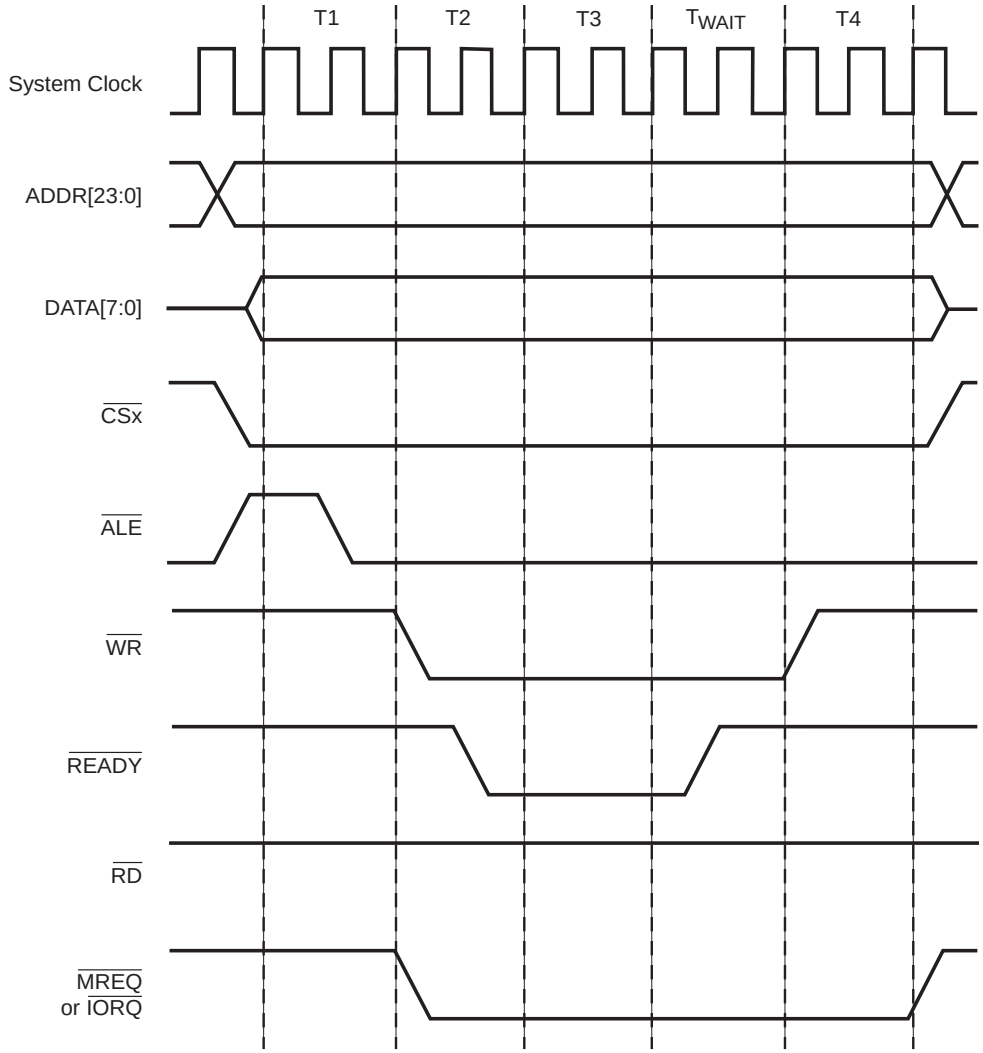


Figure 14. Example: Intel Bus Mode Write Timing—Separate Address and Data Buses

Intel™ Bus Mode—Multiplexed Address and Data Bus

During Read operations with multiplexed address and data, the Intel™ bus mode employs four states—T1, T2, T3, and T4 as listed in Table 23.

Table 23. 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. In 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 $\overline{RD}$ signal. Depending upon the instruction, either the $\overline{MREQ}$ or $\overline{IORQ}$ signal is asserted.
STATE T3	During State T3, no bus signals are altered. If the external READY ($\overline{WAIT}$) pin is driven Low at least one CPU system clock cycle prior to the beginning of State T3, additional wait states (T_{WAIT}) are asserted until the READY pin is driven High.
STATE T4	The CPU latches the Read data at the beginning of State T4. The CPU deasserts the $\overline{RD}$ signal and completes the Intel™ bus mode cycle.

During Write operations with multiplexed address and data, the Intel™ bus mode employs four states—T1, T2, T3, and T4 as listed in Table 24.

Table 24. 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 $\overline{WR}$ signal is asserted to indicate a Write operation.
STATE T3	During State T3, no bus signals are altered. If the external READY ($\overline{WAIT}$) pin is driven Low at least one CPU 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 displayed for a Read operation in Figure 15 on page 79 and for a Write operation in Figure 16 on page 80. In Figure 15 on page 79 and Figure 16 on page 80, each Intel bus mode state is 2 CPU system clock cycles in duration. Figure 15 on page 79 and Figure 16 on page 80 also display the assertion of one wait state (T_{WAIT}) by the selected peripheral.

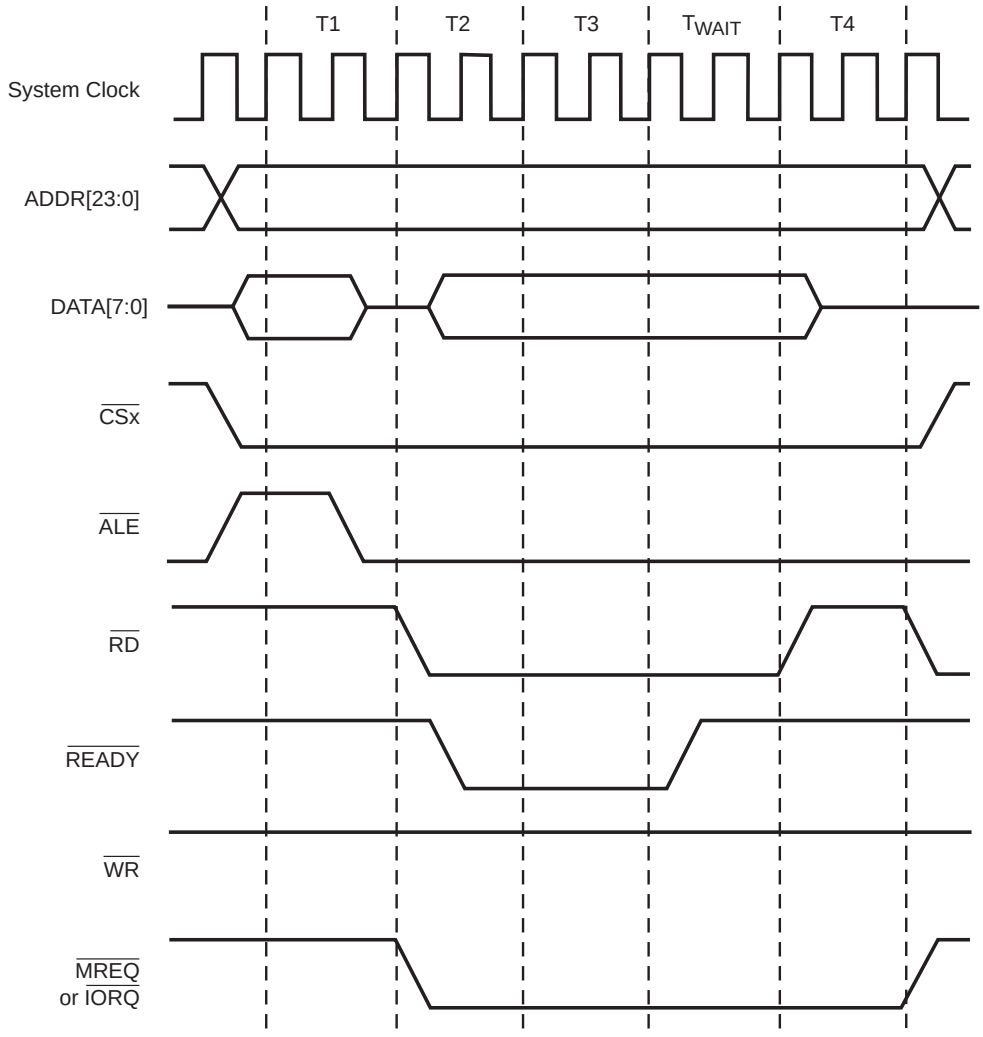


Figure 15. Example: Intel Bus Mode Read Timing—Multiplexed Address and Data Bus

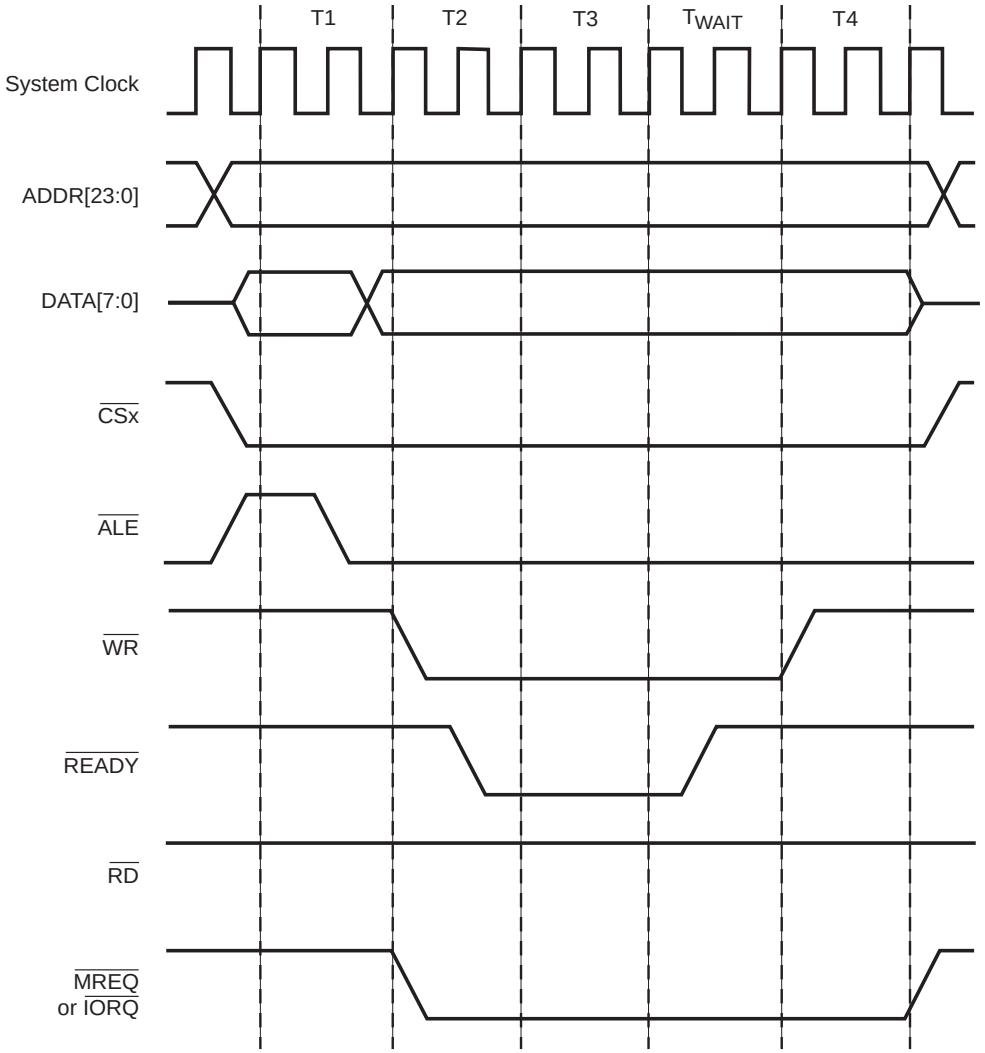


Figure 16. Example: Intel Bus Mode Write Timing—Multiplexed Address and Data Bus

Motorola Bus Mode

Chip selects configured for Motorola bus mode modify the CPU bus signals to duplicate an eight-state memory transfer similar to that on the Motorola-style microcontrollers. The bus signals (and eZ80F91 I/O pins) are mapped as displayed in Figure 17 on page 81.

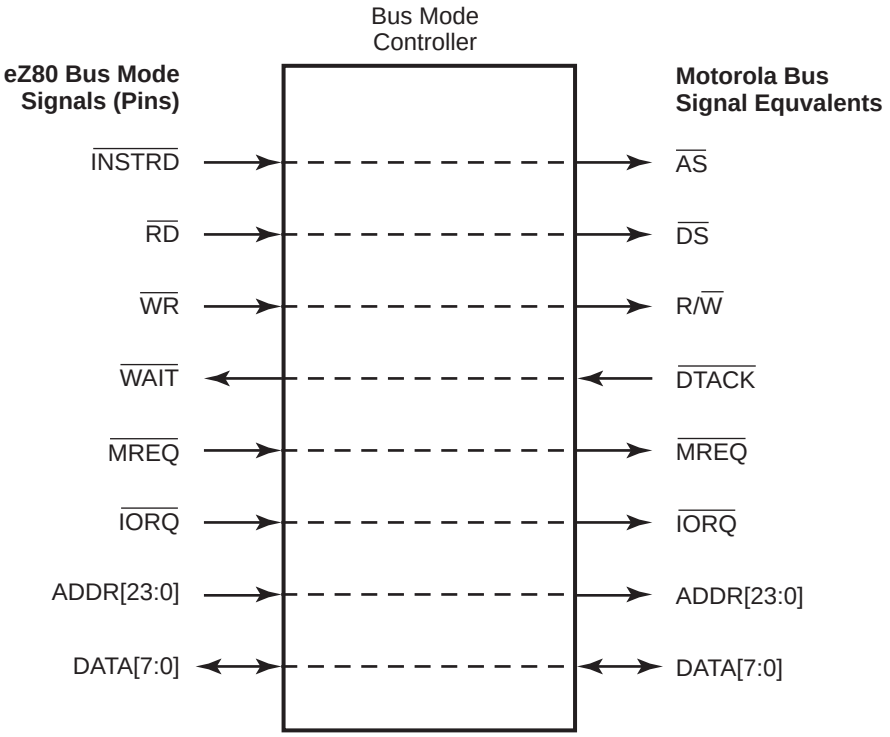


Figure 17. Motorola Bus Mode Signal and Pin Mapping

During Write operations, the Motorola bus mode employs eight states—S0, S1, S2, S3, S4, S5, S6, and S7 as listed in Table 25.

Table 25. 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 $\overline{DTACK}$ is asserted. Each wait state is a full bus mode cycle.
STATE S5	During state S5, no bus signals are altered.

Table 25. 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 $\overline{AS}$ and $\overline{DS}$. The peripheral device deasserts $\overline{DTACK}$ at this time.

The eight states for a Write operation in Motorola bus mode are listed in Table 26.

Table 26. 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/$\overline{W}$ 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 $\overline{DTACK}$ ($\overline{WAIT}$). 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 $\overline{DTACK}$ 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	On entering S7, the CPU deasserts $\overline{AS}$ and $\overline{DS}$. As the clock rises at the end of S7, the CPU drives R/$\overline{W}$ High. The peripheral device deasserts $\overline{DTACK}$ at this time.

Signal timing for Motorola bus mode is displayed for a Read operation in Figure 18 on page 83 and for a Write operation in Figure 19 on page 84. In these two figures, each Motorola bus mode state is 2 CPU system clock cycles in duration.

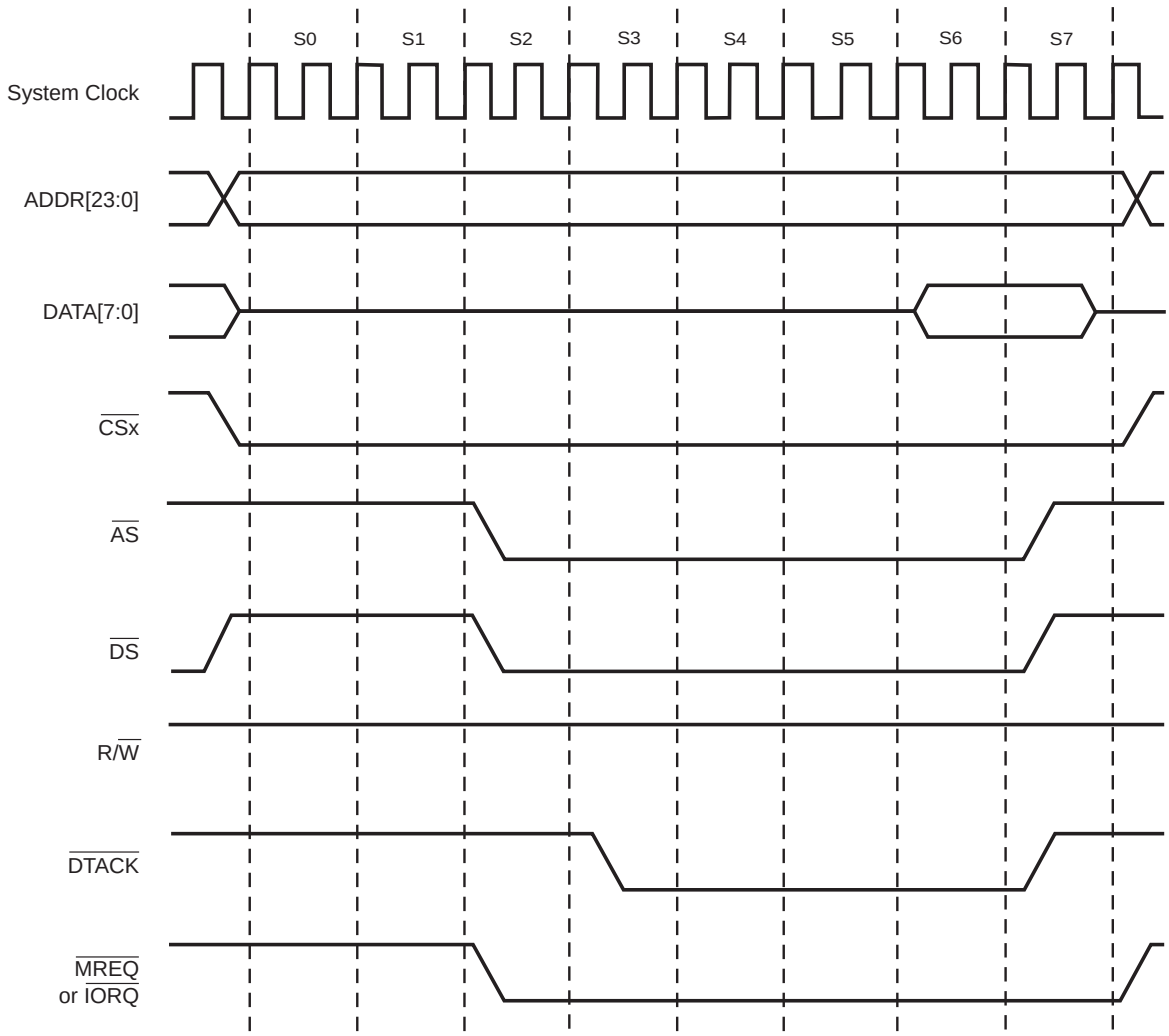


Figure 18. Example: Motorola Bus Mode Read Timing

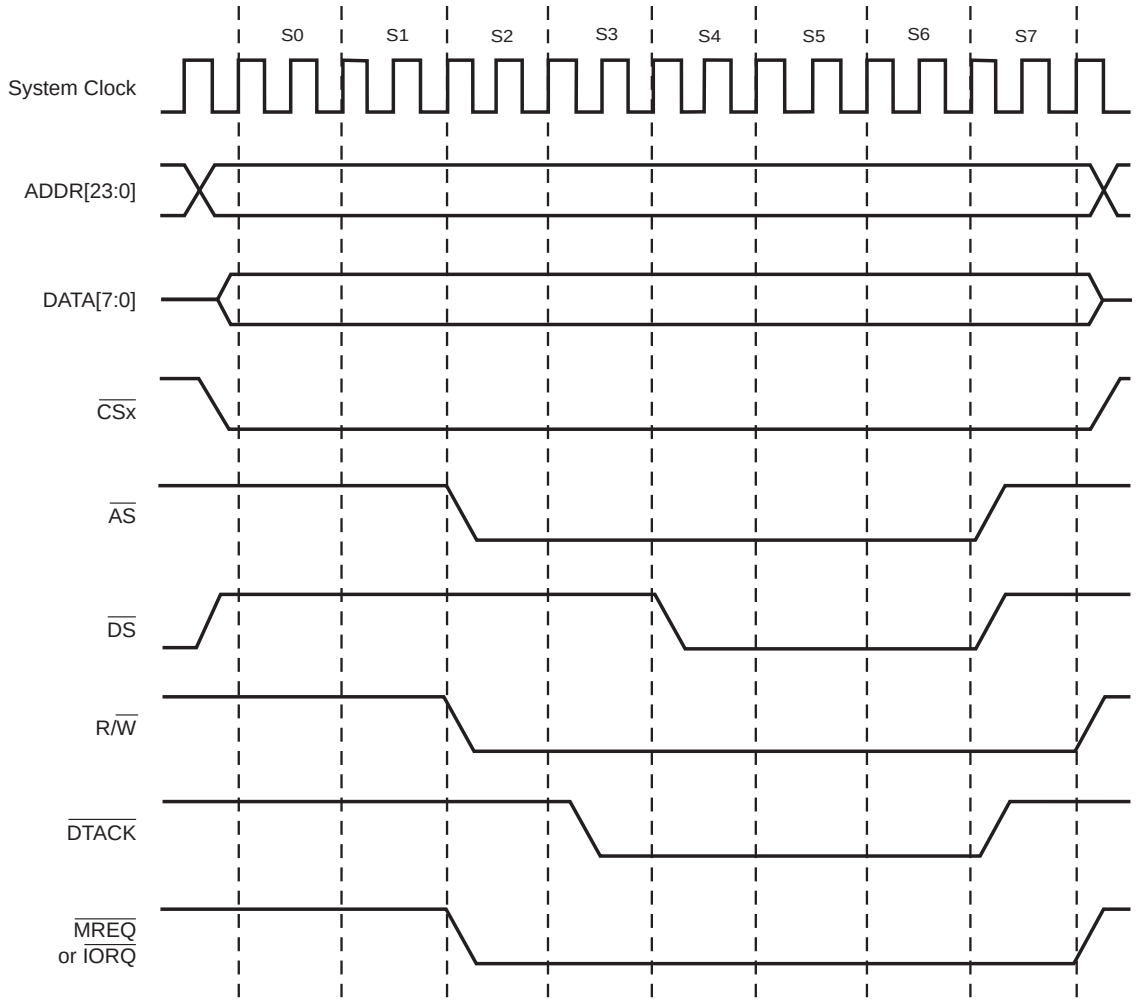


Figure 19. Example: Motorola Bus Mode Write Timing

Switching Between Bus Modes

When switching bus modes between Intel™ to Motorola, Motorola to Intel™, eZ80® to Motorola, or eZ80 to Intel™, there is one extra SCLK cycle added to the bus access. An extra clock cycle is not required for repeated access in any of the bus modes (for example, Intel™ to Intel™). An extra clock cycle is not required for Intel™ (or Motorola) to eZ80 bus mode (under normal operation). The extra clock cycle is 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 Register

For Memory chip selects, the chip select x Lower Bound register (see Table 27) defines the lower bound of the address range for which the corresponding Memory chip select (if enabled) is active. For I/O chip selects, the chip select x Lower Bound 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 27. Chip Select x Lower Bound Register (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 must 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 must be generated.

Chip Select *x* Upper Bound Register

For Memory chip selects, the Chip Select *x* Upper Bound registers, listed in Table 28, defines the upper bound of the address range for which the corresponding Chip Select (if enabled) are active. For I/O chip selects, this register produces no effect. The reset state for the Chip Select 0 Upper Bound register is FFh when the reset state for the other Chip Select Upper Bound registers is 00h.

Table 28. Chip Select x Upper Bound Register (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 must be generated.
		For I/O Chip Selects (CSx_IO = 1) No effect.

Chip Select x Control Register

The Chip Select x Control register (see Table 29) 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 when the reset state for three other Chip Select Control registers is 00h.

Table 29. Chip Select x Control Register (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.

Chip Select x Bus Mode Control Register

The Chip Select Bus Mode register (see Table 30) configures the chip select for eZ80®, Z80®, Intel™, or Motorola bus modes. Changing the bus mode allows the eZ80F91 device 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 30. Chip Select *x* Bus Mode Control Register (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 $\overline{\text{WAIT}}$ input pin in Z80 mode requires that BUS_CYCLE is set to a value greater than 1.
3. BUS_CYCLE produces no effect in eZ80 mode.

Bus Arbiter

The Bus Arbiter within the eZ80F91 allows external bus masters to gain control of the CPU memory interface bus. During normal operation, the eZ80F91 device is the bus master. External devices request master use of the bus by asserting the $\overline{\text{BUSREQ}}$ pin. The Bus Arbiter forces the CPU to release the bus after completing the current instruction. When the CPU releases the bus, the Bus Arbiter asserts the $\overline{\text{BUSACK}}$ pin to notify the external device that it can master the bus. When an external device assumes control of the memory interface bus, the bus acknowledge cycle is complete. Table 31 on page 90 lists the status of the pins on the eZ80F91 device during bus acknowledge cycles.

During a bus acknowledge cycle, the bus interface pins of the eZ80F91 device are used by an external bus master to control the memory and I/O chip selects.

Table 31. eZ80F91 Pin Status During Bus Acknowledge Cycles

Pin Symbol	Signal Direction	Description
ADDR23..ADDR0	Input	Allows external bus master to utilize the chip select logic of the eZ80F91.
CS0	Output	Normal operation.
CS1	Output	Normal operation.
CS2	Output	Normal operation.
CS3	Output	Normal operation.
DATA7..0	Tristate	Allows external bus master to communicate with external peripherals.
IORQ	Input	Allows external bus master to utilize the chip select logic of the eZ80F91.
MREQ	Input	Allows external bus master to utilize the chip select logic of the eZ80F91.
RD	Tristate	Allows external bus master to communicate with external peripherals.
WR	Tristate	Allows external bus master to communicate with external peripherals.
INSTRD	Tristate	Allows external bus master to communicate with external peripherals.

Normal bus operation of the eZ80F91 device using $\overline{CS0}$ to communicate to an external peripheral is displayed in Figure 20 on page 91. Figure 21 on page 91 displays an external bus master communicating with an external peripheral during bus acknowledge cycles.

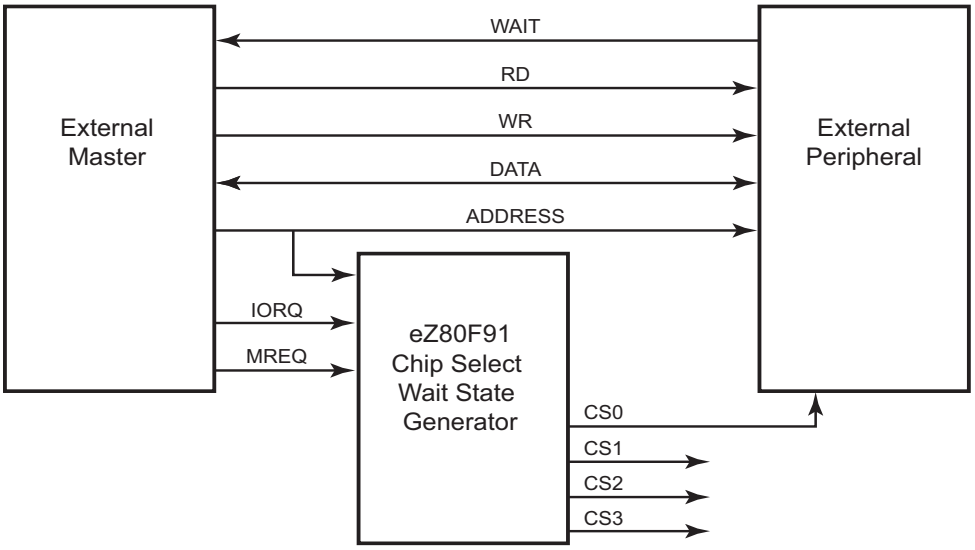


Figure 20. Memory Interface Bus Operation During CPU Bus Cycles, Normal Operation

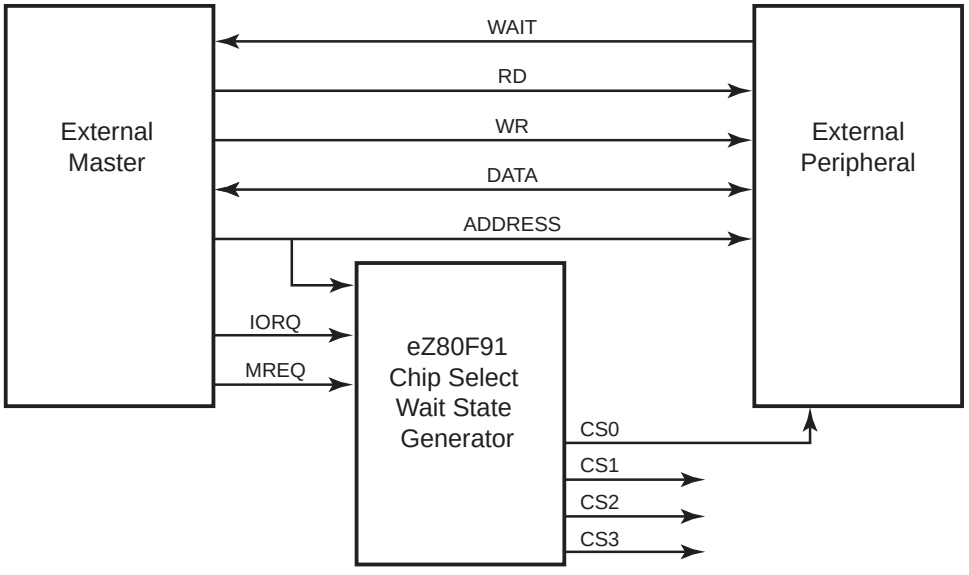


Figure 21. Memory Interface Bus Operation During Bus Acknowledge Cycles

During bus acknowledge cycles, the Memory and I/O chip select logic is controlled by the external address bus and external $\overline{\text{IORQ}}$ and $\overline{\text{MREQ}}$ signals.

The following chip select features are not available during bus acknowledge cycles:

- The chip select logic does not insert wait states during bus acknowledge cycles regardless of the WAIT configuration for the decoded chip select.
- The bus mode controller does not function during bus acknowledge cycles.
- Internal registers and memory addresses in the eZ80F91 device are not accessible during bus acknowledge cycles.

Random Access Memory

The eZ80F91 device features 8 KB (8192 bytes) of single-port data Random Access Memory (RAM) for general-purpose use and 8 KB of RAM for the EMAC. RAM is enabled or disabled, and it is relocated to the top of any 64 KB page in memory. Data is passed to and from RAM via the 8-bit data bus. On-chip RAM operates with zero wait states. EMAC RAM is accessed via the bus arbiter and executes with zero or one Wait states.

General-purpose RAM occupies memory addresses in the RAM Address Upper Byte register in the range {RAM_ADDR_U[7:0], E000h} to {RAM_ADDR_U[7:0], FFFFh}. EMAC RAM occupies memory addresses in the range {RAM_ADDR_U[7:0], C000h} to {RAM_ADDR_U[7:0], DFFFh}. Following a RESET, RAM is enabled when RAM_ADDR_U is set to FFh. Figure 22 displays a memory map for on-chip RAM. In this example, RAM_ADDR_U is set to 7Ah. Figure 22 is not drawn to scale, as RAM occupies only a very small fraction of the available 16 MB address space.

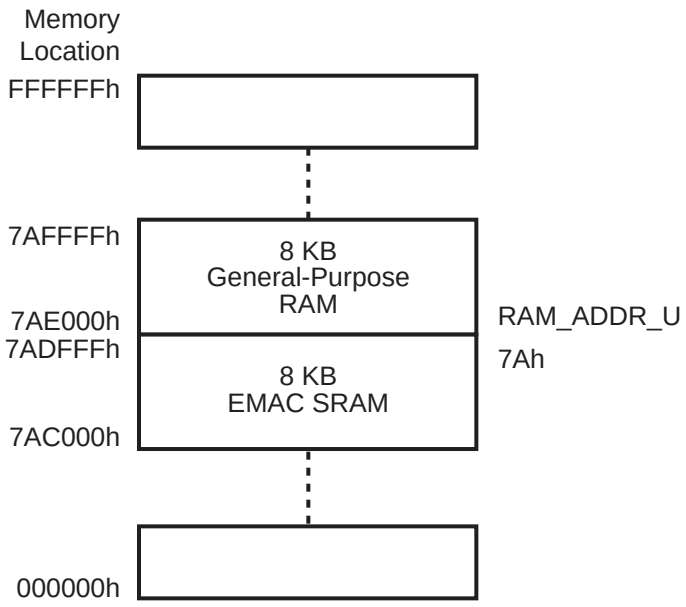


Figure 22. Example: eZ80F91 On-Chip RAM Memory Addressing

When enabled, on-chip RAM assumes priority over on-chip Flash memory and any memory chip selects that is also enabled in the same address space. If an address is generated in a range that is covered by both the RAM address space and a particular memory chip

select address space, the memory chip select is not activated. On-chip RAM is not accessible to external devices during bus acknowledge cycles.

RAM Control Registers

RAM Control Register

Internal general-purpose RAM is disabled by clearing the GPRAM_EN bit. The default on RESET is for general-purpose RAM to be enabled. See Table 32.

Table 32. RAM Control Register (RAM_CTL=00B4h)

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

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

Bit Position	Value	Description
7 GPRAM_EN	0	On-chip general-purpose RAM is disabled.
	1	On-chip general-purpose RAM is enabled.
6 ERAM_EN	0	On-chip EMAC RAM is disabled.
	1	On-chip EMAC RAM is enabled.
[5:0]	000000	Reserved.

RAM Address Upper Byte Register

The RAM_ADDR_U register defines the upper byte of the address for on-chip RAM. If enabled, RAM addresses assume priority over all Chip Selects. The external Chip Select signals are not asserted if the corresponding RAM address is enabled. See Table 33.

Table 33. RAM Address Upper Byte Register (RAM_ADDR_U=00B5h)

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.

Bit Position	Value	Description
[7:0] RAM_ADDR_U	00h–FFh	This byte defines the upper byte of the RAM address. When enabled, the general-purpose RAM address space ranges from {RAM_ADDR_U, E000h} to {RAM_ADDR_U, FFFFh}. When enabled, the EMAC RAM address space ranges from {RAM_ADDR_U, C000h} to {RAM_ADDR_U, DFFFh}.

MBIST Control

There are two Memory Built-In Self-Test (MBIST) controllers for the RAM blocks on the eZ80F91. MBIST_GPR is for General-Purpose RAM and MBIST_EMR is for EMAC RAM. Writing a 1 to MBIST_ON starts the MBIST testing. Writing a 0 to MBIST_ON stops the MBIST testing. On completion of the MBIST testing, MBIST_ON is automatically reset to 0. If RAM passes MBIST testing, MBIST_PASS is 1. The value in MBIST_PASS is only valid when MBIST_DONE is High. See Table 34.

Table 34. MBIST Control Register (MBIST_GPR=00B6h, MBIST_EMR=00B7h)

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

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

Bit Position	Value	Description
7 MBIST_ON	0	MBIST Testing of the RAM is disabled.
	1	MBIST Testing of the RAM is enabled.
6 MBIST_DONE	0	MBIST Testing has not completed.
	1	MBIST Testing has completed.
5 MBIST_PASS	0	MBIST Testing has failed.
	1	MBIST Testing has passed.
[4:0]	00000	Reserved.

Flash Memory

The eZ80F91 device features 256 KB (262,144 bytes) of non-volatile Flash memory with Read/Write/Erase capability. The main Flash memory array is arranged in 128 pages with 8 rows per page and 256 bytes per row. In addition to main Flash memory, there are two separately addressable rows which comprise a 512-byte information page.

In eight 32 KB blocks, 256 KB of main storage is protected. Protecting a 32 KB block prevents Write or Erase operations. The lower 32 KB block (00000h–07FFFh) is protected using the external $\overline{WP}$ pin. This portion of memory is called the Boot block because the CPU always starts executing code from this location at startup. If the application requires external program memory, then the Boot block must at least contain a jump instruction to move the Program Counter outside of the Flash memory space.

The Flash memory arrangement is displayed in Figure 23.

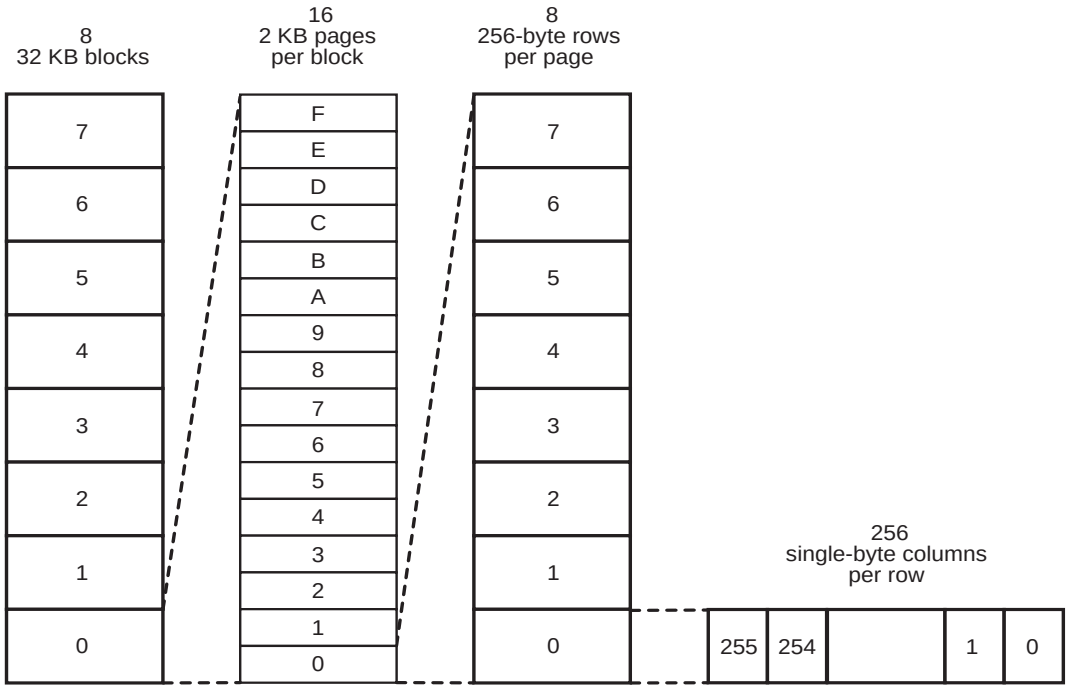


Figure 23. eZ80F91 Flash Memory Arrangement

Flash Memory Overview

The eZ80F91 device includes a Flash memory controller that automatically converts standard CPU Read and Write cycles to the specific protocol required for the Flash memory array. As such, standard memory Read and Write instructions access the Flash memory array as if it is internal RAM. The controller also supports I/O access to the Flash memory array, in effect presenting it as an indirectly addressable bank of I/O registers. These access methods are also supported via the ZDI and OCI™ interfaces.

In addition, eZ80Acclaim!® Flash Microcontrollers support a Flash Read–While–Write methodology. In other words, the eZ80® CPU continues to read and execute code from an area of Flash memory when a nonconflicting area of Flash memory is being programmed.

The Flash memory controller contains a frequency divider, a Flash register interface, and a Flash control state machine. A simplified block diagram of the Flash controller is displayed in Figure 24.

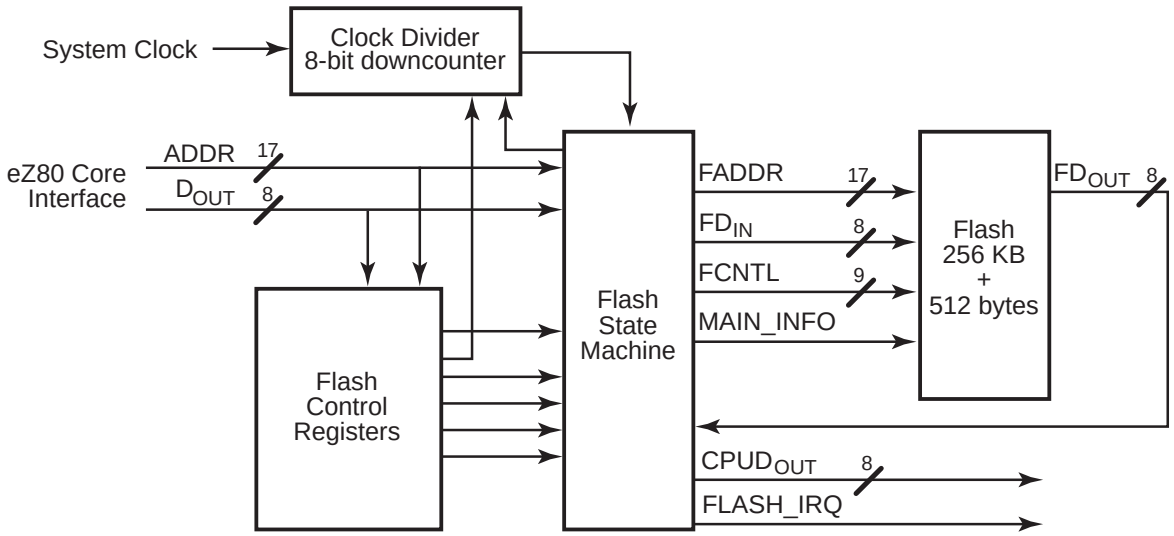


Figure 24. Flash Memory Block Diagram

Reading Flash Memory

The main Flash memory array is read using both memory and I/O operations. As an auxiliary storage area, the information page is only accessible via I/O operations. In all cases, Wait states are automatically inserted to allow for read access time.

Memory Read

A memory Read operation uses the address bus and data bus of the eZ80F91 device to read a single data byte from Flash memory. This Read operation is similar to reads from RAM. To perform Flash memory reads, the FLASH_CTRL register must be configured to enable memory access to Flash with the appropriate number of wait states. See Table 38 on page 105.

Only the main area of Flash memory is accessible via memory reads. The information page must be read using I/O access.

I/O Read

A single-byte I/O Read operation uses I/O registers for setting the column, page, and row address to be read. A Read of the FLASH_DATA register returns the contents of Flash memory at the designated address. Each access to the FLASH_DATA register causes an autoincrement of the Flash address stored in the Flash address registers (FLASH_PAGE, FLASH_ROW, FLASH_COL). To allow for Flash memory access time, the FLASH_CTRL register must be configured with the appropriate number of wait states. See Table 38 on page 105.

Programming Flash Memory

Flash memory is programmed using standard I/O or memory Write operations that the Flash memory controller automatically translates to the detailed timing and protocol required for Flash memory. The more efficient multibyte (row) programming mode is only available via I/O Writes.

Note: *To ensure data integrity and device reliability, two main restrictions exist on programming of Flash memory:*

1. *The cumulative programming time since the last erase cannot exceed 31 ms for any given row.*
2. *The same byte cannot be programmed more than once since the last erase.*

Single-Byte I/O Write

A single-byte I/O Write operation uses I/O registers for setting the column, page, and row address to be written. The FLASH_DATA register stores the data to be written. While the CPU executes an I/O instruction to load the data into the FLASH_DATA register, the Flash controller asserts the internal WAIT signal to stall the CPU until the Flash Write operation is complete. A single-byte Write takes between 66 µs and 85 µs to complete. Programming an entire row (256 bytes) using single-byte Writes therefore takes no more than 21.8 ms. This duration of time does not include the time required by the CPU to transfer data to the registers which is a function of the instructions employed and the system clock frequency. Each access to the FLASH_DATA register causes an

autoincrement of the Flash address stored in the Flash Address registers (FLASH_PAGE, FLASH_ROW, FLASH_COL).

A typical sequence that performs a single-byte I/O Write is shown below. Because the Write is self-timed, step 2 of the sequence is repeated back-to-back without requiring polling or interrupts.

1. Write the FLASH_PAGE, FLASH_ROW, and FLASH_COL registers with the address of the byte to be written.
2. Write the data value to the FLASH_DATA register.

Multibyte I/O Write (Row Programming)

Multibyte I/O Write operations use the same I/O registers as single-byte Writes. Multibyte I/O Writes allow the programming of full row and are enabled by setting the ROW_PGM bit of the Flash Program Control Register. For multibyte I/O Writes, the CPU sets the address registers, enables row programming, and then executes an I/O instruction (with repeat) to load the block of data into the FLASH_DATA register. For each individual byte written to the FLASH_DATA register during the block move, the Flash controller asserts the internal WAIT signal to stall the CPU until the current byte is programmed. Each access to the FLASH_DATA register causes an autoincrement of the Flash address stored in the Flash Address registers (FLASH_PAGE, FLASH_ROW, FLASH_COL).

During row programming, the Flash controller continuously asserts the Flash memory's high voltage signal until all bytes are programmed (column address < 255). As a result, the row programs more quickly than if the high-voltage signal is toggled for each byte. The per-byte programming time during row programming is between 41µs and 52µs. As such, programming 256 bytes of a row in this mode takes not more than 13.4 ms, leaving 17.6 ms for CPU instruction overhead to fetch the 256 bytes.

A typical sequence that performs a multibyte I/O Write is shown below:

1. Check the FLASH_IRQ register to ensure that any previous row program is completed.
2. Write the FLASH_PAGE, FLASH_ROW, and FLASH_COL registers with the address of the first byte to be written.
3. Set the ROW_PGM bit in the FLASH_PGCTL register to enable row programming mode.
4. Write the next data value to the FLASH_DATA register.
5. If the end of the row has not been reached, return to step 4.

During row programming, software must monitor the row time-out error bit either by enabling this interrupt or via polling. If a row time-out occurs, the Flash controller aborts the row programming operation, and software must assure that no further Writes are performed to the row without it first being erased. It is suggested that row programming is be used one time per row and not in combination with single-byte Writes to the same row

without first erasing it. Otherwise, the burden is on software to ensure that the 31 ms maximum cumulative programming time between erases is not exceeded for a row.

Memory Write

A single-byte memory Write operation uses the address bus and data bus of the eZ80F91 device for programming a single data byte to Flash memory. While the CPU executes a Load instruction, the Flash controller asserts the internal WAIT signal to stall the CPU until the Write is complete. A single-byte Write takes between 66 µs and 85µs to complete. Programming an entire row using memory Writes therefore takes no more than 21.8 ms. This duration of time does not include time required by the CPU to transfer data to the registers, which is a function of the instructions employed and the system clock frequency.

The memory Write function does not support multibyte row programming. Because memory Writes are self-timed, they are performed back-to-back without requiring polling or interrupts.

Erasing Flash Memory

Erasing bytes in Flash memory returns them to a value of FFh. Both the MASS and PAGE ERASE operations are self-timed by the Flash controller, leaving the CPU free to execute other operations in parallel. The DONE status bit in the Flash Interrupt Control Register are polled by software or used as an interrupt source to signal completion of an Erase operation. If the CPU attempts to access Flash memory while an erase is in progress, the Flash controller forces a wait state until the Erase operation is completed.

Mass Erase

Performing a MASS ERASE operation on Flash memory erases all bits contained in the main Flash memory array. The information page remains unaffected unless the FLASH_PAGE register bit 7(INFO_EN) is set. This self-timed operation takes approximately 200 ms to complete.

Page Erase

The smallest erasable unit in Flash memory is a page. The pages to be erased, whether they are the 128 main Flash memory pages or the information page, are determined by the setting of the FLASH_PAGE register. This self-timed operation takes approximately 10 ms to complete.

Information Page Characteristics

As noted earlier, the information page is not accessible using memory access instructions and must be accessed via the FLASH_DATA I/O register. The Flash Page Select Register contains a bit which selects the information page for I/O access.

There are two ways to erase the information page. You must set the FLASH_PAGE register(0x00FC) bit7(INFO_EN) and then you execute either a MASS ERASE (which also erases the entire main Flash memory array) operation or a PAGE ERASE operation.

Flash Control Registers

The Flash Control Register interface contains all the registers used in Flash memory. The definitions in this section describe each register.

Flash Key Register

Writing the two-byte sequence B6h, 49h in immediate succession to this register unlocks the Flash Divider and Flash Write/Erase Protection registers. If these values are not written by consecutive CPU I/O Writes (I/O reads and memory Read/Writes have no effect), the Flash Divider and Flash Write/Erase Protection registers remain locked. This prevents accidental overwrites of these critical Flash control register settings. Writing a value to either the Flash Frequency Divider Register or the Flash Write/Erase Protection Register automatically relocks both of the registers. See Table 35.

Table 35. Flash Key Register (FLASH_KEY = 00F5h)

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] FLASH_KEY	B6h, 49h	Sequential Write operations of the values B6h, 49h to this register will unlock the Flash Frequency Divider and Flash Write/Erase Protection registers.

Flash Data Register

The Flash Data register stores the data values to be programmed into Flash memory via I/O Write operations. An I/O read of the Flash Data register returns data from Flash memory. The Flash memory address used for I/O access is determined by the contents of the page, row, and column registers. Each access to the FLASH_DATA register causes an autoincrement of the Flash address stored in the Flash Address registers (FLASH_PAGE, FLASH_ROW, FLASH_COL). See Table 36.

Table 36. Flash Data Register (FLASH_DATA = 00F6h)

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: R/W = Read/Write.

Bit Position	Value	Description
[7:0] FLASH_DATA	00h-FFh	Data value to be written to Flash memory during an I/O Write operation, or the data value that is read in Flash memory, indicated by the Flash Address registers (FLASH_PAGE, FLASH_ROW, FLASH_COL).

Flash Address Upper Byte Register

The FLASH_ADDR_U register defines the upper 6 bits of the Flash memory address space. Changing the value of FLASH_ADDR_U allows on-chip 256 KB Flash memory to be mapped to any location within the 16 MB linear address space of the eZ80F91 device. If on-chip Flash memory is enabled, the Flash address assumes priority over any external Chip Selects. The external Chip Select signals are not asserted if the corresponding Flash address is enabled. Internal Flash memory does not hold priority over internal SRAM. See Table 37.

Table 37. Flash Address Upper Byte Register (FLASH_ADDR_U = 00F7h)

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
[7:2] FLASH_ADDR_U	00h–FCh	These bits define the upper byte of the Flash address. When on-chip Flash is enabled, the Flash address space begins at address {FLASH_ADDR_U, 00b, 0000h}. On-chip Flash has priority over all external Chip Selects.
[1:0]	00	Reserved (enforces alignment on a 256 KB boundary).

Flash Control Register

The Flash Control register enables or disables memory access to Flash memory. I/O access to the Flash control registers and to Flash memory is still possible while Flash memory space access is disabled.

The minimum access time of internal Flash memory is 60 ns. The Flash Control Register must be configured to provide the appropriate number of wait states based on the system clock frequency of the eZ80F91 device. Because the maximum SCLK frequency is 50 MHz (20 ns), the default on RESET is for four Wait states to be inserted for Flash memory access (Flash memory access + one eZ80® Bus Cycle = 60 ns + 20 ns = 80 ns; 80 ns ÷ 20 ns = 4 Wait states). See Table 38.

Table 38. Flash Control Register (FLASH_CTRL = 00F8h)

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

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

Bit Position	Value	Description
[7:5] FLASH_WAIT	000	0 wait states are inserted when the Flash is active.
	001	1 wait state is inserted when the Flash is active.
	010	2 wait states are inserted when the Flash is active.
	011	3 wait states are inserted when the Flash is active.
	100	4 wait states are inserted when the Flash is active.
	101	5 wait states are inserted when the Flash is active.
	110	6 wait states are inserted when the Flash is active.
	111	7 wait states are inserted when the Flash is active.
[4]	0	Reserved.
[3] FLASH_EN	0	Flash memory access is disabled.
	1	Flash memory access is enabled.
[2:0]	000	Reserved.

Flash Frequency Divider Register

The 8-bit frequency divider allows the programming of Flash memory over a range of system clock frequencies. Flash is programmed with system clock frequencies ranging from 154 kHz to 50 MHz. The Flash controller requires an input clock with a period that falls within the range of 5.1–6.5 µs. The period of the Flash controller clock is set in the Flash Frequency Divider Register. Writes to this register is allowed only after it is unlocked via the FLASH_KEY register. The Flash Frequency Divider Register value required versus the system clock frequency is listed in Table 39. System clock frequencies outside of the ranges shown are not supported. Register values for the Flash Frequency Divider are listed in Table 40.

Table 39. Flash Frequency Divider Values

System Clock Frequency	Flash Frequency Divider Value
154–196 kHz	1
308–392 kHz	2
462–588 kHz	3
616kHz–50 MHz	CEILING [System Clock Frequency (MHz) x 5.1 (µs)]*

Note: *The CEILING function rounds fractional values up to the next whole number. For example, CEILING(3.01) is 4.

Table 40. Flash Frequency Divider Register (FLASH_FDIV = 00F9h)

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

Note: R/W = Read/Write, R = Read Only. *Key sequence required to enable Writes

Bit Position	Value	Description
[7:0] FLASH_FDIV	01h–FFh	Divider value for generating the required 5.1-6.5 µs Flash controller clock period.

Flash Write/Erase Protection Register

The Flash Write/Erase Protection register prevents accidental Write or Erase operations. The protection is limited to a resolution of eight 32 KB blocks. Setting a bit to 1 protects that 32 KB block of Flash memory from accidental Writes or Erases. The default upon RESET is for all Flash memory blocks to be protected.

The $\overline{WP}$ pin works in conjunction with FLASH_PROT[0] to protect the lowest block (also called the Boot block) of Flash memory. If either the $\overline{WP}$ is held asserted or FLASH_PROT[0] is set, the Boot block is protected from Write and Erase operations.

Note: *A protect bit is not available for the information page. The information page is, however, protected excluded from a* MASS ERASE *by clearing the FLASH_PAGE register (0x00FC) bit7(INFO_EN).*

Writes to this register is allowed only after it is unlocked via the FLASH_KEY register. Any attempted Writes to this register while locked will set it to FFh, thereby protecting all blocks. See Table 41.

Table 41. Flash Write/Erase Protection Register (FLASH_PROT = 00FAh)

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 if unlocked, R = Read Only if locked. *Key sequence required to unlock.

Bit Position	Value	Description
[7] BLK7_PROT	0	Disable Write/Erase Protect on block 38000h to 3FFFFh.
	1	Enable Write/Erase Protect on block 38000h to 3FFFFh.
[6] BLK6_PROT	0	Disable Write/Erase Protect on block 30000h to 37FFFh.
	1	Enable Write/Erase Protect on block 30000h to 37FFFh.
[5] BLK5_PROT	0	Disable Write/Erase Protect on block 28000h to 2FFFFh.
	1	Enable Write/Erase Protect on block 28000h to 2FFFFh.
[4] BLK4_PROT	0	Disable Write/Erase Protect on block 20000h to 27FFFh.
	1	Enable Write/Erase Protect on block 20000h to 27FFFh.
[3] BLK3_PROT	0	Disable Write/Erase Protect on block 18000h to 1FFFFh.
	1	Enable Write/Erase Protect on block 18000h to 1FFFFh.
[2] BLK2_PROT	0	Disable Write/Erase Protect on block 10000h to 17FFFh.
	1	Enable Write/Erase Protect on block 10000h to 17FFFh.

Bit Position	Value	Description
[1] BLK1_PROT	0	Disable Write/Erase Protect on block 08000h to 0FFFFh.
	1	Enable Write/Erase Protect on block 08000h to 0FFFFh.
[0] BLK0_PROT	0	Disable Write/Erase Protect on block 00000h to 07FFFh.
	1	Enable Write/Erase Protect on block 00000h to 07FFFh.

Note: The lower 32 KB block (00000h to 07FFFh—BLK0) is called the Boot block and is protected using the external WP pin.

Flash Interrupt Control Register

There are two sources of interrupts from the Flash controller. These two sources are:

- Page Erase, Mass Erase, or Row Program completed successfully.
- An error condition occurred.

Either or both of these two interrupt sources are enabled by setting the appropriate bits in the Flash Interrupt Control register.

The Flash Interrupt Control register contains four status bits to indicate the following error conditions:

Row Program Time-Out—This bit signals a time-out during Row Programming. If the current row program operation does not complete within 4864 Flash controller clocks, the Flash controller terminates the row program operation by clearing bit 2 of the Flash Program Control Register and sets the RP_TM0 error bit to 1.

Write Violation—This bit indicates an attempt to write to a protected block of Flash memory (the Write was not performed).

Page Erase Violation—This bit indicates an attempt to erase a protected block of Flash memory (the requested page was not erased).

Mass Erase Violation—This bit indicates an attempt to MASS ERASE when there are one or more protected blocks in Flash memory (the MASS ERASE was not performed).

If the error condition interrupt is enabled, any of these four error conditions result in an interrupt request being sent to the eZ80F91device's interrupt controller. Reading the Flash Interrupt Control register clears all error condition flags and the DONE flag. See Table 42 on page 109.

Table 42. Flash Interrupt Control Register (FLASH_IRQ = 00FBh)

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	R	R	R	R	R

Note: R/W = Read/Write, R = Read Only. Read resets bits [5] and [3:0].

Bit Position	Value	Description
[7] DONE_IEN	0	Flash Erase/Row Program Done Interrupt is disabled.
	1	Flash Erase/Row Program Done Interrupt is enabled.
[6] ERR_IEN	0	Error Condition Interrupt is disabled.
	1	Error Condition Interrupt is enabled.
[5] DONE	0	Erase/Row Program Done Flag is not set.
	1	Erase/Row Program Done Flag is set.
[4]	0	Reserved.
[3] WR_VIO	0	The Write Violation Error Flag is not set.
	1	The Write Violation Error Flag is set.
[2] RP_TMO	0	The Row Program Time-Out Error Flag is not set.
	1	The Row Program Time-Out Error Flag is set.
[1] PG_VIO	0	The Page Erase Violation Error Flag is not set.
	1	The Page Erase Violation Error Flag is set.
[0] MASS_VIO	0	The Mass Erase Violation Error Flag is not set.
	1	The Mass Erase Violation Error Flag is set.

Note: The lower 32 KB block (00000h to 07FFFh) is called the Boot Block and is protected using the external $\overline{WP}$ pin. Attempts to page erase BLK0 or mass erase Flash when $\overline{WP}$ is asserted result in failure and signal an erase violation.

Flash Page Select Register

The msb of this register is used to select whether I/O Flash access and PAGE ERASE operations are directed to the 512-byte information page or to the main Flash memory array, and also whether the information page is included in MASS ERASE operations. The lower 7 bits are used to select one of the main 128 pages for PAGE ERASE or I/O operations.

To perform a PAGE ERASE, the software must set the proper page value prior to setting the page erase bit in the Flash Control Register. In addition, each access to the

FLASH_DATA register causes an autoincrement of the Flash address stored in the Flash Address registers (FLASH_PAGE, FLASH_ROW, FLASH_COL). See Table 43.

Table 43. Flash Page Select Register (FLASH_PAGE = 00FCh)

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, R = Read Only.

Bit Position	Value	Description
[7] INFO_EN	0	Flash I/O access to PAGE ERASE operations are directed to main Flash memory. Info page is NOT affected by a MASS ERASE operation.
	1	Flash I/O access to PAGE ERASE operations are directed to the information page. PAGE ERASE operations only affect the information page. Info page is included during a MASS ERASE operation.
[6:0] FLASH_PAGE	00h–7Fh	Page address of Flash memory to be used during the PAGE ERASE or I/O access of main Flash memory. When INFO_EN is set to 1, this field is ignored.

Flash Row Select Register

The Flash Row Select Register is a 3-bit value used to define one of the 8 rows of Flash on a single page. This register is used for all I/O access to Flash memory. In addition, each access to the FLASH_DATA register causes an autoincrement of the Flash address stored in the Flash Address registers (FLASH_PAGE, FLASH_ROW, FLASH_COL). See Table 44.

Table 44. Flash Row Select Register (FLASH_ROW = 00FDh)

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

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

Bit Position	Value	Description
[7:3]	00h	Reserved.
[2:0] FLASH_ROW	0h–7h	Row address of Flash memory to be used during an I/O access of Flash memory. When INFO_EN is 1 in the Flash Page Select Register, values for this field are restricted to 0h–1h, which selects between the two rows in the information page.

Flash Column Select Register

The Flash Column Select Register is an 8-bit value used to define one of the 256 bytes of Flash memory contained in a single row. This register is used for all I/O access to Flash memory. In addition, each access to the FLASH_DATA register causes an autoincrement of the Flash address stored in the Flash Address registers (FLASH_PAGE, FLASH_ROW, FLASH_COL). See Table 45.

Table 45. Flash Column Select Register (FLASH_COL = 00FEh)

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, R = Read Only.

Bit Position	Value	Description
[7:0] FLASH_COL	00h–FFh	Column address of Flash memory to be used during an I/O access of Flash memory.

Flash Program Control Register

The Flash Program Control Register is used to perform the functions of MASS ERASE, PAGE ERASE, and ROW PROGRAM. MASS ERASE and PAGE ERASE are self-clearing functions.

MASS ERASE requires approximately 200 ms to completely erase the full 256 KB of main Flash and the 512-byte information page if the FLASH_PAGE register(0x00FC) bit7(INFO_EN) is set. The 200 ms time is not reduced by excluding the 512 byte information page from erasing.

PAGE ERASE requires approximately 10 ms to erase a 2 KB page.

On completion of either a MASS ERASE or PAGE ERASE, the value of each corresponding bit is reset to 0.

When Flash is being erased, any Read or Write access to Flash forces the CPU into a Wait state until the Erase operation is complete and the Flash is accessed. Reads and Writes to areas other than Flash memory proceeds as usual while an Erase operation is underway.

During row programming, any reads of Flash memory force a WAIT condition until the row programming operation completes or times out. See Table 46 on page 113.

Table 46. Flash Program Control Register (FLASH_PGCTL = 00FFh)

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/W	R/W	R/W

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

Bit Position	Value	Description
[7:3]	00h	Reserved.
[2] ROW_PGM	0	Row Program Disable or Row Program completed.
	1	Row Program Enable. This bit automatically resets to 0 when the row address reaches 256 or when the Row Program operation times out.
[1] PG_ERASE	0	Page Erase Disable (Page Erase completed).
	1	Page Erase Enable. This bit automatically resets to 0 when the PAGE ERASE operation is complete.
[0] MASS_ERASE	0	Mass Erase Disable (Mass Erase completed).
	1	Mass Erase Enable. This bit automatically resets to 0 when the MASS ERASE operation is complete.

Watchdog Timer

The Watchdog Timer (WDT) helps protect against corrupt or unreliable software, power faults, and other system-level problems which places the CPU into unsuitable operating states. The eZ80F91 WDT features:

- Four programmable time-out ranges (depending on the WDT clock source). The four ranges are:
  - 03.2–5.20 ms
  - 51.2–83.9 ms
  - 0.50–0.82 sec
  - 2.68–4.00 sec
- Three selectable WDT clock sources:
  - Internal RC oscillator
  - System clock
  - Real-Time Clock source (on-chip 32 kHz crystal oscillator or 50/60 Hz signal)
- A selectable time-out response: a time-out is configured to generate either a RESET or a nonmaskable interrupt (NMI)
- A WDT time-out RESET indicator flag

Figure 25 displays a block diagram of the Watchdog Timer.

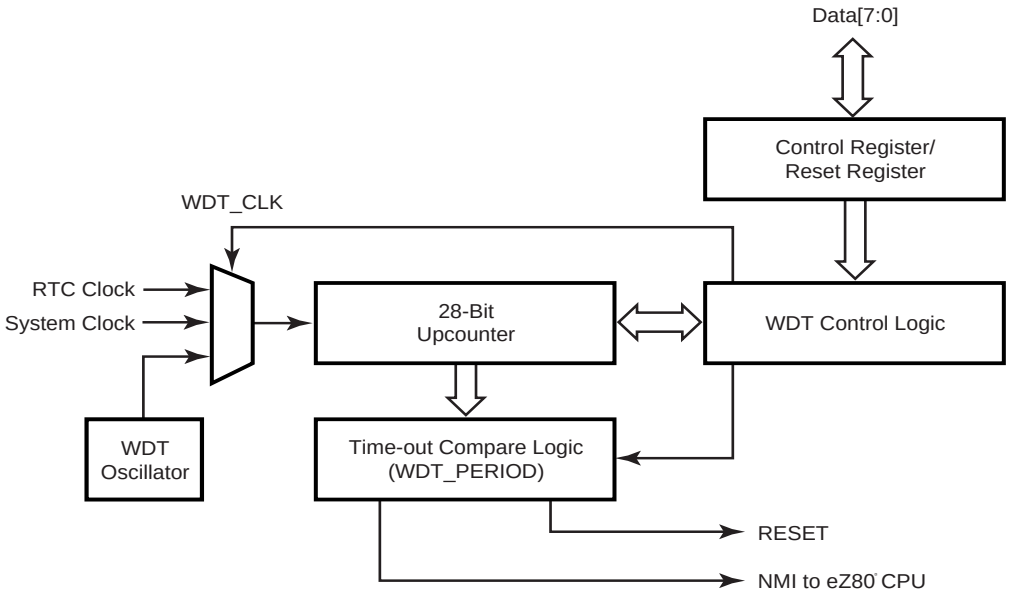


Figure 25. Watchdog Timer Block Diagram

Watchdog Timer Operation

Enabling and Disabling the Watchdog Timer

The WDT is disabled on a RESET. To enable the WDT, the application program must set WDT_EN, which is bit 7 of the WDT_CTL register. After WDT_EN is set, no Writes are allowed to the WDT_CTL register. When enabled, the WDT cannot be disabled except by a RESET.

Time-Out Period Selection

There are four choices of time-out periods for the WDT. The WDT time-out period is defined by the WDT_PERIOD WDT_CTL[1:0] field and WDT_CLK WDT_CTL[3:2] field of the Watchdog Timer control register (WDT_CTL = 0093h). The approximate time-out period and corresponding clock cycles for three different WDT clock sources are listed in Table 47.

The WDT time-out period divider is set to one of the four available settings for the selected frequency of the WDT clock source. Basing the divider settings on the clock source values provides a time-out range from few seconds to few msecs, regardless of the frequency setting.

Table 47. WDT Approximate Time-Out Delays for Possible Clock Sources

WDT_CLK[3:2]	**00**		**01**		**10**		**11**	
	50 MHz system clock		32.768 kHz RTC clock		Internal RC oscillator (~10 kHz)		Reserved	
WDT_PERIOD[1:0]	Divider	Timeout	Divider	Timeout	Divider	Timeout	Divider	Timeout
00	2^{27}	2.68 s	2^{17}	4.00 s	2^{15}	3.28 s	-	-
01	2^{25}	0.67 s	2^{14}	0.5 s	2^{13}	0.82 s	-	-
10	2^{22}	83.9 ms	2^{11}	62.5 ms	2^{9}	51.2 ms	-	-
11	2^{18}	5.2 ms	2^{7}	3.9 ms	2^{5}	3.2 ms	-	-

RESET or NMI Generation

A WDT time-out causes a RESET or sends a NMI signal to the CPU. The default operation is for the WDT to cause a RESET.

If the NMI_OUT bit in the WDT_CTL register is set to 0, then on a WDT time-out, the RST_FLAG bit in the WDT_CTL register is set to 1. The RST_FLAG bit is 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 on time-out, the WDT asserts an NMI for CPU processing. The NMI_FLAG bit is polled by the CPU to determine the source of the NMI event.

Watchdog Timer Registers

Watchdog Timer Control Register

The Watchdog Timer Control register (see Table 48) is an 8-bit Read/Write register used to enable the Watchdog Timer, set the time-out period, indicate the source of the most recent RESET or NMI, and select the required operation on WDT time-out.

The default clock source for the WDT is the WDT oscillator (WDT_CLK = 10b). To power-down the WDT oscillator, another clock source must be selected. The power-up sequence of the WDT oscillator takes approximately 20 ms.

Table 48. Watchdog Timer Control Register (WDT_CTL = 0093h)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0/1	0	1	0	0	0
CPU Access	R/W	R/W	R	R	R/W	R/W	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 RESET.
6 NMI_OUT	0	WDT time-out resets the CPU.
	1	WDT time-out generates a 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, only if the NMI_OUT flag is set to 0. The CPU polls this bit to determine the source of the RESET. This flag is cleared by a non-WDT generated reset.
4 NMI_FLAG	0	NMI caused by external source.
	1	NMI caused by WDT time-out. This flag is set by the WDT time-out, only if the NMI_OUT flag is set to 1. The CPU polls this bit to determine the source of the NMI. This flag is cleared by a non-WDT NMI.

Bit Position	Value	Description
[3:2] WDT_CLK	00	WDT clock source is system clock.
	01	WDT clock source is Real-Time Clock source (32 kHz on-chip oscillator or 50/60 Hz input as set by RTC_CTRL[4]).
	10	WDT clock source is internal RC oscillator (10 kHz typical).
	11	Reserved.
[1:0] WDT_PERIOD	00	WDT_CLK = 00 WDT time-out period is 2^{27} clock cycles.
		WDT_CLK = 01 WDT time-out period is 2^{17} clock cycles.
		WDT_CLK = 10 WDT time-out period is 2^{15} clock cycles.
		WDT_CLK = 11 Reserved.
	01	WDT_CLK = 00 WDT time-out period is 2^{25} clock cycles.
		WDT_CLK = 01 WDT time-out period is 2^{14} clock cycles.
		WDT_CLK = 10 WDT time-out period is 2^{13} clock cycles.
		WDT_CLK = 11 Reserved.
	10	WDT_CLK = 00 WDT time-out period is 2^{22} clock cycles.
		WDT_CLK = 01 WDT time-out period is 2^{11} clock cycles.
		WDT_CLK = 10 WDT time-out period is 2^{9} clock cycles.
		WDT_CLK = 11 Reserved.
	11	WDT_CLK = 00 WDT time-out period is 2^{18} clock cycles.
		WDT_CLK = 01 WDT time-out period is 2^{7} clock cycles.
		WDT_CLK = 10 WDT time-out period is 2^{5} clock cycles.
		WDT_CLK = 11 Reserved.

Note: When the WDT is enabled, no Writes are allowed to the WDT_CTL register.

Watchdog Timer Reset Register

The WDT Reset register (see Table 49) is an 8-bit Write only register. The WDT is reset when an A5h value followed by a 5Ah value is written to this register. Any amount of time occurs between the writing of A5h value and the 5Ah value, so long as the WDT time-out does not occur prior to completion. Any value other than 5Ah written to the WDT Reset register after the A5h value requires that the sequence of Writes (A5h,5Ah) be restarted for the timer to be reset.

Table 49. Watchdog Timer Reset Register (WDT_RR = 0094h)

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

Note: X = Undefined; W = Write Only.

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

Programmable Reload Timers

The eZ80F91 device features four programmable reload timers. The core of each timer is a 16-bit downcounter. In addition, each timer features a selectable clock source, adjustable prescaling and operates in either SINGLE PASS or CONTINUOUS mode.

In addition to the basic timer functionality, some of the timers support specialty modes that performs event counting, input capture, output compare, and Pulse-Width Modulation (PWM) generation functions. PWM mode supports four individually-configurable outputs and a power trip function.

Each of the four timers available on the eZ80F91 device are 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 displayed in Figure 26.

Each timer features its own interrupt which is triggered either by the timer reaching zero or after a successful comparison occurs. As with the other eZ80F91 interrupts, the priority is fully programmable.

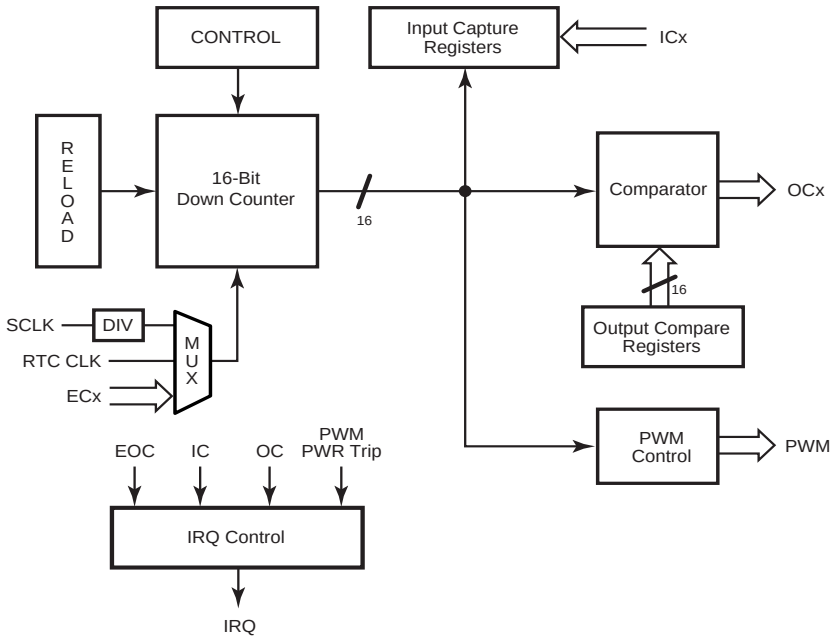


Figure 26. Programmable Reload Timer Block Diagram

Basic Timer Operation

Basic timer operation is controlled by a timer control register and a programmable reload value. The CPU uses the control register to setup the prescaling, the input clock source, the end-of-count behavior, and to start the timer. The 16-bit reload value is used to determine the duration of the timer's count before either halting or reloading.

After choosing a timer period and writing the appropriate values to the reload registers, the CPU must set the timer enable bit (TMR*x*_CTL[TIM_EN]) by allowing the count to begin. The reload bit (TMR*x*_CTL[RLD]) must also be asserted so that the timer counts down from the reload value rather than from 0000h. On the system clock cycle, after the assertion of the reload bit, the timer loads with the 16-bit reload value and begins counting down. The reload bit is automatically cleared after the loading operation. The timer is enabled and reloaded on the same cycle; however, the timer does not require disabling to reload and reloading is performed at any time. It is also possible to halt the timer by deasserting the timer enable bit and resuming the count at a later time from the same point by reasserting the bit.

Reading the Current Count Value

The CPU reads the current count value when the timer is running. Because the count is a 16-bit value, the hardware latches the value of the upper byte into temporary storage when the lower byte is read. This value in temporary storage is the value returned when the upper byte is read. Therefore, the software must read the lower byte first. If it attempts to read the upper byte first, it does not obtain the current upper byte of the count. Instead, it obtains the last latched value. This Read operation does not affect timer operation.

Setting Timer Duration

There are three factors to consider while 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 before the end-of-count is signaled. Depending on the TMR*x*_CTL[CLK_SEL] bits of the control register, the clock is either the system clock, or an on-chip RC oscillator output or an input from a pin.

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

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

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

Minimum time-out duration is four 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 TMR*x*_CTL[TIM_EN] bit resets to 0. To re-enable the timer, the CPU must set the TIM_EN bit to 1. An example of a PRT operating in SINGLE PASS mode is displayed in Figure 27. Timer register information is listed in Table 50.

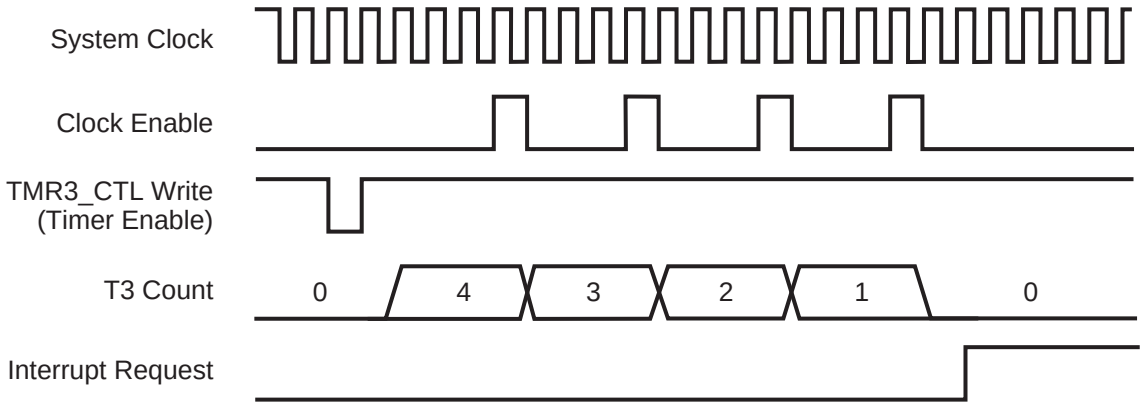


Figure 27. Example: PRT SINGLE PASS Mode Operation

Table 50. Example: PRT SINGLE PASS Mode Parameters

Parameter	Control Register(s)	Value
Timer Enable	TMR*x*_CTL[TIM_EN]	1
Reload	TMR*x*_CTL[RLD]	1
Prescaler Divider = 4	TMR*x*_CTL[CLK_DIV]	00b
SINGLE PASS Mode	TMR*x*_CTL[TIM_CONT]	0
End of Count Interrupt Enable	TMR*x*_IER[IRQ_EOC_EN]	1
Timer Reload Value	{TMR*x*_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 and the timer continues to count until disabled. An example of the timer operating in CONTINUOUS mode is displayed in Figure 28. Timer register information is listed in Table 51.

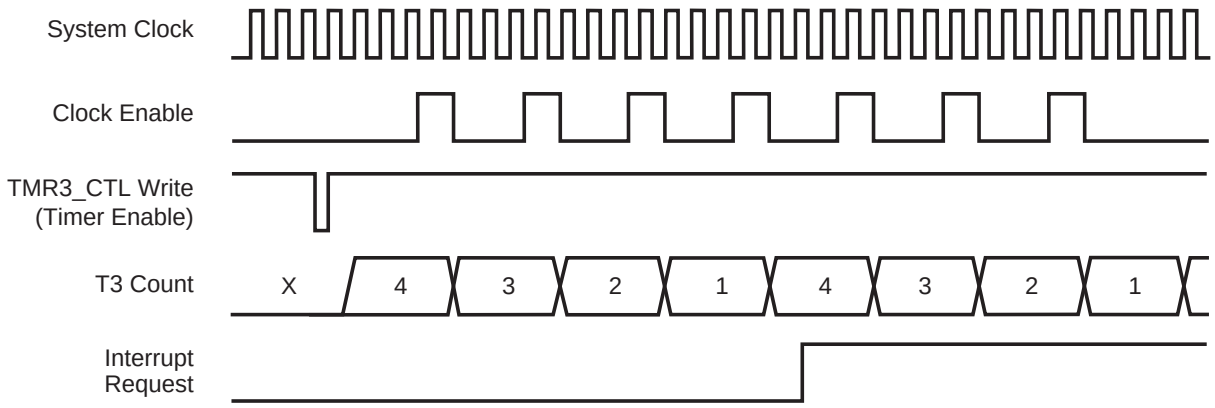


Figure 28. Example: PRT CONTINUOUS Mode Operation

Table 51. Example: PRT CONTINUOUS Mode Parameters

Parameter	Control Register(s)	Value
Timer Enable	TMR*x*_CTL[TIM_EN]	1
Reload	TMR*x*_CTL[RLD]	1
Prescaler Divider = 4	TMR*x*_CTL[CLK_DIV]	00b
CONTINUOUS Mode	TMR*x*_CTL[TIM_CONT]	1
End of Count Interrupt Enable	TMR*x*_IER[IRQ_EOC_EN]	1
Timer Reload Value	{TMR*x*_RR_H, TMR*x*_RR_L}	0004h

Timer Interrupts

The terminal count flag (TMR*x*_IIR[EOC]) is set to 1 whenever the timer reaches `0000h`, its end-of-count value in SINGLE PASS mode, or when the timer reloads the start value in CONTINUOUS mode. The terminal count 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 is programmed to poll the EOC bit for the time-out event. Alternatively, an interrupt service request signal is sent to the CPU by setting the TMR*x*_IER[EOC] bit to 1.

And when the end-of-count value (0000h) is reached, the EOC bit is set to 1 and an interrupt service request signal is passed to the CPU. The interrupt service request signal is deactivated by a CPU read of the timer interrupt identification register, TMR*x*_IIR. All bits in that register are reset by the Read.

The response of the CPU to this interrupt service request is a function of the CPU's interrupt enable flag, IEF1. For more information about this flag, refer to the *eZ80*® CPU *User Manual (UM0077)* available on www.zilog.com.

Timer Input Source Selection

Timers 0–3 features programmable input source selection. By default, the input is taken from the eZ80F91's system clock. The timers also use the Real-Time Clock source (50, 60, or 32768 Hz) as their clock sources. The input source for these timers is set using the timer control register. (TMR*x*_CTL[CLK_SEL])

Timer Output

The timer count is directed to the GPIO output pins, if required. To enable the Timer Output feature, the GPIO port pin must be configured as an output and for alternate functions. The GPIO output pin toggles each time the timer reaches its end-of-count value. In CONTINUOUS mode operation, enabling the Timer Output feature results in a Timer Output signal period which is twice the timer time-out period. Examples of Timer Output operation is displayed in Figure 29 on page 126 and listed in Table 52 on page 126. The initial value for the timer output is zero.

Logic to support timer output exists in all timers; but for the eZ80F91 device, only Timer 0 and 2 route the actual timer output to the pins. Because Timer 3 uses the T_{OUT} pins for PWMxN signals, the timer outputs are not available when using complementary PWM outputs. See Table 52 on page 126 for details.

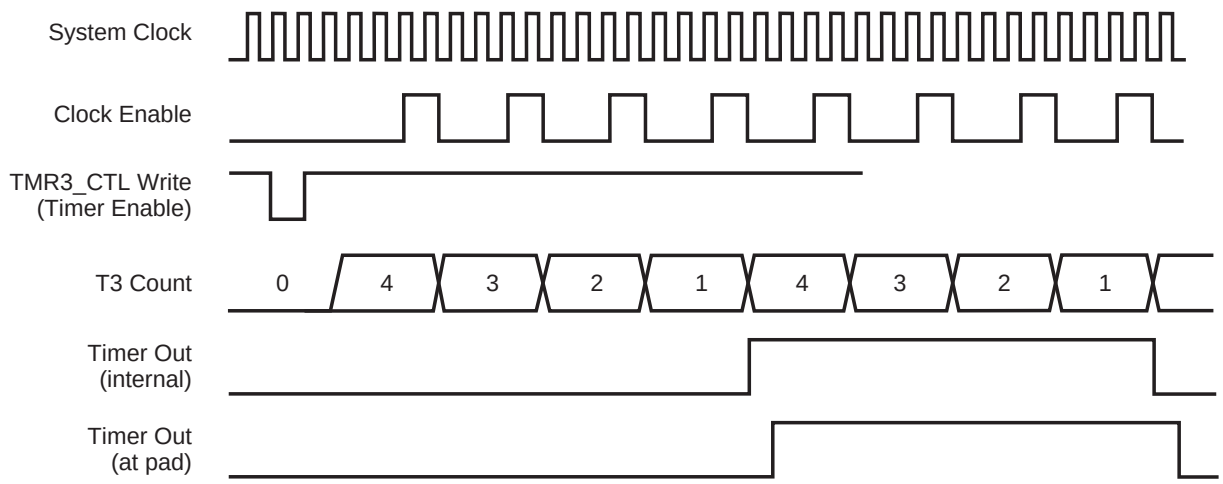


Figure 29. Example: PRT Timer Output Operation

Table 52. Example: PRT Timer Out Parameters

Parameter	Control Register(s)	Value
Timer Enable	TMR*x*_CTL[TIM_EN]	1
Reload	TMR*x*_CTL[RLD]	1
Prescaler Divider = 4	TMR*x*_CTL[CLK_DIV]	00b
CONTINUOUS Mode	TMR*x*_CTL[TIM_CONT]	1
Timer Reload Value	{TMR*x*_RR_H, TMRx_RR_L}	0003h

When the eZ80F91 device is running in DEBUG mode, encountering a break point causes all CPU functions to halt. However, the timers keep running. This instance makes debugging timer-related software much more difficult. Therefore, the control register contains a BRK_STP bit. Setting this bit causes the count value to be held during debug break points.

Specialty Timer Modes

The features described above are common to all timers in the eZ80F91 device. In addition to these common features, some of the timers have additional functionality.

The following is a list of the special features for each timer:

- Timer 0
  - No special functions
- Timer 1
  - One event counter (EC0)
  - Two input captures (IC0 and IC1)
- Timer 2
  - One event counter (EC1)
- Timer 3
  - Two input captures (IC2 and IC3)
  - Four output compares (OC0, OC1, OC2, and OC3)
  - Four PWM outputs (PWM0, PWM1, PWM2, and PWM3)

Timer 3 consists of three specialty modes. Each of these modes are enabled using bits in their respective control registers (TMR3_CAP_CTL, TMR3_OC_CTL1, TMR3_PWM_CTL1). When PWM mode is enabled, the OUTPUT COMPARE and INPUT CAPTURE modes are not available. This instance is due to address space sharing requirements. However, INPUT CAPTURE and OUTPUT COMPARE modes run simultaneously.

Timers with specialty modes offer multiple ways to generate an interrupt. When the interrupt controller services a timer interrupt, the software must read the timers interrupt identification register (TMR*x*_IIR) to determine the causes for an interrupt request. This register is cleared each time it is read, allowing subsequent events to be identified without interference from prior events.

Event Counter

When a timer is configured to take its input from a port input pin (ECx), it functions as an event counter. For event counting, the clock prescaler is automatically bypassed and edges (events) cause the timer to decrement. You must select the rising or the falling edge for counting. Also, the port pins must be configured as inputs.

Input sampling on the port pins results in the counter being updated on the third rising edge of the system clock after the edge event occurs at the port pin. Due to sampling, the frequency of the event input is limited to one-half the system clock frequency under ideal conditions. In practice, the event frequency must be less than this value due to duty cycle variation and system clock jitter.

This EVENT COUNT mode is identical to basic timer operation, except for the clock source. Therefore, interrupts are managed in the same manner.

RTC Oscillator Input

When the timer clock source is the Real-Time Clock (RTC) signal, the timer functions just as it does in EVENT COUNT mode, except that it samples the internal RTC clock rather than the ECx pin.

Input Capture

INPUT CAPTURE mode allows the CPU to determine the timing of specified events on a set of external pins.

A timer intended for use in INPUT CAPTURE mode is setup the same way as in BASIC mode, with one exception. The CPU must also write the TMR*x*_CAP_CTL register to select the edge on which to capture: rising, falling, or both. When one of these events occurs on an input capture pin, the current 16 bit timer value is latched into the capture value register pair (TMR*x*_CAP_A or TMR*x*_CAP_B depending on the IC pin exhibiting the event).

Reading the Low byte of the register pair causes the timer to ignore other capture events on the associated external pin until the High byte is read. This instance prevents a subsequent capture event from overwriting the High byte between the two Reads and generating an invalid capture value. The capture value registers are Read Only.

A capture flag (ICA or ICB) in the TMR*x*_IIR register is set whenever a capture event occurs. Setting the interrupt identification register bit TMR*x*_IER[IRQ_ICx_EN] enables the capture event to generate a timer interrupt. The port pins must be configured as alternate functions, see GPIO Mode 7—Alternate Functions on page 51.

Output Compare

The output compare function reverses the input capture function. Rather than store a timer value when an external event occurs, OUTPUT COMPARE mode waits until the timer reaches a specified value, then generates an external event. Although the same base timer is used, up to four separate external pins are driven each with its own compare value.

To use OUTPUT COMPARE mode, the CPU must first configure the basic timer parameters. Then it must load up to four 16-bit compare values into the four TMR3_OCx register pairs. Next, it must load the TMR3_ OC_CTL2 register to specify the event that occurs on comparison. You can select the following events: SET, CLEAR, and TOGGLE. Finally, the CPU must enable OUTPUT COMPARE mode by asserting TMR3_OC_CTL1[OC_EN].

The initial value for the OCx pins in OUTPUT COMPARE mode is 0 by default. It is possible to initialize this value to 1 or force a value at a later time. Setting the TMR3_OC_CTL2[OCx_MODE] value to 0 forces the OCx pin to the selected state provided by the TMR3_OC_CTL1[OCx_INIT] bits. Regardless of any compare events, the pin stays at the forced value until OCx_MODE is changed. After release, it retains the forced value until modified by an OUTPUT COMPARE event.

Asserting TMR3_OC_CTL1[MAST_MODE] selects MASTER MODE for all OUTPUT COMPARE events and sets output 0 as the master. As a result, outputs 1, 2, and 3 are caused to disregard output-specific configuration and comparison values and instead mimic the current settings for output 0.

The OCx bits in the TMR3_IIR register are set whenever the corresponding timer compares occur. TMR3_IER[IRQ_OCx_EN] allows the compare event to generate a timer interrupt.

Timer Port Pin Allocation

The eZ80F91 device timers interface to the outside world via Ports A and B. These ports are also used for GPIO as well as other assorted functions. Table 53 on page 129 lists the timer pins and their respective functions.

Table 53. GPIO Mode Selection Using Timer Pins

			Timer Function	
Port	**GPIO Port Bits**	**GPIO Port Mode**	PWM_CTL1 MPWM_EN = 0	PWM_CTL1 MPWM_EN = 1
A	PA0	7	OC0	PWM0
	PA1	7	OC1	PWM1
	PA2	7	OC2	PWM2
	PA3	7	OC3	PWM3
			PWM_CTL1 PAIR_EN = 0	PWM_CTL1 PAIR_EN = 1
	PA4	7	TOUT0	PWM0
	PA5	7	TOUT2	PWM1
	PA6	7	EC1	PWM2
	PA7	7		PWM3
B	PB0	7	IC0/EC0	
	PB1	7	IC1	
	PB4	7	IC2	
	PB5	7	IC3	

Timer Registers

The CPU monitors and controls the timer using seven 8-bit registers. These registers are the control register, the interrupt identification register, the interrupt enable register and the reload register pair (High and Low byte). There are also a pair of data registers used to read the current timer count value.

The variable *x* can be 0, 1, 2, or 3 to represent each of the four available timers.

Basic Timer Register Set

Each timer requires a different set of registers for configuration and control. However, all timers contain the following seven registers, each of which is necessary for basic operation:

- Timer Control Register (TMR*x*_CTL)
- Interrupt Identification Register (TMR*x*_IIR)
- Interrupt Enable Register (TMR*x*_IER)
- Timer Data Registers (TMR*x*_DR_H and TMR*x*_DR_L)
- Timer Reload Registers (TMR*x*_RR_H and TMR*x*_RR_L)

The Timer Data Register is Read Only, when the Timer Reload Register is Write Only. The address space for these two registers is shared.

Register Set for Capture in Timer 1

In addition to the basic register set, Timer 1 uses the following five registers for its INPUT CAPTURE mode:

- Capture Control Register (TMR1_CAP_CTL)
- Capture Value Registers (TMR1_CAP_B_H, TMR1_CAP_B_L, TMR1_CAP_A_H, TMR1_CAP_A_L)

Register Set for Capture/Compare/PWM in Timer 3

In addition to the basic register set, Timer 3 uses 19 registers for INPUT CAPTURE, OUTPUT COMPARE, and PWM modes. PWM and capture/compare functions cannot be used simultaneously so, their register address space is shared. INPUT CAPTURE and OUTPUT COMPARE are used concurrently and their address space is not shared.

The INPUT CAPTURE mode registers are equivalent to those used in Timer 1 above (substitute TMR3 for TMR1).

OUTPUT COMPARE mode uses the following nine registers:

- Output Compare Control Registers
  - TMR3_OC_CTL1

  - TMR3_OC_CTL2
- Compare Value Registers
  - TMR3_OC3_H
  - TMR3_OC3_L
  - TMR3_OC2_H
  - TMR3_OC2_L
  - TMR3_OC1_H
  - TMR3_OC1_L
  - TMR3_OC0_H
  - TMR3_OC0_L

Multiple PWM mode uses the following 19 registers:

- PWM Control Registers
  - TMR3_PWM_CTL1
  - TMR3_PWM_CTL2
  - TMR3_PWM_CTL3
- PWM Rising Edge Values
  - TMR3_PWM3R_H
  - TMR3_PWM3R_L
  - TMR3_PWM2R_H
  - TMR3_PWM2R_L
  - TMR3_PWM1R_H
  - TMR*x*_PWM1R_L
  - TMR3_PWM0R_H
  - TMR3_PWM0R_L
- PWM Falling Edge Values
  - TMR3_PWM3F_H
  - TMR*x*_PWM3F_L
  - TMR3_PWM2F_H
  - TMR3_PWM2F_L
  - TMR3_PWM1F_H
  - TMR3_PWM1F_L
  - TMR3_PWM0F_H
  - TMR3_PWM0F_L

Timer Control Register

The Timer x Control Register (see Table 54) is used to control timer operations including enabling the timer, selecting the clock source, selecting the clock divider, selecting between CONTINUOUS and SINGLEPASS modes, and enabling the auto-reload feature.

Table 54. Timer Control Register (TMR0_CTL = 0060h, TMR1_CTL = 0065h, TMR2_CTL = 006Fh, TMR3_CTL = 0074h)

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 BRK_STOP	0	The timer continues to operate during debug break points.
	1	The timer stops operation and holds count value during debug break points.
[6:5] CLK_SEL	00	Timer source is the system clock divided by the prescaler.
	01	Timer source is the Real Time Clock Input.
	10	Timer source is the Event Count (ECx) input—falling edge. For Timer 1 this is EC0. For Timer 2, this is EC1.
	11	Timer source is the Event Count (ECx) input—rising edge. For Timer 1 this is EC0. For Timer 2, this is EC1.
[4:3] CLK_DIV	00	System clock divider = 4.
	01	System clock divider = 16.
	10	System clock divider = 64.
	11	System clock divider = 256.
2 TIM_CONT	0	The timer operates in SINGLE PASS mode. TIM_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.

1 RLD	0	Reload function is not forced.
	1	Force reload. When 1 is written to this bit, the values in the reload registers are loaded into the downcounter.
0 TIM_EN	0	The programmable reload timer is disabled.
	1	The programmable reload timer is enabled.

Timer Interrupt Enable Register

The Timer x Interrupt Enable Register (see Table 55) is used to control timer interrupt operations. Only bits related to functions present in a given timer are active.

Table 55. Timer Interrupt Enable (TMR0_IER = 0061h, TMR1_IER = 0066h, TMR2_IER = 0070h, TMR3_IER = 0075h)

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	Unused.
6 IRQ_OC3_EN	0	Interrupt requests for OC3 are disabled (valid only in OUTPUT COMPARE mode). OC operations occur in Timer 3.
	1	Interrupt requests for OC3 are enabled (valid only in OUTPUT COMPARE mode). OC operations occur in Timer 3.
5 IRQ_OC2_EN	0	Interrupt requests for OC2 are disabled (valid only in OUTPUT COMPARE mode). OC operations occur in Timer 3.
	1	Interrupt requests for OC2 are enabled (valid only in OUTPUT COMPARE mode). OC operations occur in Timer 3.
4 IRQ_OC1_EN	0	Interrupt requests for OC1 are disabled (valid only in OUTPUT COMPARE mode). OC operations occur in Timer 3.
	1	Interrupt requests for OC1 are enabled (valid only in OUTPUT COMPARE mode). OC operations occur in Timer 3.
3 IRQ_OC0_EN	0	Interrupt requests for OC0 are disabled (valid only in OUTPUT COMPARE mode). OC operations occur in Timer 3.
	1	Interrupt requests for OC0 are enabled (valid only in OUTPUT COMPARE mode). OC operations occur in Timer 3.

2 IRQ_ICB_EN	0	Interrupt requests for ICx are disabled (valid only in INPUT CAPTURE mode). Timer 1: the capture pin is IC1. Timer 3: the capture pin is IC3.
	1	Interrupt requests for ICx are enabled (valid only in INPUT CAPTURE mode). For Timer 1: the capture pin is IC1. For Timer 3: the capture pin is IC3.
1 IRQ_ICA_EN	0	Interrupt requests for ICA or PWM power trip are disabled (valid only in INPUT CAPTURE and PWM modes). For Timer 1: the capture pin is IC0. For Timer 3: the capture pin is IC2.
	1	Interrupt requests for ICA or PWM power trip are enabled (valid only in INPUT CAPTURE and PWM modes). For Timer 1: the capture pin is IC0. For Timer 3: the capture pin is IC2.
0 IRQ_EOC_EN	0	Interrupt on end-of-count is disabled.
	1	Interrupt on end-of-count is enabled.

Timer Interrupt Identification Register

The TImer *x* Interrupt Identification Register (see Table 56) is used to flag timer events so that the CPU determines the cause of a timer interrupt. This register is cleared by a CPU Read.

Table 56. Timer Interrupt Identification Register (TMR0_IIR = 0062h, TMR1_IIR = 0067h, TMR2_IIR = 0071h, TMR3_IIR = 0076h)

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	Unused.
6 OC3	0	Output compare, OC3, does not occur.
	1	Output compare, OC3, occurs.
5 OC2	0	Output compare, OC2, does not occur.
	1	Output compare, OC2, occurs.
4 OC1	0	Output compare, OC1, does not occur.
	1	Output compare, OC1, occurs.
3 OC0	0	Output compare, OC0, does not occur.
	1	Output compare, OC0, occurs.
2 ICB	0	Input capture, ICB, does not occur. For Timer 1, the capture pin is IC1. For Timer 3, the capture pin is IC3.
	1	Input capture, ICB, occurs. For Timer 1, the capture pin is IC1. For Timer 3, the capture pin is IC3.
1 ICA	0	Input capture, ICA, or PWM power trip does not occur. For Timer 1, the capture pin is IC0. For Timer 3, the capture pin is IC2.
	1	Input capture, ICA, or PWM power trip occurs. For Timer 1, the capture pin is IC0. For Timer 3, the capture pin is IC2.
0 EOC	0	End-of-count does not occur.
	1	End-of-count occurs.

Timer Data Register—Low Byte

The Timer x Data Register—Low Byte returns the Low byte of the current count value of the selected timer. The Timer Data Register—Low Byte (see Table 57) is read when 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, {TMR*x*_DR_H[7:0], TMR*x*_DR_L[7:0]}, first read the Timer Data Register—Low Byte, followed by the Timer Data Register—High Byte. The Timer Data Register—High Byte value is latched into temporary storage when a Read of the Timer Data Register—Low Byte occurs.

This register shares its address with the corresponding timer reload register.

Table 57. Timer Data Register—Low Byte (TMR0_DR_L = 0063h, TMR1_DR_L = 0068h, TMR2_DR_L = 0072h, TMR3_DR_L = 0077h)

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] TMR_DR_L	00h–FFh	These bits represent the Low byte of the 2-byte timer data value, {TMR*x*_DR_H[7:0], TMR*x*_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 Register—High Byte

The Timer *x* Data Register—High Byte returns the High byte of the count value of the selected timer as it existed at the time that the Low byte was read. The Timer Data Register—High Byte (see Table 58) is read when 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, {TMR*x*_DR_H[7:0], TMR*x*_DR_L[7:0]}, first read the Timer Data Register—Low Byte followed by the Timer Data Register—High Byte. The Timer Data Register—High Byte value is latched into temporary storage when a Read of the Timer Data Register—Low Byte occurs.

This register shares its address with the corresponding timer reload register.

Table 58. Timer Data Register—High Byte (TMR0_DR_H = 0064h, TMR1_DR_H = 0069h, TMR2_DR_H = 0073h, TMR3_DR_H = 0078h)

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] TMR_DR_H	00h–FFh	These bits represent the High byte of the 2-byte timer data value, {TMR*x*_DR_H[7:0], TMR*x*_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 Register—Low Byte

The Timer x Reload Register—Low Byte (see Table 59) 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 on end-of-count. When the reload bit (TMR*x*_CTL[RLD]) is set to 1 forcing the reload function, the timer reload value is written to the timer on the next rising edge of the clock.

This register shares its address with the corresponding timer data register.

Table 59. Timer Reload Register—Low Byte (TMR0_RR_L = 0063h, TMR1_RR_L = 0068h, TMR2_RR_L = 0072h, TMR3_RR_L = 0077h)

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] TMR_RR_L	00h–FFh	These bits represent the Low byte of the 2-byte timer reload value, {TMR*x*_RR_H[7:0], TMR*x*_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 Register—High Byte

The Timer x Reload Register—High Byte (see Table 60) 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 the reload bit (TMR*x*_CTL[RLD]) is set to 1, it forces the reload function, the timer reload value is written to the timer on the next rising edge of the clock.

This register shares its address with the corresponding timer data register.

Table 60. Timer Reload Register—High Byte (TMR0_RR_H = 0064h, TMR1_RR_H = 0069h, TMR2_RR_H = 0073h, TMR3_RR_H = 0078h)

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] TMR_RR_H	00h–FFh	These bits represent the High byte of the 2-byte timer reload value, {TMR*x*_RR_H[7:0], TMR*x*_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 Capture Control Register

The Timer x Input Capture Control Register (see Table 61) is used to select the edge or edges to be captured. For Timer 1, CAP_EDGE_B is used for IC1 and CAP_EDGE_A is for IC0. For Timer 3, CAP_EDGE_B is for IC3, and CAP_EDGE_A is for IC2.

Table 61. Timer Input Capture Control Register (TMR1_CAP_CTL = 006Ah, TMR3_CAP_CTL = 007Bh)

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
[7:4]	0000	Reserved

[3:2] CAP_EDGE_B	00	Disable capture on ICB.
	01	Enable capture only on the falling edge of ICB.
	10	Enable capture only on the rising edge of ICB.
	11	Enable capture on both edges of ICB.
[1:0] CAP_EDGE_A	00	Disable capture on ICA.
	01	Enable capture only on the falling edge of ICA
	10	Enable capture only on the rising edge of ICA.
	11	Enable capture on both edges of ICA.

Timer Input Capture Value A Register—Low Byte

The Timer x Input Capture Value A Register—Low Byte (see Table 62) stores the Low byte of the capture value for external input A. For Timer 1, the external input is IC0. For Timer 3, it is IC2.

Table 62. Timer Input Capture Value Register A—Low Byte (TMR1_CAPA_L = 006Bh, TMR3_CAPA_L = 007Ch)

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] TMR*x*_CAPA_L	00h–FFh	These bits represent the Low byte of the 2-byte capture value, {TMR*x*_CAPA_H[7:0], TMR*x*_CAPA_L[7:0]}. Bit 7 is bit 7 of the 16-bit data value. Bit 0 is bit 0 (lsb) of the 16-bit timer data value.

Timer Input Capture Value A Register—High Byte

The Timer x Input Capture Value A Register—High Byte (see Table 63) stores the High byte of the capture value for external input A. For Timer 1, the external input is IC0. For Timer 3, it is IC2.

Table 63. Timer Input Capture Value Register A—High Byte (TMR1_CAPA_H = 006Ch, TMR3_CAPA_H = 007Dh)

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] TMR*x*_CAPA_H	00h–FFh	These bits represent the High byte of the 2-byte capture value, {TMR*x*_CAPA_H[7:0], TMR*x*_CAPA_L[7:0]}. Bit 7 is bit 15 (msb) of the 16-bit data value. Bit 0 is bit 8 of the 16-bit timer data value.

Timer Input Capture Value B Register—Low Byte

The Timer x Input Capture Value B Register—Low Byte (see Table 64) stores the Low byte of the capture value for external input B. For Timer 1, the external input is IC1. For Timer 3, it is IC3.

Table 64. Timer Input Capture Value Register B—Low Byte (TMR1_CAPB_L = 006Dh, TMR3_CAPB_L = 007Eh)

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] TMR*x*_CAPB_L	00h–FFh	These bits represent the Low byte of the 2-byte capture value, {TMR*x*_CAPB_H[7:0], TMR*x*_CAPB_L[7:0]}. Bit 7 is bit 7 of the 16-bit data value. Bit 0 is bit 0 (lsb) of the 16-bit timer data value.

Timer Input Capture Value B Register—High Byte

The Timer x Input Capture Value B Register—High Byte (see Table 65) stores the High byte of the capture value for external input B. For Timer 1, the external input is IC0. For Timer 3, it is IC3.

Table 65. Timer Input Capture Value Register B—High Byte (TMR1_CAPB_H = 006Eh, TMR3_CAPB_H = 007Fh)

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] TMR*x*_CAPB_H	00h–FFh	These bits represent the High byte of the 2-byte capture value, {TMR*x*_CAPB_H[7:0], TMR*x*_CAPB_L[7:0]}. Bit 7 is bit 15 (msb) of the 16-bit data value. Bit 0 is bit 8 of the 16-bit timer data value.

Timer Output Compare Control Register 1

The Timer3 Output Compare Control Register 1 (see Table 66) is used to select the Master Mode and to provide initial values for the OC pins.

Table 66. Timer Output Compare Control Register 1 (TMR3_OC_CTL1 = 0080h)

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
[7:6]	00	Unused.
5 OC3_INIT	0	OC pin cleared when initialized.
	1	OC pin set when initialized.
4 OC2_INIT	0	OC pin cleared when initialized.
	1	OC pin set when initialized.

3 OC1_INIT	0	OC pin cleared when initialized.
	1	OC pin set when initialized.
2 OC0_INIT	0	OC pin cleared when initialized.
	1	OC pin set when initialized.
1 MAST_MODE	0	OC pins are independent.
	1	OC pins all mimic OC0.
0 OC_EN	0	OUTPUT COMPARE mode is disabled.
	1	OUTPUT COMPARE mode is enabled.

Timer Output Compare Control Register 2

The Timer3 Output Compare Control Register 2 (see Table 67) is used to select the event that occurs on the output compare pins when a timer compare happens.

Table 67. Timer Output Compare Control Register 2 (TMR3_OC_CTL2 = 0081h)

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] OC3_MODE	00	Initialize OC pin to value specified in TMR3_OC_CTL1[OC3_INT].
	01	OC pin is cleared upon timer compare.
	10	OC pin is set upon timer compare.
	11	OC pin toggles upon timer compare.
[5:4] OC2_MODE	00	Initialize OC pin to value specified in TMR3_OC_CTL1[OC2_INT].
	01	OC pin is cleared upon timer compare.
	10	OC pin is set upon timer compare.
	11	OC pin toggles upon timer compare.

[3:2] OC1_MODE	00	Initialize OC pin to value specified in TMR3_OC_CTL1[OC1_INT].
	01	OC pin is cleared upon timer compare.
	10	OC pin is set upon timer compare.
	11	OC pin toggles upon timer compare.
[1:0] OC0_MODE	00	Initialize OC pin to value specified in TMR3_OC_CTL1[OC0_INT].
	01	OC pin is cleared upon timer compare.
	10	OC pin is set upon timer compare.
	11	OC pin toggles upon timer compare.

Timer Output Compare Value Register—Low Byte

The Timer3 Output Compare x Value Register—Low Byte (see Table 68) stores the Low byte of the compare value for OC0–OC3.

Table 68. Compare Value Register—Low Byte (TMR3_OC0_L = 0082h, TMR3_OC1_L = 0084h, TMR3_OC2_L = 0086h, TMR3_OC3_L = 0088h)

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] TMR3_OCx_L	00h–FFh	These bits represent the Low byte of the 2-byte compare value, {TMR3_OCx_H[7:0], TMR3_OCx_L[7:0]}. Bit 7 is bit 7 of the 16-bit data value. Bit 0 is bit 0 (lsb) of the 16-bit timer compare value.

Timer Output Compare Value Register—High Byte

The Timer3 Output Compare x Value Register—High Byte (see Table 69) stores the High byte of the compare value for OC0–OC3.

Table 69. Compare Value Register—High Byte (TMR3_OC0_H = 0083h, TMR3_OC1_H = 0085h, TMR3_OC2_H = 0087h, TMR3_OC3_H = 0089h)

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] TMR3_OCx_H	00h–FFh	These bits represent the High byte of the 2-byte compare value, {TMR3_OCx_H[7:0], TMR3_OCx_L[7:0]}. Bit 7 is bit 15 (msb) of the 16-bit data value. Bit 0 is bit 8 of the 16-bit timer compare value.

Multi-PWM Mode

The special Multi-PWM mode uses the Timer 3 16-bit counter as the primary timekeeper to control up to four PWM generators. The 16-bit reload value for Timer 3 sets a common period for each of the PWM signals. However, the duty cycle and phase for each generator are independent that is, the High and Low periods for each PWM generator are set independently. In addition, each of the four PWM generators are enabled independently. The eight PWM signals (four PWM output signals and their inverses) are output via Port A. A functional block diagram of the Multi-PWM is displayed in Figure 30 on page 146.

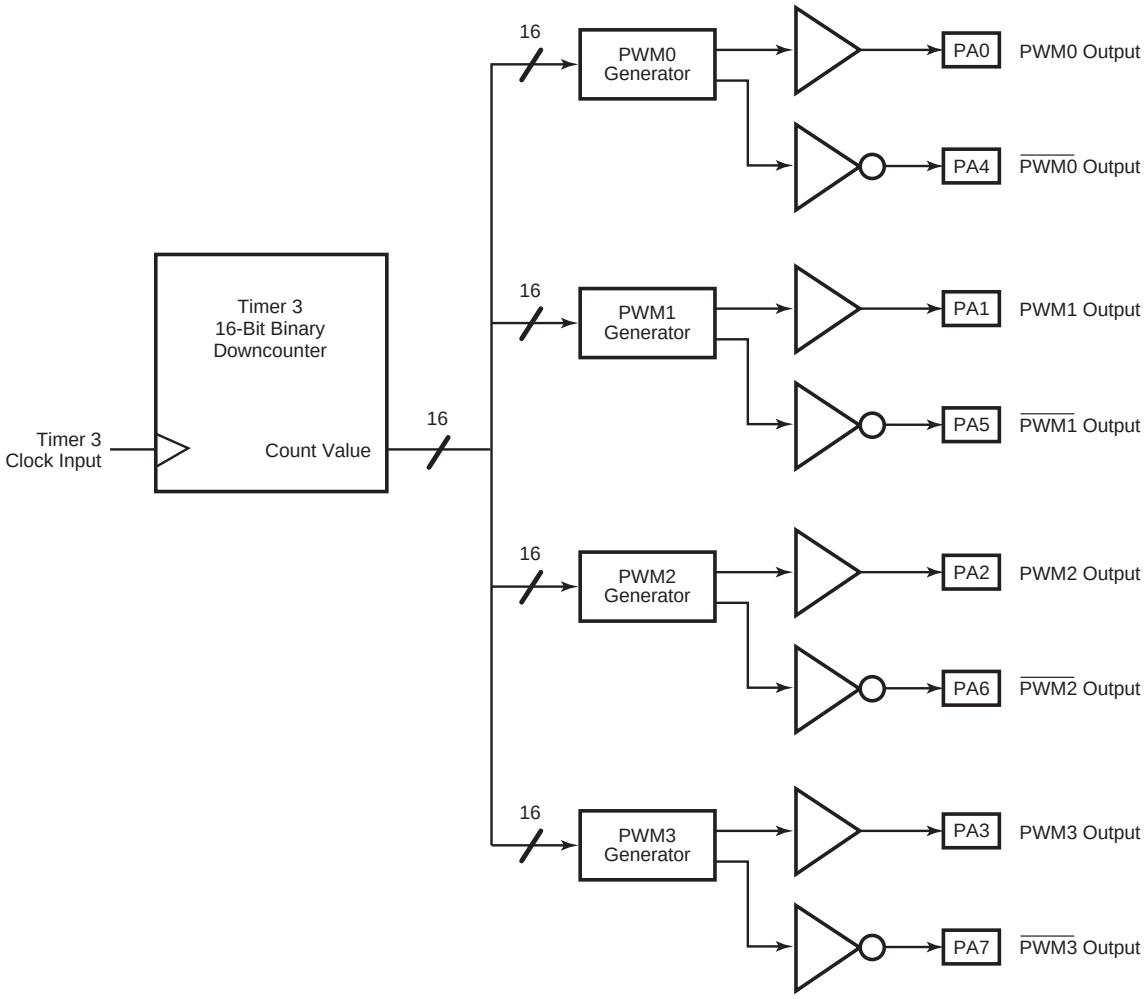


Figure 30. Multi-PWM Simplified Block Diagram

Setting TMR3_PWM_CTL1[MPWM_EN] to 1 enables Multi-PWM mode. The TMR3_PWM_CTL1 register bits enable the four individual PWM generators by adjusting settings according to the list provided in Table 70.

Table 70. Enabling PWM Generators

Enable PWM generator 0 by setting TMR3_PWM_CTL1[PWM0_EN] to 1.
Enable PWM generator 1 by setting TMR3_PWM_CTL1[PWM1_EN] to 1.
Enable PWM generator 2 by setting TMR3_PWM_CTL1[PWM2_EN] to 1.
Enable PWM generator 3 by setting TMR3_PWM_CTL1[PWM3_EN] to 1.

The inverted PWM outputs $\overline{\text{PWM0}}$, $\overline{\text{PWM1}}$, $\overline{\text{PWM2}}$, and $\overline{\text{PWM3}}$ are globally enabled by setting TMR3_PWM_CTL1[PAIR_EN] to 1. The individual PWM generators must be enabled for the associated inverted PWM signals to be output.

For each of the 4 PWM generators, there is a 16-bit rising edge value {TMR3_PWM*x*R_H[PWMxR_H], TMR3_PWM*x*R_L[PWMxR_L]} and a 16-bit falling edge value {TMR3_PWM*x*F_H[PWMxF_H], TMR3_PWM*x*F_L[PWMxF_L]} for a total of 16 registers. The rising-edge byte pairs define the timer count at which the PWM*x* output transitions from Low to High. Conversely, the falling-edge byte pairs define the timer count at which the PWM*x* output transitions from High to Low. On reset, all enabled PWM outputs begin Low and all $\overline{\text{PWMx}}$ outputs begin High. When the PWMx output is Low, the logic is looking for a match between the timer count and the rising edge value, and vice versa. Therefore, in a case in which the rising edge value is the same as the falling edge value, the PWM output frequency is one-half the rate at which the counter passes through its entire count cycle (from reload value down to 0000h).

Figure 31and Figure 32 display a simple Multi-PWM output and an expanded view of the timing, respectively. Associated control values are listed in Table 71 on page 148.

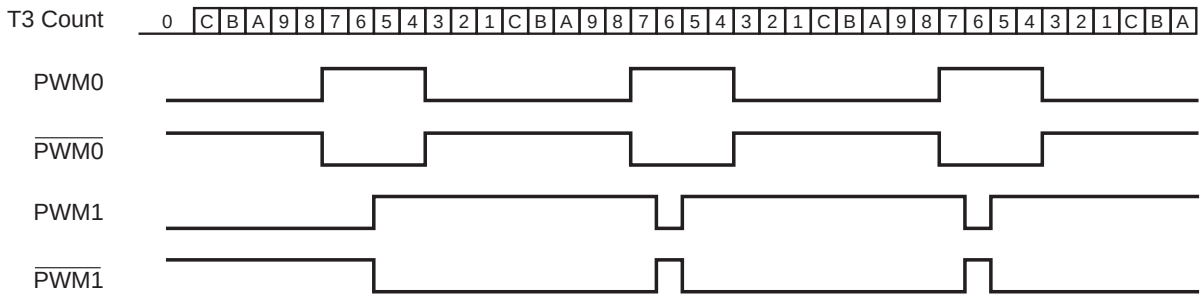


Figure 31. Multi-PWM Operation

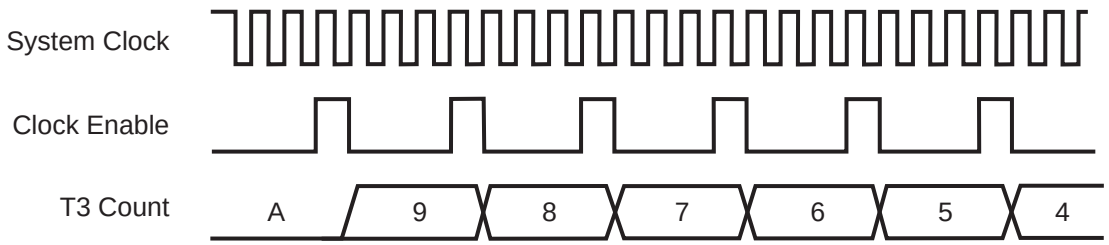


Figure 32. Multi-PWM Operation—Expanded View of Timing

Table 71. Example: Multi-PWM Addressing

Parameter	Control Register(s)	Value
Timer Reload Value	{TMR3_RR_H, TMR3_RR_L}	000Ch
PWM0 rising edge	{TMR3_PWM0R_H, TMR3_PWM0R_L}	0008h
PWM0 falling edge	{TMR3_PWM0F_H, TMR3_PWM0F_L}	0004h
PWM1 rising edge	{TMR3_PWM1R_H, TMR3_PWM1R_L}	0006h
PWM1 falling edge	{TMR3_PWM1F_H, TMR3_PWM1F_L}	0007h
$\overline{\text{PWM}}$ enable	TMR3_PWM_CTL1[PAIR_EN]	1
PWM0 enable	TMR3_PWM_CTL1[PWM0_EN]	1
PWM1 enable	TMR3_PWM_CTL1[PWM1_EN]	1
Multi-PWM enable	TMR3_PWM_CTL1[MPWM_EN]	1
Prescaler Divider = 4	TMR3_CTL[CLK_DIV]	00b
PWM nonoverlapping delay = 0	TMR3_PWM_CTL2[PWM_DLY]	0000b

PWM Master Mode

In PWM Master mode, the pair of output signals generated from the PWM0 generator (PWM0 and $\overline{\text{PWM0}}$) are directed to all four sets of PWM output pairs. Setting TMR3_PWM_CTL1[MM_EN] to 1 enables PWM Master mode. Assuming the outputs are all enabled and no AND/OR gating is used, all four PWM output pairs transition simultaneously under the direction of PWM0 and $\overline{\text{PWM0}}$. In PWM Master mode, the outputs still be gated individually using the AND/OR gating functions described in the next section. Multi-PWM mode and the individual PWM outputs must be enabled along with PWM Master mode. It is possible to enable or disable any combination of the four PWM outputs while running in PWM Master mode.

Modification of Edge Transition Values

Special circuitry is included for the update of the PWM edge transition values. Normal use requires that these values be updated while the PWM generator is running.

Note: *Under certain circumstances, electric motors driven by the PWM logic encounters rough operation. In other words, cycles are skipped if the PWM waveform edge is not carefully modified.*

Without special consideration, if a PWM generator looks for a particular count to make a state transition and if the edge transition value changes to a value that already occurred in

the current counter count-down cycle, then the transition is missed. The PWM generator holds the current output state until the counter reloads and cycles through to the appropriate edge transition value again. In effect, an entire cycle of the PWM waveform is skipped with the signal held at a DC value. The change in PWM waveform duty cycle from cycle to cycle must be limited to some fraction of a period to avoid rough running. To avoid unintentional roughness due to timing of the load operation for the register values in question, the PWM edge transition values are double-buffered and exhibit the following behavior:

- When the PWM generators are disabled, PWM edge transition values written by the CPU are immediately loaded into the PWM edge transition registers.
- When the PWM generators are enabled, a PWM edge transition value is loaded into a buffer register and transferred to its destination register only during a specific transition event. A rising edge transition value is only loaded upon a falling edge transition event, and a falling edge transition value is only loaded upon a rising edge transition event.

AND/OR Gating of the PWM Outputs

When in Multi-PWM mode, it is possible for you to turn off PWM propagation to the pins without disabling the PWM generator. This feature is global and applies to all enabled PWM generators. The function is implemented by applying digital logic (AND or OR functions) to combine the corresponding bits in the port output register with the PWM and $\overline{\text{PWM}}$ outputs.

The AND or OR functions are enabled on all PWM outputs by setting TMR3_PWM_CTL2[AO_EN] to either a `01b` (AND) or `10b` (OR). Any other value disables this feature. Likewise, the AND or OR functions are enabled on all $\overline{\text{PWM}}$ outputs by setting TMR3_PWM_CTL2[AON_EN] to either a `01b` (AND) or `10b` (OR). Any other value disables this feature. A functional block diagram for the AND/OR gating feature for PWM0 and $\overline{\text{PWM0}}$ is displayed in Figure 33 on page 150. The functionality for the other three PWM pairs are identical.

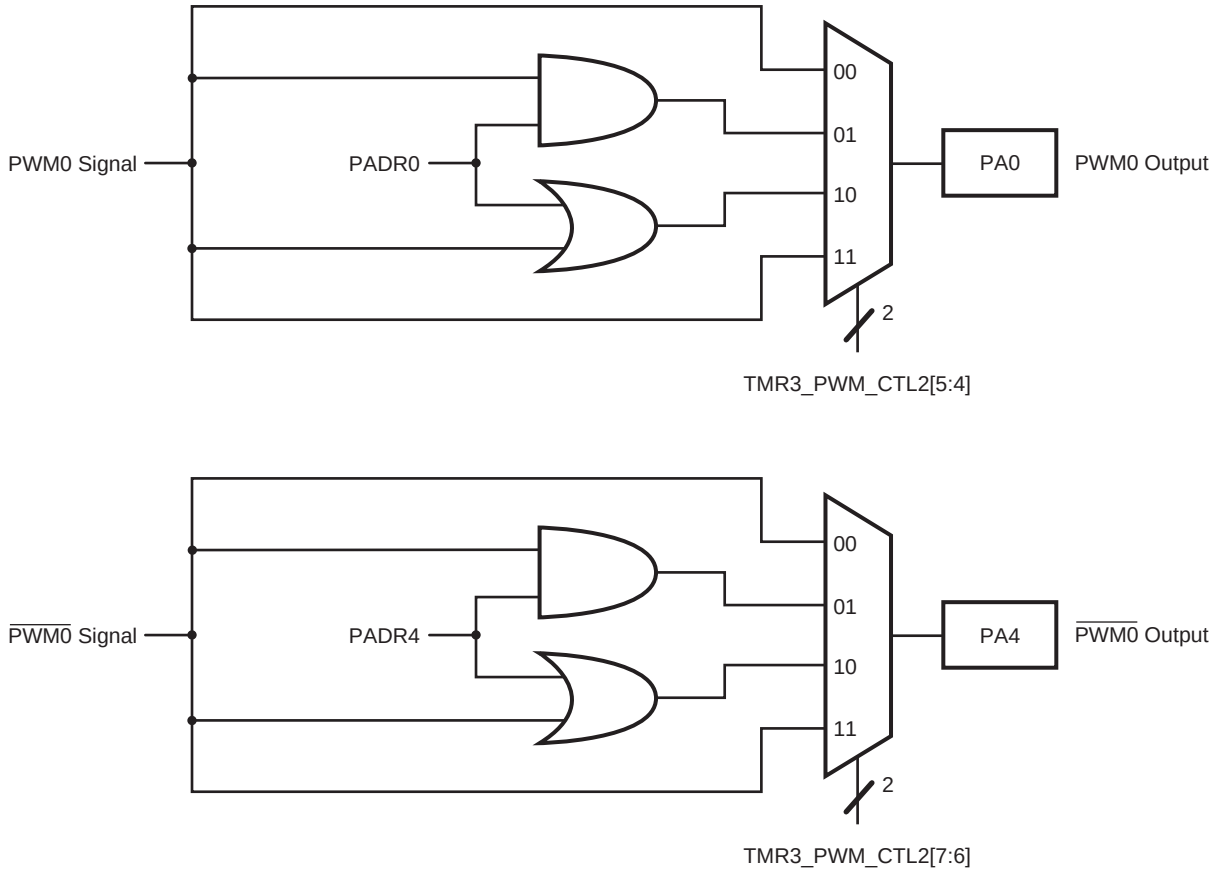


Figure 33. PWM AND/OR Gating Functional Diagram

If you enable the OR function on all PWM outputs and PADR0 is set to 1, then the PWM0 output on PA0 is forced High. Similarly, if you select the AND function on all PWM outputs and PADR0 is set to a 0, then the PWM0 output on PA0 is forced Low.

PWM Nonoverlapping Output Pair Delays

A delay is added between the falling edge of the PWM ($\overline{PWM}$) outputs and the rising edge of the $\overline{PWM}$ (PWM) outputs. This delay is set to assure that even with load and output drive variations there will be no overlap between the falling edge of a PWM ($\overline{PWM}$) output and the rising edge of its paired output. The selected delay is global to all four PWM pairs. The delay duration is software-selectable using the 4-bit field TMR3_PWM_CTL2[PWM_DLY]. The duration is programmable in units of the system clock (SCLK), from 0 SCLK periods to 15 SCLK periods. The TMR3_PWM_CTL2[PWM_DLY] bits are mapped directly to a counter, such that a

setting of 0000b represents a delay of 0 system clock periods and a setting of 1111b represents a delay of 15 system clock periods. The PWM delay feature is displayed in Figure 34 with associated addressing listed in Table 72.

Note: *The PWM nonoverlapping delay time must always be defined to be less than the delay between the rising and falling edges (and the delay between the falling and rising edges) of all Multi-PWM outputs. In other words, a rising (falling) edge cannot be delayed beyond the time at which it is subsequently scheduled to fall (rise).*

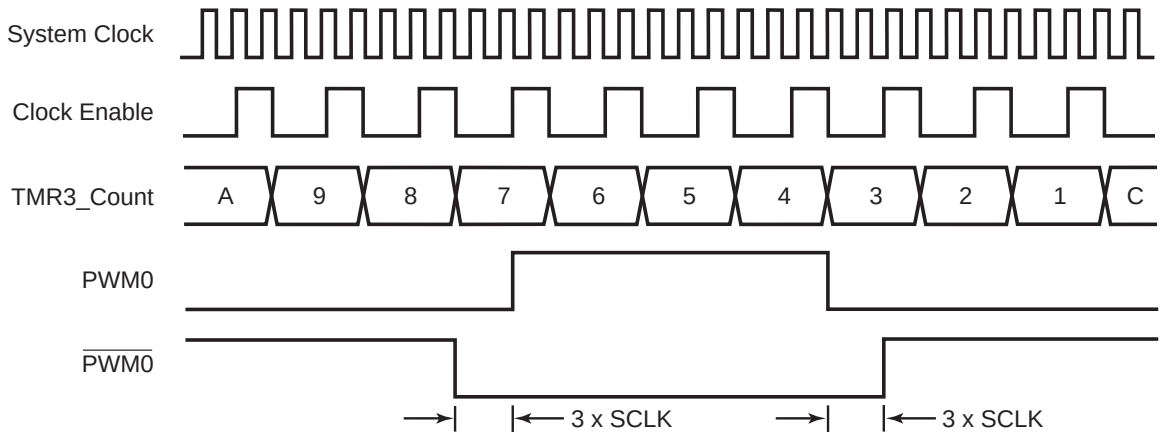


Figure 34. PWM Nonoverlapping Output Delay

Table 72. PWM Nonoverlapping Output Addressing

Parameter	Control Register(s)	Value
Timer clock is SCLK ÷ 4	TMR3_CTL[CLK_DIV]	00b
Timer reload value	{TMR3_RR_H, TMR3_RR_L}	000Ch
PWM0 rising edge	{TMR3_PWM0R_H, TMR3_PWM0R_L}	0008h
PWM0 falling edge	{TMR3_PWM0F_H, TMR3_PWM0F_L}	0004h
Prescaler divider = 4	TMR3_CTL[CLK_DIV]	00b
PWM nonoverlapping delay = 3	TMR3_PWM_CTL2[PWM_DLY]	0011b
$\overline{\text{PWM}}$ enable	TMR3_PWM_CTL1[PAIR_EN]	1
PWM0 enable	TMR3_PWM_CTL1[PWM0_EN]	1
Multi-PWM enable	TMR3_PWM_CTL1[MPWN_EN]	1

Multi-PWM Power-Trip Mode

When enabled, the Multi-PWM power-trip feature forces the enabled PWM outputs to a predetermined state when an interrupt is generated from an external source via IC0, IC1, IC2, or IC3. One or multiple external interrupt sources are enabled at any given time. If multiple sources are enabled, any of the selected external sources trigger an interrupt. Configuring the PWM_CTL3 register enables or disables interrupt sources. See Table 75 on page 156.

The possible interrupt sources for a Multi-PWM power-trip are:

- IC0—digital input
- IC1—digital input
- IC2—digital input
- IC3—digital input

When the power-trip is detected, TMR3_PWM_CTL3[PTD] is set to 1 to indicate detection of the power-trip. A value of 0 signifies that no power-trip is detected.

The PWMs are released only after a power-trip when TMR3_PWM_CTL3[PTD] is written back to 0 by software. As a result, you are allowed to check the conditions of the motor being controlled before releasing the PWMs. The explicit release also prevents noise glitches after a power-trip from causing an accidental exit or re-entry of the PWM power-trip state.

The programmable power-trip states of the PWMs are globally grouped for the PWM outputs and the inverting $\overline{\text{PWM}}$ outputs. Upon detection of a power-trip, the PWM outputs are forced to either a High state, a Low state, or high-impedance. The settings for the power-trip states are made with power-trip control bits TMR3_PWM_CTL3[PT_LVL], TMR3_PWM_CTL3[PT_LVL_N], and TMR3_PWM_CTL3[PT_TRI].

Multi-PWM Control Registers

Pulse-Width Modulation Control Register 1

The PWM Control Register 1 (see Table 73) controls PWM function enables.

Table 73. PWM Control Register 1 (PWM_CTL1 = 0079h)

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 PAIR_EN	0	Global disable of the $\overline{\text{PWM}}$ outputs (PWM outputs enabled only).
	1	Global enable of the PWM and $\overline{\text{PWM}}$ output pairs.
6 PT_EN	0	Disable power-trip feature.
	1	Enable power-trip feature.
5 MM_EN	0	Disable Master mode.
	1	Enable Master mode.
4 pwm3_en	0	Disable PWM generator 3.
	1	Enable PWM generator 3.
3 pwm2_en	0	Disable PWM generator 2.
	1	Enable PWM generator 2.
2 pwm1_en	0	Disable PWM generator 1.
	1	Enable PWM generator 1.
1 PWM0_EN	0	Disable PWM generator 0.
	1	Enable PWM generator 0.
0 mpwm_en	0	Disable Multi-PWM mode.
	1	Enable Multi-PWM mode.

Pulse-Width Modulation Control Register 2

The PWM Control Register 2 (see Table 74) controls pulse-width modulation AND/OR and edge delay functions.

Table 74. PWM Control Register 2 (PWM_CTL2 = 007Ah)

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] AON_EN	00	Disable AND/OR features on $\overline{\text{PWM}}$
	01	Enable AND logic on $\overline{\text{PWM}}$
	10	Enable OR logic on $\overline{\text{PWM}}$
	11	Disable AND/OR features on $\overline{\text{PWM}}$
[5:4] AO_EN	00	Disable AND/OR features on PWM
	01	Enable AND logic on PWM
	10	Enable OR logic on PWM
	11	Disable AND/OR features on PWM

[3:0] PWM_DLY	0000	No delay between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	0001	Delay of 1 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	0010	Delay of 2 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	0011	Delay of 3 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	0100	Delay of 4 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	0101	Delay of 5 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	0110	Delay of 6 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	0111	Delay of 7 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	1000	Delay of 8 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	1001	Delay of 9 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	1010	Delay of 10 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	1011	Delay of 11 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	1100	Delay of 12 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	1101	Delay of 13 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	1110	Delay of 14 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)
	1111	Delay of 15 SCLK periods between falling edge of PWM ($\overline{PWM}$) and rising edge of $\overline{PWM}$ (PWM)

Pulse-Width Modulation Control Register 3

The PWM Control Register 3 (see Table 75) is used to configure the PWM power trip functionality.

Table 75. PWM Control Register 3 (PWM_CTL3 = 007Bh)

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

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

Bit Position	Value	Description
7 PT_IC3_EN	0	Power trip disabled on IC3.
	1	Power trip enabled on IC3.
6 PT_IC2_EN	0	Power trip disabled on IC2.
	1	Power trip enabled on IC2.
5 PT_IC1_EN	0	Power trip disabled on IC1.
	1	Power trip enabled on IC1.
4 PT_IC0_EN	0	Power trip disabled on IC0.
	1	Power trip enabled on IC0.
3 PT_TRI	0	All PWM trip levels are open-drain.
	1	All PWM trip levels are defined by PT_LVL and PT_LVL_N.
2 PT_LVL	0	After power trip, PWMx outputs are set to one.
	1	After power trip, PWMx outputs are set to zero.
1 PT_LVL_N	0	After power trip, $\overline{\text{PWMx}}$ outputs are set to one.
	1	After power trip, $\overline{\text{PWMx}}$ outputs are set to zero.
0 PTD	0	Power trip has been cleared.
	1	This bit is set after power trip event.

Pulse-Width Modulation Rising Edge—Low Byte

A parallel 16-bit Write of {TMR3_PWMxR_H[7–0], TMR3_PWMxR_L[7–0]} occurs when software initiates a Write to TMR3_PWMxR_L. The register is listed in Table 76.

Table 76. PWMx Rising-Edge Register—Low Byte (TMR3_PWM0R_L = 007Ch, TMR3_PWM1R_L = 007Eh, TMR3_PWM2R_L = 0080h, TMR3_PWM3R_L = 0082h)

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] PWMXR_L	00h–FFh	These bits represent the Low byte of the 16-bit value to set the rising edge COMPARE value for PWMx, {TMR3_PWMXR_H[7:0], TMR3_PWMXR_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.

Pulse-Width Modulation Rising Edge—High Byte

Writing to TMR3_PWMxR_H stores the value in a temporary holding register. A parallel 16-bit Write of {TMR3_PWMxR_H[7–0], TMR3_PWMxR_L[7–0]} occurs when software initiates a Write to TMR3_PWMxR_L. The register is listed in Table 77.

Table 77. PWMx Rising-Edge Register—High Byte (TMR3_PWM0R_H = 007Dh, TMR3_PWM1R_H = 007Fh, TMR3_PWM2R_H = 0081h, TMR3_PWM3R_H = 0083h)

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] PWMXR_H	00h–FFh	These bits represent the High byte of the 16-bit value to set the rising edge COMPARE value for PWMx, {TMR3_PWMXR_H[7:0], TMR3_PWMXR_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.

Pulse-Width Modulation Falling Edge—Low Byte

A parallel 16-bit Write of {TMR3_PWMxF_H[7–0], TMR3_PWMxF_L[7–0]} occurs when software initiates a Write to TMR3_PWMxF_L. The register is listed in Table 78.

Table 78. PWMx Falling-Edge Register—Low Byte (TMR3_PWM0F_L = 0084h, TMR3_PWM1F_L = 0086h, TMR3_PWM2F_L = 0088h, TMR3_PWM3F_L = 008Ah)

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] PWMXF_L	00h–FFh	These bits represent the Low byte of the 16-bit value to set the falling edge COMPARE value for PWMx, {TMR3_PWMXF_H[7:0], TMR3_PWMXF_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.

Pulse-Width Modulation Falling Edge—High Byte

Writing to TMR3_PWMxF_H stores the value in a temporary holding register. A parallel 16-bit Write of {TMR3_PWMxF_H[7–0], TMR3_PWMxF_L[7–0]} occurs when software initiates a Write to TMR3_PWMxF_L. The register is listed in Table 79.

Table 79. PWMx Falling-Edge Register—High Byte (TMR3_PWM0F_H = 0085h, TMR3_PWM1F_H = 0087h, TMR3_PWM2F_H = 0089h, TMR3_PWM3F_H = 008Bh)

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] PWMXF_H	00h–FFh	These bits represent the High byte of the 16-bit value to set the falling edge COMPARE value for PWMx, {TMR3_PWMXF_H[7:0], TMR3_PWMXF_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.

Real-Time Clock

Real-Time Clock Overview

The Real-Time Clock (RTC) maintains time by keeping 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) operations. 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, 32 kHz oscillator is displayed in Figure 35. Connections to an external battery supply and 32 kHz crystal network is also displayed in Figure 35.

Note: *For users NOT using the RTC the following RTC signal pins must be connected as follows to avoid a 10 uA leakage within the RTC circuit block. RTC_Xin (pin 61) must be left floating or connected to ground.*

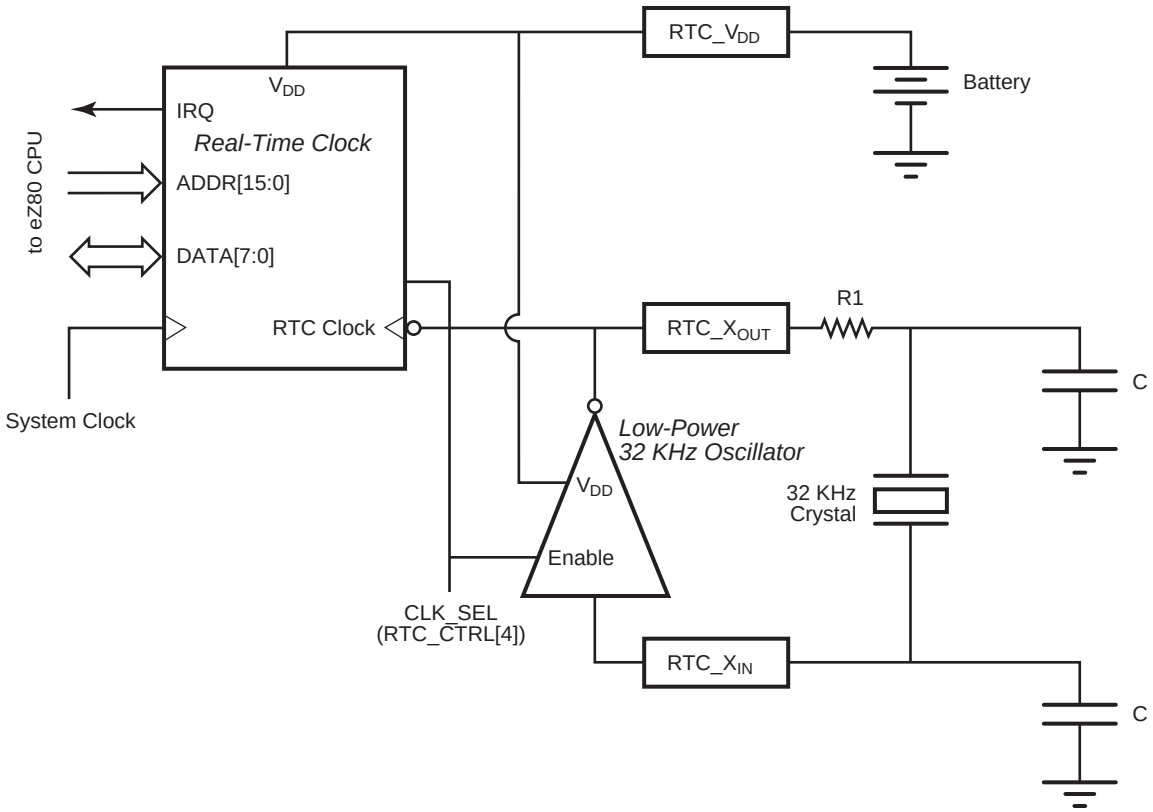


Figure 35. Real-Time Clock and 32 kHz Oscillator Block Diagram

Real-Time Clock Alarm

The clock is 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 is 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 a specified hour on a specified day. The alarm triggers an interrupt if the interrupt enable bit, INT_EN, is set to 1. 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 are 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 is also forced to perform a comparison at any time by writing a 0 to the RTC_UNLOCK bit (the RTC_UNLOCK bit 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 32 kHz RTC oscillator or an external 50/60 Hz CMOS-level clock signal (typically derived from the AC power line frequency). The on-chip oscillator requires an external 32 kHz crystal connected to RTC_X_{IN} and RTC_X_{OUT} as displayed in Figure 35 on page 159. If an external 50/60 Hz clock signal is used, connect it to RTC_$X_{OUT.}$

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 RTC and associated low-power 32 kHz oscillator is isolated from the other power supply pins on the eZ80F91 device. To ensure that the RTC continues to keep time in the event of loss of line power to the application, a battery is used to supply power to the RTC and the oscillator via the RTC_V_{DD} pin. All V_{SS} (ground) pins must be connected together on the printed circuit assembly.

Real-Time Clock Recommended Operation

Following a initial system reset from a power-down condition of V_{DD} and V_{DD}_RTC, the counter values of the RTC are undefined and all alarms are disabled. The following procedure is recommended to initialize the Real-Time Clock:

- Write to RTC_CTRL to set RTC_UNLOCK and disable the RTC counter; this action also clears the clock divider

- 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 RTC registers are accessed via the address and data buses using I/O instructions. The RTC_UNLOCK control bit 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 80.

Table 80. 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. The value in the RTC_MIN 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 81.

Table 81. 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. The value in the RTC_HRS 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 82.

Table 82. 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. The value in the RTC_DOW register is unchanged by a RESET. The current setting of BCD_EN determines whether the value in this register is 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 83.

Table 83. 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. The value in the RTC_DOM 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 84.

Table 84. 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. The RTC_MON register begins counting at 01h. The value in the RTC_MON 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 85.

Table 85. 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. The value in the RTC_YR 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 86.

Table 86. 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. The value in the RTC_CEN 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 87.

Table 87. 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. The value in the RTC_ASEC 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). See Table 88.

Table 88. 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. The value in the RTC_AMIN 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). See Table 89.

Table 89. 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. The value in the RTC_AHRS 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). See Table 90.

Table 90. 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. The value in the RTC_ADOW register is unchanged by a RESET. The current setting of BCD_EN determines whether the value in this register is binary (BCD_EN = 0) or binary-coded decimal (BCD_EN = 1). See Table 91.

Table 91. 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 control bits for the Real-Time Clock. The RTC_ACTRL register is cleared by a RESET. See Table 92.

Table 92. Real-Time Clock Alarm Control Register (RTC_ACTRL = 00ECh)

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

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

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

Real-Time Clock Control Register

This register contains control and status bits for the Real-Time Clock. Some bits in the RTC_CTRL register are cleared by a RESET. The ALARM bit flag and associated interrupt (if INT_EN is enabled) are cleared by reading this register. The ALARM bit flag is updated by clearing (locking) the RTC_UNLOCK bit 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 is 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 the BCD_EN bit causes the RTC to use binary-coded decimal (BCD) counting in all registers including the alarm set points.

The CLK_SEL and FREQ_SEL bits 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 the FREQ_SEL bit, and the 32 kHz oscillator is disabled. See Table 93.

Table 93. 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/W	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 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 32768Hz oscillator is disabled.
3 FREQ_SEL	0	Power-line frequency is 60 Hz.
	1	Power-line frequency is 50 Hz.
2 DAY_SAV	0	Suggested value for Daylight Savings Time not selected.
	1	Suggested value for Daylight Savings Time selected. This register bit has been allocated as a storage location only for software applications that use DST. No action is performed in the eZ80F91 when setting or clearing this bit.
1 SLP_WAKE	0	RTC did not generate a sleep-mode recovery reset.
	1	RTC Alarm generated 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 the asynchronous communications protocol. The module also implements two separate 16-byte-deep FIFOs for both transmission and reception. A block diagram of the UART is displayed in Figure 36.

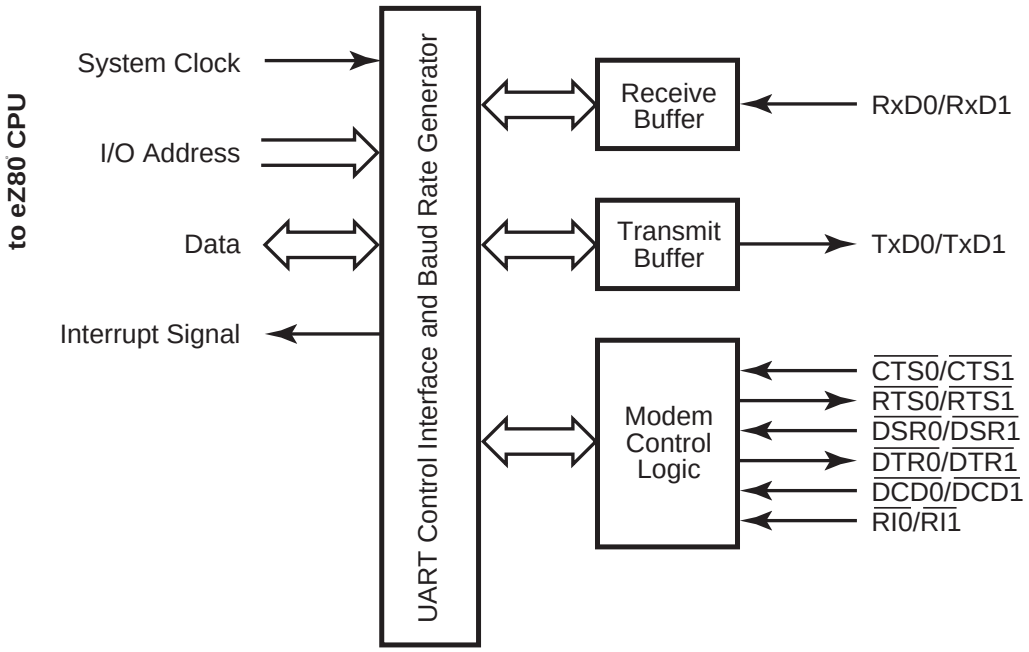


Figure 36. UART Block Diagram

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

- 5-, 6-, 7-, 8- or 9-bit data transmission.
- Even/odd, space/mark, address/data, 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 Baud Rate Generator (BRG) creates the clock for the serial transmit and receive functions. The UART module supports all of the various options in the asynchronous transmission and reception protocol including:

- 5- to 9-bit transmit/receive
- Start bit generation and detection
- Parity generation and detection
- Stop bit generation and detection
- Break generation and detection

The UART contains 16-byte-deep FIFOs in each direction. The FIFOs are enabled or disabled by the application. The receive FIFO features trigger-level detection logic, which enables the CPU to block-transfer data bytes from the receive FIFO.

UART Functions

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, access via the UARTx_THR register, the transmit shift register, the parity generator, and control logic for the transmitter to control parameters for the asynchronous communications protocol.

The UARTx_THR is a Write Only register. The CPU writes the data byte to be transmitted into this register. In FIFO mode, up to 16 data bytes are 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 via 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. An interrupt is sent to the CPU if interrupts are enabled. The CPU resets 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 LSb of the byte to be transmitted is shifted out first and the MSb is shifted out last. The control logic within the block adds the asynchronous communications 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. When enabled, an interrupt is generated after the final protocol bit is transmitted which the CPU resets by loading data into the UARTx_THR register. The TxD output is set to 1 if the transmitter is idle (that is, the transmitter does not contain any data to be transmitted).

The transmitter operates with the 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. For 9-bit data, the host CPU programs the parity bit generator so that it marks the byte as either address (mark parity) or data (space parity).

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 CPU 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 CPU reads all of the received data bytes. If the number of bits received is less than eight, the unused MSb of the data byte Read are 0.

For 9-bit data, the receiver checks incoming bytes for space parity. A line status interrupt is generated when an address byte is received, because address bytes maintain high parity bits. The CPU clears the interrupt by determining if the address matches its own, then configures the receiver to either accept the subsequent data bytes if the address matches, or ignore the data if the address does not match.

The receiver uses the clock from the BRG for receiving the data. This clock must operate at 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 $\overline{RI}$, is detected and an interrupt is generated. For $\overline{RI}$, an interrupt is generated only when the trailing edge of the $\overline{RI}$ is detected. The module also provides LOOP mode for self-diagnostics.

UART Interrupts

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

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

UART Transmitter Interrupt

A Transmitter Hold Register Empty interrupt is generated if there is no data available in the hold register. By the same token, a transmission complete interrupt is generated after the data in the shift register is sent. Both interrupts are disabled using individual interrupt enable bits, or cleared by writing data into the UARTx_THR register.

UART Receiver Interrupts

A receiver interrupt is generated by three possible events. The first event, a receiver data ready interrupt event, indicates that one or more data bytes are received and are ready to be read. Next, 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 results from:

- Incorrect received parity.

Note: *For 9-bit data, incorrect parity indicates detection of an address byte.*

- Incorrect framing (that is, the stop bit) is not detected by receiver at the end of the byte.
- Receiver overrun 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 case of 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 CPU reads the UARTx_MSR register.

UART Recommended Usage

The following standard sequence of events occurs in the UART block of the eZ80F91 device. A description of each follows.

- Module Reset
- Control Transfers to Configure UART Operation
- 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 to Configure UART Operation

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 to 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 whether the interrupt occurs due to either an empty UARTx_THR register or a

completed transmission. When the application makes this determination, it 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 writes 16 bytes at a time. If not, the application writes one byte at a time. As a result of the first Write, the interrupt is deactivated. The CPU then waits for the next interrupt. When the interrupt is raised by the UART module, the CPU 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 described above.

In RS485 multidrop mode, the first byte of the message is the station address and the rest of the message contains the data for that station. You must set the Even Parity Select (EPS bit 4) and Parity Enable (PEN bit 3) in the UARTx_LCTL before sending the station address. We recommend that in your UART initialization routine set up the UARTx_LCTL register for your data transfer format and set the Parity Enable (PEN bit 3) bit. Each time you want to send a new message you must perform these three steps:

1. Since the UART automatically clears the Even Parity Select (EPS bit 4) bit in the UARTx_LCTL after a byte is sent, before starting a new message you have to wait for the transmitter to go idle. The Transmit Empty (TEMT bit 6) of the UARTx_LSR will be set. If you set the EPS bit of the UARTx_LCTL before the last byte of the previous message is transmitted, the EPS bit will be cleared and the new station address will be sent as data instead of being used as an address.
2. Set the Even Parity Select (EPS bit 4) bit in the UARTx_LCTL register being careful not to alter the other bits in the register sets the address mark. Write station address to the UARTx_THR. The UART will automatically clear the EPS bit after the station address byte is transmitted.
3. Send the rest of the message. Write data to the UART Transmit Holding Register UARTx_THR whenever the Transmit Holding Register Empty (THRE bit 5) in the UARTx_LSR is set.

In multidrop mode, during receiving start address marks, you will see a receive line interrupt (INSTS bits[3:1]) in the IIR register. Read the LSR and check for receive errors only and ignore any parity errors. The parity is only used for address marks in this multidrop mode.

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 discards 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 described 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:

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

Upon RESET, the 16-bit BRG divisor value resets to the smallest allowable value of 0002h. Therefore, the minimum BRG clock divisor ratio is 2. 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 are accessed only 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 Use of the Baud Rate Generator

The following is the normal sequence of operations that must occur after the eZ80F91 is powered on to configure the BRG:

1. Assert and deassert RESET.
2. Set UARTx_LCTL[7] to 1 to enable access of the BRG divisor registers.
3. Program the UARTx_BRG_L and UARTx_BRG_H registers.
4. Clear UARTx_LCTL[7] to 0 to disable access of the BRG divisor registers.

BRG Control Registers

UART Baud Rate Generator Register—Low and High Bytes

The registers hold the Low and High bytes of the 16-bit divisor count loaded by the CPU 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, because the values 0000h and 0001h are invalid and proper operation is not guaranteed at these two values. 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 (UART*x*_LCTL) must be set to 1 to access this register. See Table 94 and Table 95 on page 183. For more information, see UART Line Control Register on page 188.

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 94. UART Baud Rate Generator Register—Low Bytes (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] UART_BRG_L	00h–FFh	These bits represent the Low byte of the 16-bit BRG divider value. The complete BRG divisor value is returned by {UART_BRG_H, UART_BRG_L}.

Table 95. UART Baud Rate Generator Register—High Bytes (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] UART_BRG_H	00h–FFh	These bits represent the High byte of the 16-bit BRG divider value. The complete BRG divisor value is returned by {UART_BRG_H, UART_BRG_L}.

UART Registers

After a system reset, all UART registers are set to their default values. Any Writes to unused registers or register bits are ignored and reads return a value of 0. For compatibility with future revisions, unused bits within a register must 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 Register

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. You 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 96 on page 184.

Table 96. 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 Register

The bits in this register reflect the data received. If less than eight bits are programmed for reception, 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 97.

Table 97. 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 Register

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 98 on page 185.

Table 98. 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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: R/W = Read/Write.

Bit Position	Value	Description
[7:5]	000	Reserved.
4 TCIE	0	Transmission complete interrupt is disabled.
	1	Transmission complete interrupt is generated when both the transmit hold register and the transmit shift register are empty.
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.
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 Register

The Read Only UARTx_IIR register allows you 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 Table 99 and Table 100.

Table 99. 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] FSTS	0	FIFO is disabled.
	1	FIFO is enabled.
[6:4]	000	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 100 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 100. 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
101	Fourth	Transmission Complete
001	Fifth	Transmit Buffer Empty
000	Lowest	Modem Status

UART FIFO Control Register

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 101.

Table 101. 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 4bytes 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]	000b	Reserved—must be 000b.
2 CLRTxF	0	Transmit Disable. This register bit works differently than the standard 16550 UART. This bit must be set to transmit data. When it is reset the transmit FIFO logic is reset along with the associated transmit logic to keep them in sync. This bit is now persistent–it does not self clear and it must remain at 1 to transmit data.
	1	Transmit Enable.
1 CLRRxF	0	Receive Disable. This register bit works differently than the standard 16550 UART. This bit must be set to receive data. When it is reset the receive FIFO logic is reset along with the associated receive logic to keep them in sync and avoid the previous version's lookup problem. This bit is now persistent–it does not self clear and it must remain at 1 to receive data.
	1	Receive Enable.

Bit Position	Value	Description
0 FIFOEN	0	FIFOs are not used.
	1	Receive and transmit FIFOs are used–You must clear the FIFO logic using bits 1 and 2. First enable the FIFOs by setting bit 0 to 1 then enable the receiver and transmitter by setting bits 1 and 2.

UART Line Control Register

This register is used to control the communication control parameters. See Table 102 and Table 103 on page 189.

Table 102. 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 C0h, C1h, D0h and D1h is enabled.
	1	Access to the Baud Rate Generator registers at I/O addresses C0h, C1h, D0h and D1h is enabled.
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 must 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 parity enable bit (pen) are both 1, an incorrect parity bit is transmitted with the data byte.

Bit Position	Value	Description
4 EPS	0	Even Parity Select. Use odd parity for transmit and receive. The total number of 1 bits in the transmit data plus parity bit is odd. Used as SPACE bit in Multidrop Mode. See Table 104 on page 189 for parity select definitions. Note: Receive Parity is set to SPACE in multidrop mode.
	1	Use even parity for transmit and receive. The total number of 1 bits in the transmit data plus parity bit is even. Used as MARK bit in Multidrop Mode. See Table 104 on page 189 for parity select definitions.
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. In Multidrop Mode, receive parity is checked for space parity.
[2:0] CHAR	000–111	UART Character Parameter Selection. See Table 103 on page 189 for a description of the values.

Table 103. 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
100	5	2
101	6	2
110	7	2
111	8	2

Table 104. Parity Select Definition for Multidrop Communications

Multidrop Mode	Even Parity Select	Parity Type
0	0	odd
0	1	even
1	0	space
1	1*	mark

Note: *In Multidrop Mode, EPS resets to 0 after the first character is sent.

UART Modem Control Register

This register is used to control and check the modem status. See Table 105.

Table 105. 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/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	Reserved.
6 POLARITY	0	TxD and RxD signals—Normal Polarity.
	1	Invert Polarity of TxD and RxD signals.
5 MDM	0	Multidrop Mode disabled.
	1	Multidrop Mode enabled. See Table 104 on page 189 for parity select definitions.
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.

Bit Position	Value	Description
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.
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 $\overline{\text{RTS}}$ 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 $\overline{\text{DTR}}$ 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 Register

This register is used to show the status of UART interrupts and registers. See Table 106.

Table 106. 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 in 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. 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. In multidrop mode, this indicates that the received character is a data byte. 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. In multidrop mode, this indicates that the received character is an address byte.
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 Register

This register is used to show the status of the UART signals. See Table 107.

Table 107. 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 $\overline{\text{RI}}$x 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 $\overline{\text{DCD}}$. This bit is set to 1 whenever the $\overline{\text{DCDx}}$ pin changes state. This bit is reset to 0 when the UARTx_MSR register is read.
2 TERI	0–1	Trailing Edge Change on $\overline{\text{RI}}$. This bit is set to 1 whenever a falling edge is detected on the $\overline{\text{RIx}}$ pin. This bit is reset to 0 when the UARTx_MSR register is read.
1 DDSR	0–1	Delta Status Change of $\overline{\text{DSR}}$. This bit is set to 1 whenever the $\overline{\text{DSRx}}$ pin changes state. This bit is reset to 0 when the UARTx_MSR register is read.
0 DCTS	0–1	Delta Status Change of $\overline{\text{CTS}}$. This bit is set to 1 whenever the $\overline{\text{CTSx}}$ pin changes state. This bit is reset to 0 when the UARTx_MSRs register is read.

UART Scratch Pad Register

The UARTx_SPR register is used by the system as a general-purpose Read/Write register. See Table 108.

Table 108. 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. In multi-drop 9 bit mode, this register is used to store the address value.

Infrared Encoder/Decoder

The eZ80F91 device contains a UART to an infrared encoder/decoder (endec). The endec is integrated with the on-chip UART0 to allow easy communication between the CPU and IrDA Physical Layer Specification Version 1.4-compatible infrared transceivers, as displayed in Figure 37. Infrared communication provides secure, reliable, high-speed, low-cost, point-to-point communication between PCs, PDAs, mobile telephones, printers, and other infrared-enabled devices.

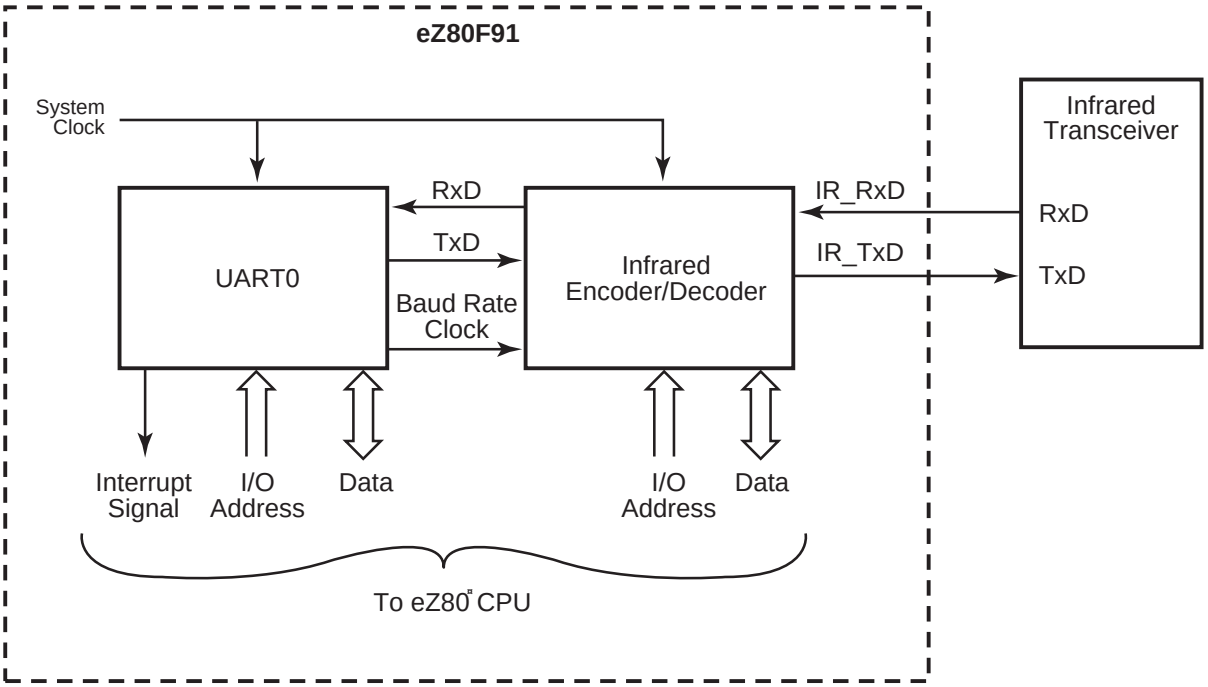


Figure 37. Infrared System Block Diagram

Functional Description

When the endec 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 endec 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 (BRG), which supports IrDA standard baud rates from 9600 bps to 115.2 kbps. Higher baud rates are possible, but do not meet

IrDA specifications. The UART must be enabled to use the endec. For more information on the UART and its BRG, see Universal Asynchronous Receiver/Transmitter on page 175.

Transmit

The data to be transmitted via the IR transceiver is the data sent to UART0. The UART transmit signal, TxD, and Baud Rate Clock are used by the endec 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 displayed in Figure 38. During data transmission, the IR receive function must be disabled by clearing the IR_RxEN bit in the IR_CTL reg to 0 to prevent transmitter-to-receiver crosstalk.

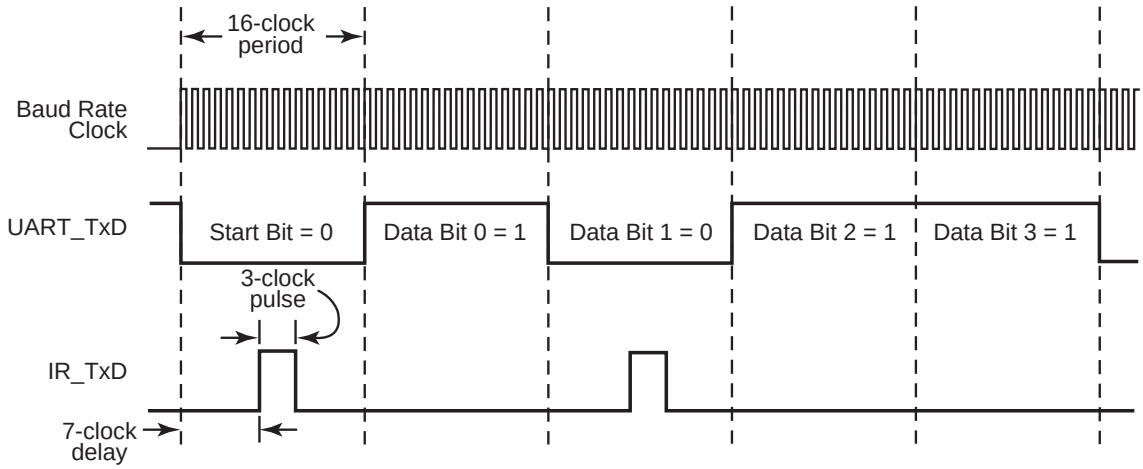


Figure 38. Infrared Data Transmission

Receive

Data received from the IR transceiver via the IR_RxD signal is decoded by the endec and passed to the UART. The IR_RxEN bit in the IR_CTL register must be set to enable the receiver decoder. The IrDA serial infrared (SIR) data format uses half duplex communication. Therefore, the UART must not be allowed to transmit while the receiver decoder is enabled. The UART Baud Rate Clock is used by the endec 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 delayed Low (0) pulse is output on RxD. Data transmission is displayed in Figure 39.

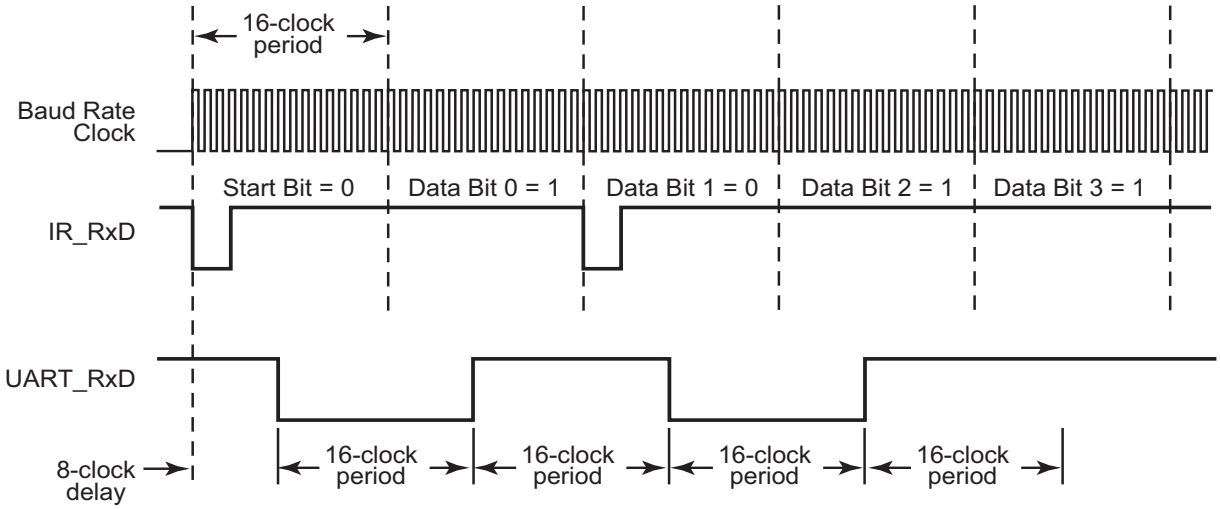


Figure 39. Infrared Data Reception

The IrDA endec is designed to ignore pulses on IR_RxD which do not comply with IrDA pulse width specifications. Input pulses wider than five baud clocks (that is, 5/16 of a bit period) are always ignored, as this would be a violation of the maximum pulse width specified for any standard baud rate up to 115.2 kbps. The check for minimum pulse widths is optional, since using a slow system clock frequency limits the ability to accurately measure narrow pulse widths near the IrDA specification minimum of 1.41 us for the 2.4–115.2 kbps rate range.

To enable checks of minimum input pulse width on IR_RxD, a non-zero value must be programmed into the MIN_PULSE field of IR_CTL (bits [7:4]). This field forms the most-significant four bits of the 6-bit down-counter used to determine if an input pulse will be ignored because it is too narrow. The lower two counter bits are hard-coded to load with 0x3, resulting in a total down-count equal to ((MIN_PULSE* 4) + 3). To be accepted, input pulses must have a width greater than or equal to the down-count value times the system clock period.

The following equation is used to determine an appropriate setting for MIN_PULSE:

$MIN\_PULSE = INT(((F_{sys} * W_{min}) - 3) / 4)$

Where,

F_{sys} is the frequency of the system clock, and,

W_{min} is the minimum width of recognized input pulses.

If this equation results in a value less than one, MIN_PULSE must be set to 0x0h which enables edge detection and ensures that valid pulses wider than W_{min} are accepted. The field's maximum setting of 0xFh supports a W_{min} of 1.25 us when F_{sys} is 50 MHz.

Jitter

Due to the inherent sampling of the received IR_RxD signal by the Bit Rate Clock, some jitter is 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 endec signal pins, IR_TxD and IR_RxD, are multiplexed with General-Purpose Input/Output (GPIO) pins. These GPIO pins must be configured for alternate function operation for the endec to operate.

The remaining six UART0 pins, $\overline{CTS0}$, $\overline{DCD0}$, $\overline{DSR0}$, $\overline{DTR0}$, $\overline{RTS}$, and $\overline{RI0}$, are not required for use with the endec. The UART0 modem status interrupt must be disabled to prevent unwanted interrupts from these pins. The GPIO pins corresponding to these six unused UART0 pins are used for inputs, outputs, or interrupt sources. Recommended GPIO Port D control register settings are listed in Table 109. See General-Purpose Input/Output on page 49 for additional information on setting the GPIO Port modes.

Table 109. 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 is accomplished with the endec on the eZ80F91 device. Internal loopback testing is enabled by setting the LOOP_BACK bit to 1. During internal loopback, the IR_TxD output signal is inverted and connected on-chip to the IR_RxD input. External loopback testing of the off-chip IrDA transceiver is 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. The IR_CTL register is listed in Table 110.

Table 110. Infrared Encoder/Decoder Control Registers (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	R/W	R/W	R/W

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

Bit Position	Value	Description
[7:4] MIN_PULSE	0000	Minimum receive pulse width control. When this field is equal to 0x0, the IrDA decoder uses edge detection to accept arbitrarily narrow (that is, short) input pulses.
	1h-Fh	When not equal to 0x0, this field forms the most-significant four bits of the 6-bit down-counter used to determine if an input pulse will be ignored because it is too narrow. The lower two counter bits are hard-coded to load with 0x3, resulting in a total down-count equal to ((IR_CTL[4:0]MIN_PULSE * 4) + 3). To be accepted, input pulses must have a width greater than or equal to the down-count value times the system clock period.
3	0	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	Endec is disabled.
	1	Endec 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 is configured either as a master or as a slave. The connection of two SPI devices (one master and one slave) and the direction of data transfer is displayed in Figure 40 and Figure 41.

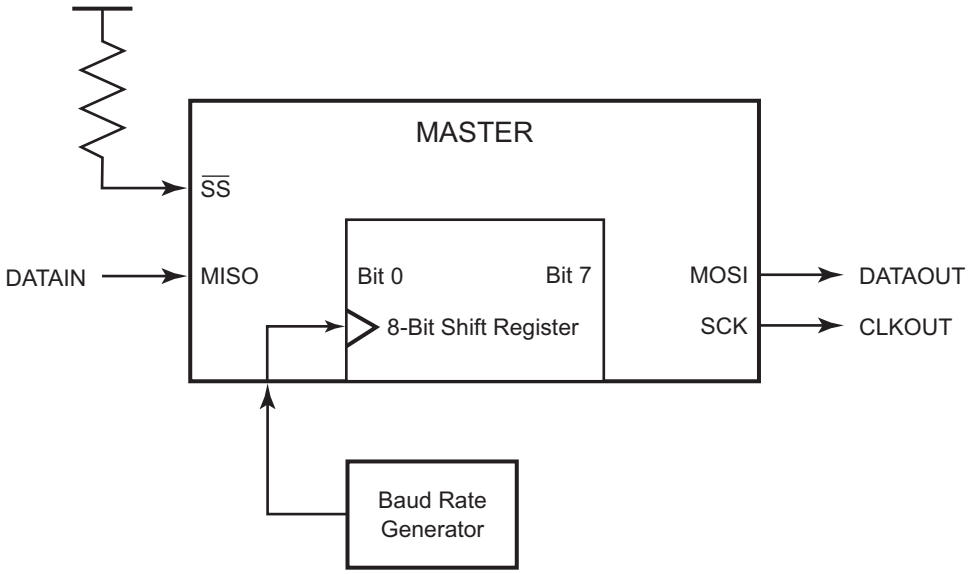


Figure 40. SPI Master Device

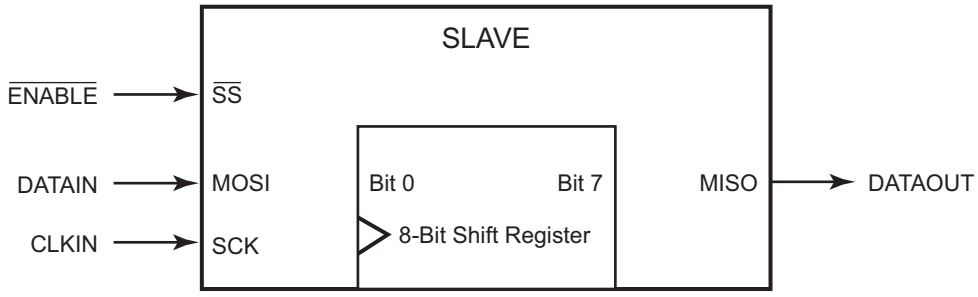


Figure 41. SPI Slave Device

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 (msb) 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 msb 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 $\overline{SS}$ signal goes Low in Master mode, a Mode Fault error flag (MODF) is set in the SPI_SR register. For more information, see SPI Status Register on page 209.

When the clock phase (CPHA) is set to 0, the shift clock is the logical OR of $\overline{SS}$ with SCK. In this clock phase mode, $\overline{SS}$ must go High between successive characters in an SPI message. When CPHA is set to 1, $\overline{SS}$ remains Low for several SPI characters. In cases where there is only one SPI slave, its $\overline{SS}$ line could be tied Low as long as CPHA is set to 1. For more information on CPHA, see SPI Control Register on page 208.

Serial Clock

The Serial Clock (SCK) is used to synchronize data movement both in and out of the device via 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 displayed in Figure 42 and Table 111, four possible timing relations are chosen by using the clock polarity (CPOL) and clock phase CPHA control bits in the SPI Control register. See SPI Control Register on page 208. Both the master and slave must operate with the identical timing, CPOL, and CPHA. The master device always places data on the MOSI line a half-cycle before the clock edge (SCK signal), for the slave device to latch the data.

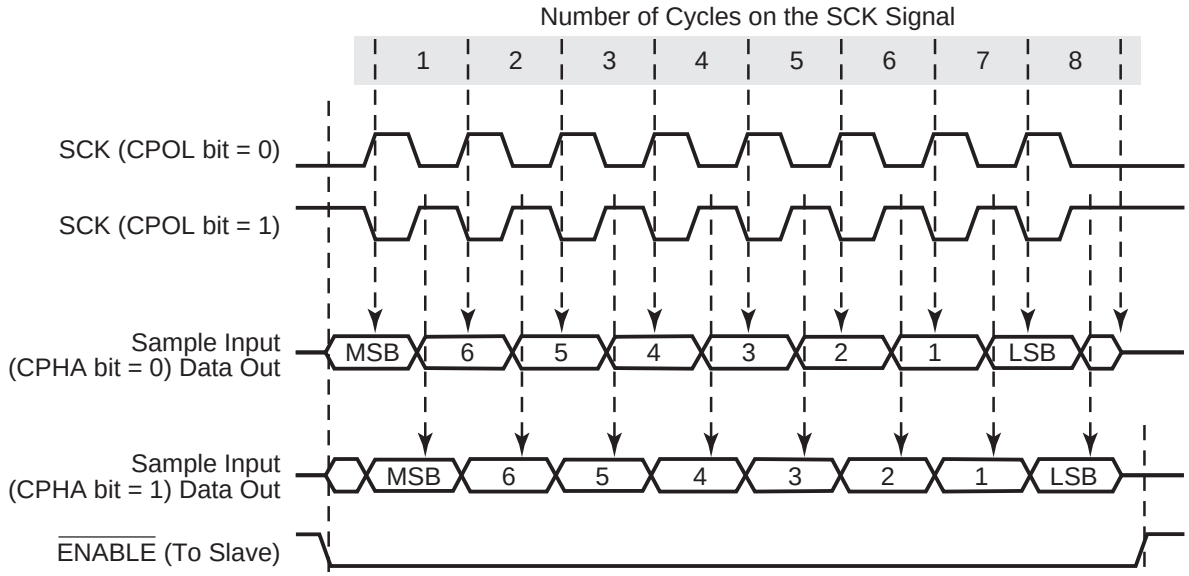


Figure 42. SPI Timing

Table 111. SPI Clock Phase and Clock Polarity Operation

CPHA	CPOL	SCK Transmit Edge	SCK Receive Edge	SCK Idle State	$\overline{SS}$ 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 result is a full-duplex transmission, with both *data out* and *data in* synchronized with the same clock signal. The byte transmitted is replaced by the byte received, eliminating the need for separate transmit-empty and receive-full status bits. A single status bit, SPIF, is used to signify that the I/O operation is complete. See SPI Status Register on page 209.

The SPI is double-buffered during reads, but not during Writes. 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 SPI flag of the SPI_SR register is set to 1.

In SPI MASTER mode, the SCK pin functions as 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 via the MOSI pin while shifting in eight bits of data via the MISO pin. After transfer, the SCK signal becomes idle.

In SPI SLAVE mode, the start logic receives a logic Low from the $\overline{SS}$ pin and a clock input at the SCK pin; as a result, the slave is synchronized to the master. Data from the master is received serially from the slave MOSI signal and is loaded into 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. Next, 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 the $\overline{SS}$ pin signal goes Low. 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. The transfer ends when the SPI flag is set to 1.

SPI Flags

Mode Fault

The Mode Fault flag (MODF) indicates that there is a multimaster conflict in the 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 sequence 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 to 0.

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 to 1.

SPI Baud Rate Generator

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

Baud Rate Generator Functional Description

The SPI's BRG consists of a 16-bit downcounter, two 8-bit registers, and associated decoding logic. The BRG'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 the count.

The SPI Data Rate is calculated using the following equation:

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

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 a Master

The following list describes the procedure for transferring data from a master SPI device to a slave SPI device.

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

Data Transfer Procedure with SPI Configured as a Slave

The following list describes the procedure for transferring data from a slave SPI device to a master SPI device.

1. Load the SPI BRG 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 $\overline{SS}$.

SPI Registers

There are six registers in the Serial Peripheral Interface that 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 CPU 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 a restart of the count. See Table 112 on page 207 and Table 113 on page 207.

Table 112. 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 BRG divider value. The complete BRG divisor value is returned by {SPI_BRG_H, SPI_BRG_L}.

Table 113. 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 BRG 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 must be disabled prior to making any changes to CPHA or CPOL. See Table 114.

Table 114. 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	$\overline{SS}$ must go High after transfer of every byte of data.
	1	$\overline{SS}$ remains 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 115.

Table 115. 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 over 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 SPI Flag (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 116 on page 210.

Table 116. 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; otherwise, an overrun condition exists. In the event of an overrun, the byte that causes 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 117.

Table 117. 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 Inter-Integrated Circuit (I²C) serial I/O bus is a two-wire communication interface that operates in four modes:

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

The I²C interface consists of a Serial Clock (SCL) and Serial Data (SDA). Both SCL and SDA 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 are 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.

The Low period of the clock is stretched by a slave to slow down the bus master. The Low period is also stretched for handshaking purposes. This result is accomplished 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 to 0.

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 (NACK) bit, the I²C returns to an idle state. If arbitration is lost during the transmission of an address, the I²C switches to SLAVE mode so that it recognizes 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 changes only when the clock signal on the SCL line is Low, as displayed in Figure 43.

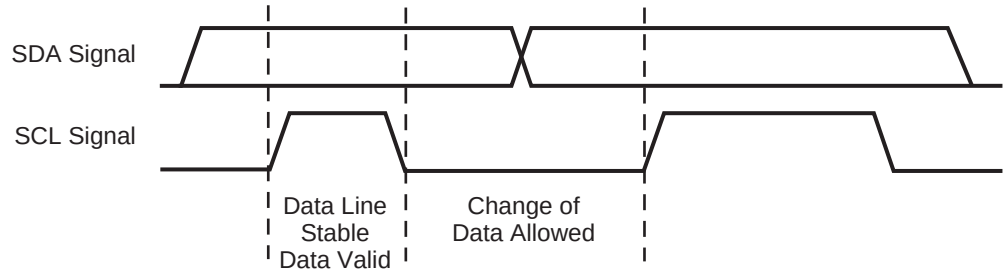


Figure 43. I²C Clock and Data Relationship

START and STOP Conditions

Within the I²C bus protocol, unique situations arise which are defined as START and STOP conditions. Figure 44 displays a High-to-Low transition on the SDA line while SCL is High, indicating 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 a START condition. The bus is considered to be free for a defined time after a STOP condition.

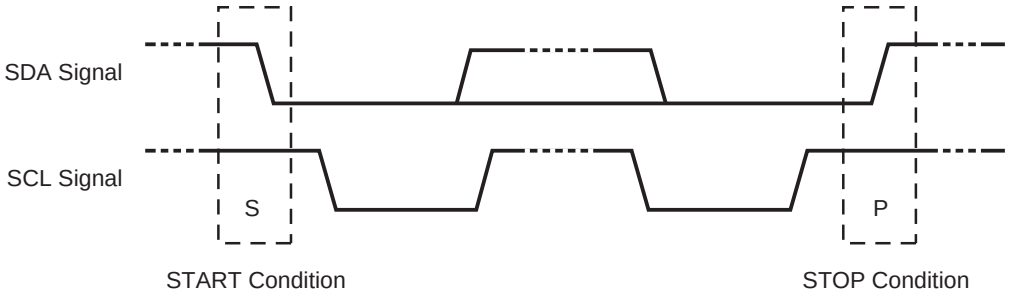


Figure 44. START and STOP Conditions In I²C Protocol

Transferring Data

Byte Format

Every character transferred on the SDA line must be a single 8-bit byte. The number of bytes that is transmitted per transfer is unrestricted. Each byte must be followed by an Acknowledge (ACK). Data is transferred with the most-significant bit (msb) first. Figure 45 displays a receiver that holds 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.

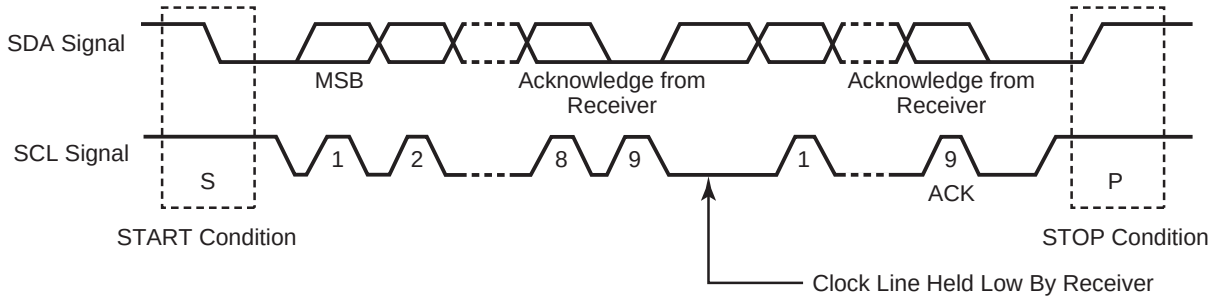


Figure 45. I²C Frame Structure

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 46 on page 214.

A receiver that is addressed is obliged to generate an ACK after each byte is received. When a slave receiver does not acknowledge the slave address (for example, unable to receive because it is 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.

If a master receiver is involved in a transfer, it must signal the end of the data stream 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.

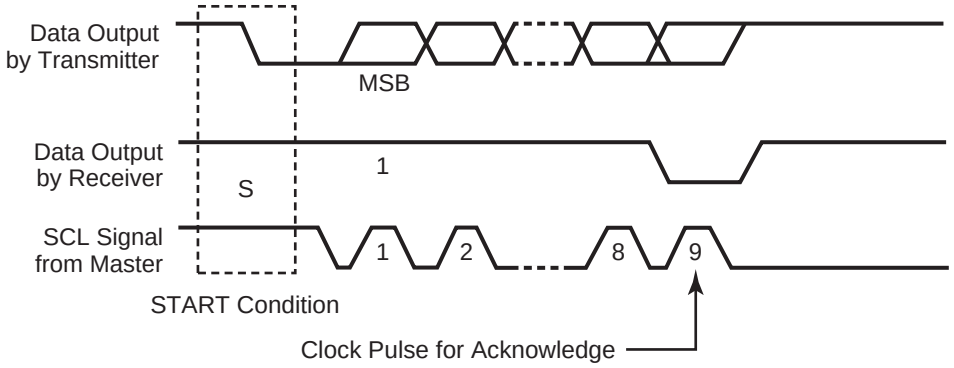


Figure 46. I²C Acknowledge

Clock Synchronization

All masters generate their own clocks on the SCL line to transfer messages on the I²C bus. Data is only valid during the High period of each clock.

Clock synchronization is performed using the wired AND connection of the I²C 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 47 on page 215. The Low-to-High transition of this clock, however, cannot 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 count off the 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; all of the devices start counting the 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 device with the shortest clock High period.

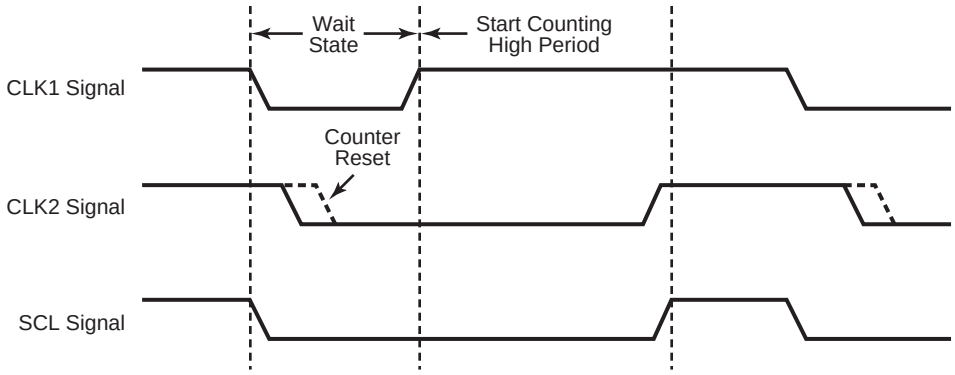


Figure 47. Clock Synchronization In I²C Protocol

Arbitration

Any master initiates a transfer if the bus is free. As a result, multiple masters each generates a START condition if the bus is free within a minimum period. If multiple masters generate a START condition, a START is defined for the bus. However, arbitration defines which MASTER controls the bus. Arbitration takes place on the SDA line. As mentioned, START conditions are initiated only while the SCL line is held High. If during this period, a master (M1) initiates a High-to-Low transition—that is, a START condition—while a second master (M2) transmits a Low signal on the line, then the first master, M1, cannot take control of the bus. As a result, the data output stage for M1 is disabled.

Arbitration continues 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 a comparison of the data. Because address and data information on the I²C bus is used for arbitration, no information is lost during this process. A master that loses the arbitration generates 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 is possible that the winning master is trying to address it. The losing master must switch over immediately to its slave receiver mode. Figure 47 displays the arbitration procedure for two masters. Of course, more masters can 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 I²C 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 I²C 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 functions as a handshake, enabling receivers to cope with fast data transfers, on either a byte or a 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^2C 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^2C then tests the I^2C 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 must 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 118 on page 217.

Table 118. I²C Master Transmit Status Codes

Code	I²C State	Microcontroller Response	Next I²C Action
18h	Addr+W transmitted ACK received[1]	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 address received, ACK transmitted	Same as code 68h	Same as code 68h
B0h	Arbitration lost, SLA+R received, ACK transmitted[3]	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; that is, the lsb is cleared to 0.
2. AAK is an I²C control bit that identifies which ACK signal to transmit.
3. R is defined as the Read bit; that is, the lsb is set to 1.

If 10-bit addressing is used, 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 listed in Table 119.

Table 119. I²C 10-Bit Master Transmit Status Codes

Code	I²C State	Microcontroller Response	Next I²C 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[1]	Clear IFLG, clear AAK = 0[2]	Receive data byte, transmit NACK
		Or clear IFLG, set AAK = 1	Receive data byte, transmit ACK
B0h	Arbitration lost, SLA+R received, ACK transmitted[3]	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

Notes
1. W is defined as the Write bit; that is, the lsb is cleared to 0.
2. AAK is an I²C control bit that identifies which ACK signal to transmit.
3. R is defined as the Read bit; that is, the lsb is set to 1.

If a repeated START condition is transmitted, the status code is `10h` instead of `08h`. After each data byte is transmitted, the IFLG is set to 1 and one of the status codes listed in Table 120 is loaded into the I2C_SR register.

Table 120. I²C Master Transmit Status Codes For Data Bytes

Code	I²C State	Microcontroller 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 microcontroller must 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 an 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 into the I2C_SR register. The I2C_DR register must 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 must 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 121 on page 220 is loaded into the I2C_SR register.

Table 121. I²C Master Receive Status Codes

Code	I²C State	Microcontroller Response	Next I²C Action
40h	Addr + R transmitted, ACK received	For a 7-bit address, clear IFLG, AAK = 0[1]	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[2]	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[3]	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

Notes
1. AAK is an I²C control bit that identifies which ACK signal to transmit.
2. R is defined as the Read bit; that is, the lsb is set to 1.
3. W is defined as the Write bit; that is, the lsb is cleared to 0.

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, this time 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 to 1 and one of the status codes listed in Table 122 is loaded into the I2C_SR register.

Table 122. I2C Master Receive Status Codes For Data Bytes

Code	I2C State	Microcontroller Response	Next I2C 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

Note: AAK is an I2C control bit that identifies which ACK signal to transmit.

When all bytes are received, a NACK must be sent, then the microcontroller must write 1 to the STP bit in the I2C_CTL register. The I2C then transmits a STOP condition, clears the STP bit and returns to an idle state.

Slave Transmit

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

The I2C enters SLAVE TRANSMIT mode when it receives its own slave address and a Read bit after a START condition. The I2C then transmits an ACK bit (if the AAK bit is set to 1); it then sets the IFLG bit in the I2C_CTL register. As a result, the I2C_SR register contains the status code A8h.

Note: *When I2C contains a 10-bit slave address (signified by the address range F0h–F7h in the I2C_SAR register), it transmits an ACK when the first address byte is received after a restart. An interrupt is generated and IFLG is set to 1; however, 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 is cleared to 0. After the I²C transmits the byte and receives an ACK, the IFLG bit is set to 1 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 to 0. After the final byte is transmitted, the IFLG is set and the I²C_SR register contains C8h and the I²C returns to an idle state. The AAK bit must be set to 1 before reentering SLAVE mode.

If no ACK is received after transmitting a byte, the IFLG is set and the I²C_SR register contains C0h. The I²C then returns to an idle state. If a STOP condition is detected after an ACK bit, the I²C returns to an 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 ACK 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 ACK bit (Low level on SDA) is transmitted and the IFLG bit is set after each byte is received. The I²C_SR register contains the two status codes 80h or 90h if SLAVE RECEIVE mode is entered with the general call address. The received data byte are 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 I²C transmits a NACK bit (High level on SDA) after the next byte is received, and sets the IFLG bit to 1. The I²C_SR register

contains the two status codes 88h or 98h if SLAVE RECEIVE mode is entered with the general call address. The I2C returns to an idle state when the IFLG bit is cleared to 0.

I2C Registers

The section that follows describes each of the eZ80F91 MCU's Inter-Integrated Circuit (I2C) registers.

Addressing

The CPU interface provides access to seven 8-bit registers: four Read/Write registers, one Read Only register and two Write Only registers, as listed in Table 123.

Table 123. I2C 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 I2C Registers

Hardware Reset—When the I2C is reset by a hardware reset of the eZ80F91 device, 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 I2C Software Reset Register (I2C_SRR). A software reset clears the STP, STA, and IFLG bits of the I2C_CTL register to 0 and sets the I2C back to an idle state.

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 (I^2C_XSAR) is received, the I^2C generates an interrupt and enters SLAVE mode.Then I^2C_SAR[2:1] are used as the upper 2 bits for the 10-bit extended address. The full 10-bit address is supplied by {I^2C_SAR[2:1], I^2C_XSAR[7:0]}. See Table 124.

Table 124. I^2C 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, I^2C_SAR[2:1], of address when operating in 10-bit mode.
0 GCE	0	I^2C not enabled to recognize the General Call Address.
	1	I^2C enabled to recognize the General Call Address.

I^2C Extended Slave Address Register

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

When the register receives an address starting with F7h to F0h (I^2C_SAR[7:3] = 11110b), the I^2C recognizes that a 10-bit slave addressing mode is being selected. The I^2C sends an ACK after receiving the I^2C_XSAR byte (the device does not generate an interrupt at this point). After the next byte of the address (I^2C_XSAR) is received, the I^2C generates an interrupt and enters SLAVE mode.Then I^2C_SAR[2:1] are used as the upper 2 bits for the 10-bit extended address. The full 10-bit address is supplied by {I^2C_SAR[2:1], I^2C_XSAR[7:0]}. See Table 125.

Table 125. I^2C 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 MSb of the byte is transmitted first. In RECEIVE mode, the first bit received is placed in the MSb 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 126.

Table 126. 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 I²C bus inputs SCLx and SDAx are ignored and the I²C module does not respond to any address on the bus. When ENAB is set to 1, the I²C 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 I²C 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 I²C 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 I²C block is being accessed in SLAVE mode, the I²C 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 0 to the STA 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 mode, 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), then transmits the START condition. The STP bit is cleared to 0 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 I^2C_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 SLAVE TRANSMIT mode, the byte in the I^2C_DR register is assumed to be the final byte. After this byte is transmitted, the I^2C block enters the C8h state, then returns to an idle state. The I^2C module does not respond to its slave address unless AAK is set to 1. See Table 127 on page 226.

Table 127. I^2C Control Register (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	I^2C interrupt is disabled.
	1	I^2C interrupt is enabled.
6 ENAB	0	The I^2C bus (SCL/SDA) is disabled and all inputs are ignored.
	1	The I^2C bus (SCL/SDA) is enabled.

Bit Position	Value	Description
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	I^2C interrupt flag is not set.
	1	I^2C interrupt flag is set.
2 AAK	0	Not Acknowledge.
	1	Acknowledge.
[1:0]	00	Reserved.

I^2C Status Register

The I^2C_SR register is a Read Only register that contains a 5-bit status code in the five MSbs; the three LSbs are always 0. The Read Only I2C_SR registers share the same I/O addresses as the Write Only I2C_CCR registers. See Table 128.

Table 128. I^2C 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 I^2C status code.
[2:0]	000	Reserved.

There are 29 possible status codes, as listed in Table 129. When the I^2C_SR register contains the status code F8h, no relevant status information is available, no interrupt is generated, and the IFLG bit in the I^2C_CTL register is not set. All other status codes correspond to a defined state of the I^2C.

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 to 1. When the IFLG bit is cleared, the status code returns to F8h.

Table 129. 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.
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.

Table 129. I²C Status Codes (Continued)

Code	Status
D8h	Second Address byte and Write bit transmitted, ACK not received.
F8h	No relevant status information, IFLG = 0.

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

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

I²C Clock Control Register

The I²C_CCR register is a Write Only register. The seven LSBs control the frequency at which the I²C bus is sampled and the frequency of the I²C clock line (SCL) when the I²C is in MASTER mode. The Write Only I²C_CCR registers share the same I/O addresses as the Read Only I2C_SR registers. See Table 130.

Table 130. I²C Clock Control Registers (I2C_CCR = 00CCh)

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

Note: W = Read only.

Bit Position	Value	Description
7	0	Reserved.
[6:3] M	0000–1111	I²C clock divider scalar value.
[2:0] N	000–111	I²C clock divider exponent.

The I²C clocks are derived from the system clock of the eZ80F91 device. The frequency of this system clock is f_{SCK}. The I²C bus is sampled by the I²C block at the frequency f_{SAMP} supplied by the following equation:

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

In MASTER mode, the I²C clock output frequency on SCL (f_{SCL}) is supplied by the following equation:

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

The use of two separately-programmable dividers allows the MASTER mode output frequency to be set independently of the frequency at which the I²C bus is sampled. This feature is particularly useful in multimaster systems because the frequency at which the I²C 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 is ensured while allowing the MASTER mode output to be set to a lower frequency.

Bus Clock Speed

The I²C 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 I²C must sample the I²C bus at least ten times faster than the bus clock speed of the fastest master on the bus. The sampling frequency must therefore be at least 1 MHz (4 MHz in FAST mode) to guarantee correct operation with other bus masters.

The I²C sampling frequency is determined by the frequency of the eZ80F91 system clock and the value in the I²C_CCR bits 2 to 0. The bus clock speed generated by the I²C 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].

I²C Software Reset Register

The I²C_SRR register is a Write Only register. Writing any value to this register performs a software reset of the I²C module. See Table 131.

Table 131. I²C 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 I²C module.

Zilog Debug Interface

Introduction

The Zilog Debug Interface (ZDI) provides a built-in debugging interface to the 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 II (ZDS II).

The above features are built into the silicon. Control is provided via a two-wire interface that is connected to the USB Smart Cable emulator. Figure 48 displays a typical setup using a a target board, USB Smart Cable, and the host PC running Zilog Developer Studio II. For more information on USB Smart Cable and ZDS II, refer to www.zilog.com.

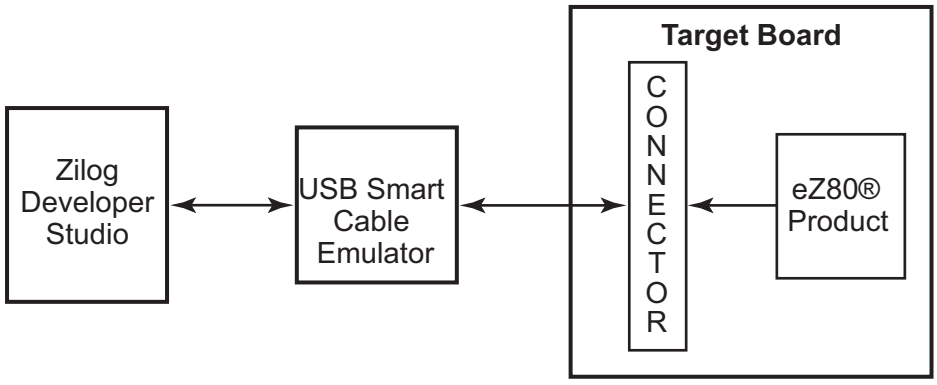


Figure 48. Typical ZDI Debug Setup

ZDI allows reading and writing of most internal registers without disturbing the state of the machine. Reads and Writes to memory occurs as fast as the ZDI downloads and uploads data, with a maximum supported ZDI clock frequency of 0.4 times the eZ80F91 system clock frequency. Also, regardless of the ZDI clock frequency, the duration of the low-phase of the ZDI clock (that is, ZCL = 0) must be at least 1.25 times the system clock period.

For the description on how to enable the ZDI interface on the exit of RESET, see the OCI Activation on page 258.

Table 132. Recommend ZDI Clock versus 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 eZ80F91 device is considered a slave in all data transfers.

Figure 49 on page 233 displays the schematic for building a connector on a target board. This connector allows you to connect directly to the USB Smart Cable emulator using a six-pin header.

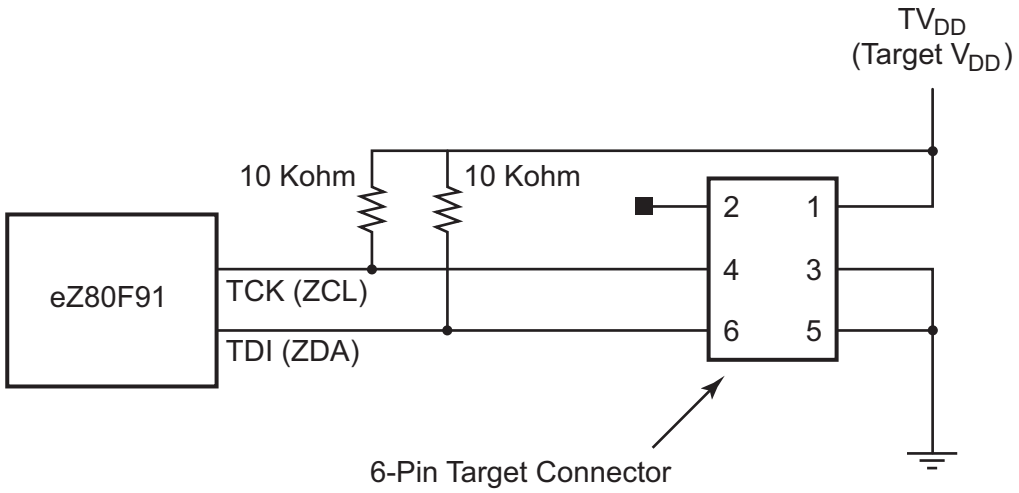


Figure 49. Schematic For Building a Target Board USB Smart Cable Connector

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 eZ80F91, 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 changes 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 MSb (bit 7) of each byte being first in time, and the LSb (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 eZ80F91 device 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. Figure 50 on page 234 and Figure 51 on page 234 displays 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 displayed in Figure 50. Data is shifted out during a Read from the ZDI block on the falling edge of ZCL as displayed in Figure 51. When an operation is completed, the master stops during the ninth cycle and holds the ZCL signal High.

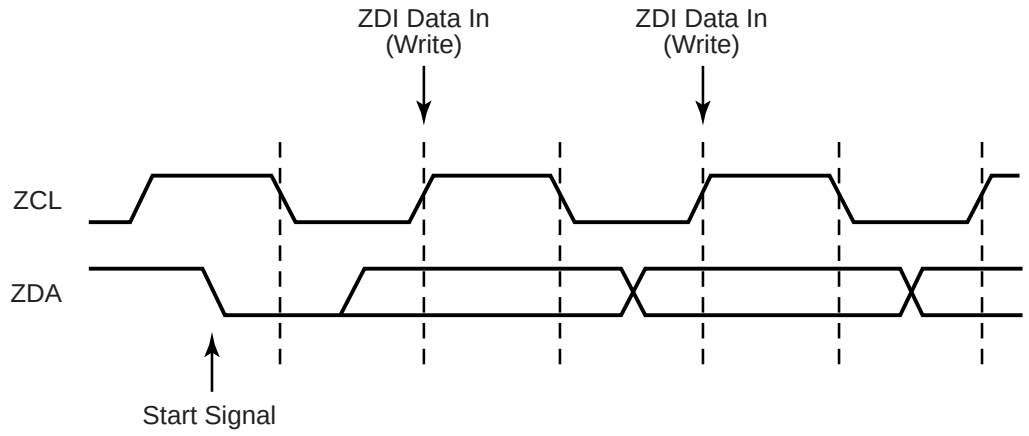


Figure 50. ZDI Write Timing

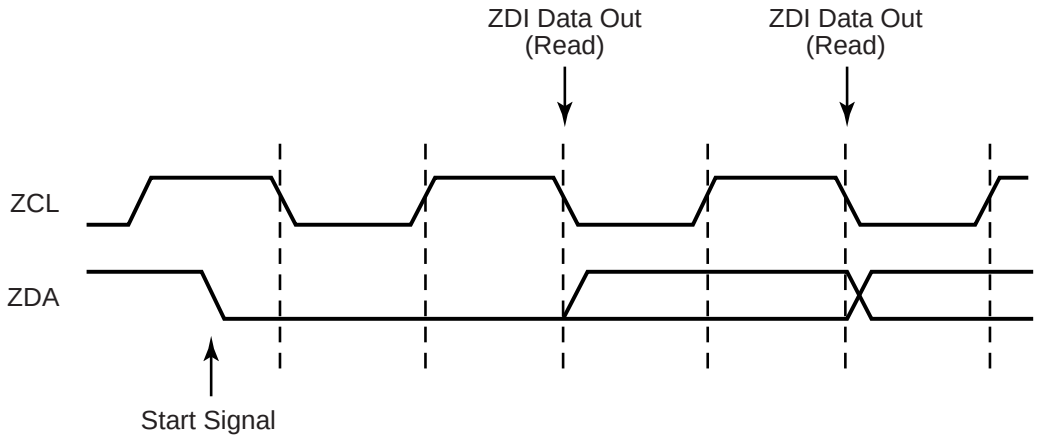


Figure 51. ZDI Read Timing

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 is 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 must 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, see Bus Requests During ZDI Debug Mode on page 238.

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 must not be confused with the eZ80F91 device 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. For 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/$\overline{W}$ bit (0 = Write, 1 = Read). If no new START command is issued at completion of the Read or Write operation, the operation is 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 52 displays the timing for address Writes to ZDI registers.

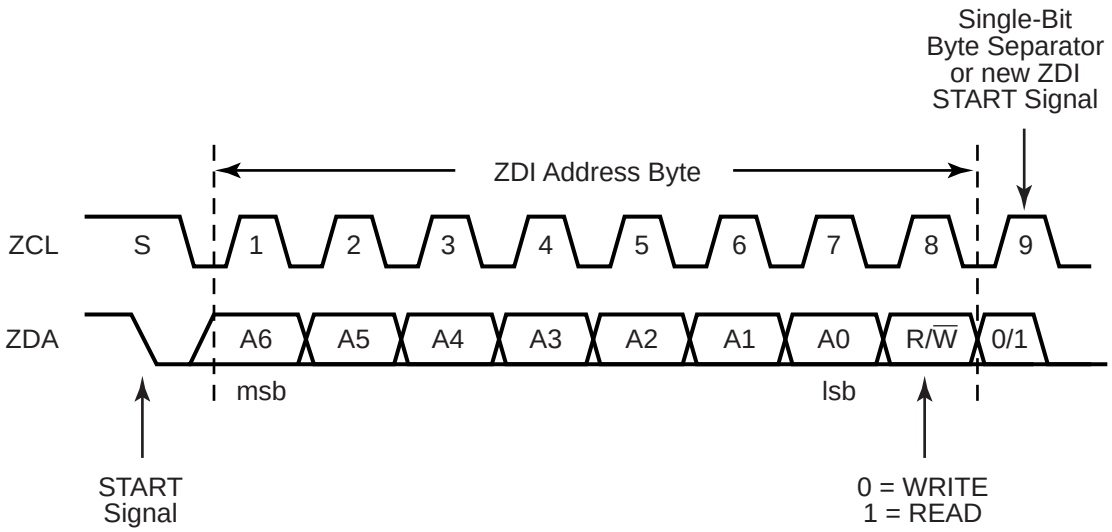


Figure 52. ZDI Address Write Timing

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 53 displays the timing for ZDI single-byte Write operations.

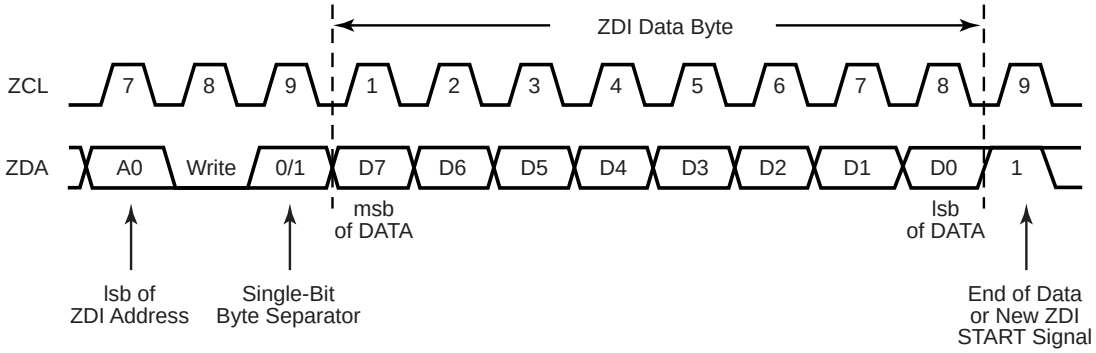


Figure 53. ZDI Single-Byte Data Write Timing

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 continues to transmit additional bytes of data to the ZDI slave on the eZ80F91 device. 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 54 displays the timing for ZDI block Write operations.

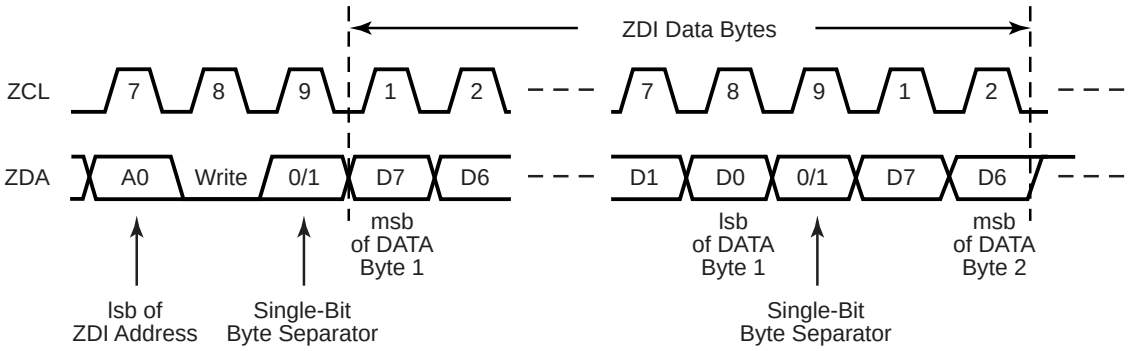


Figure 54. ZDI Block Data Write Timing

ZDI Read Operations

ZDI Single-Byte Read

Single-byte Read operations are initiated in the same manner as single-byte Write operations, with the exception that the R/$\overline{W}$ bit of the ZDI register address is set to 1. Upon receipt of a slave address with the R/$\overline{W}$ bit set to 1, the eZ80F91 device'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 55 displays the timing for ZDI single-byte Read operations.

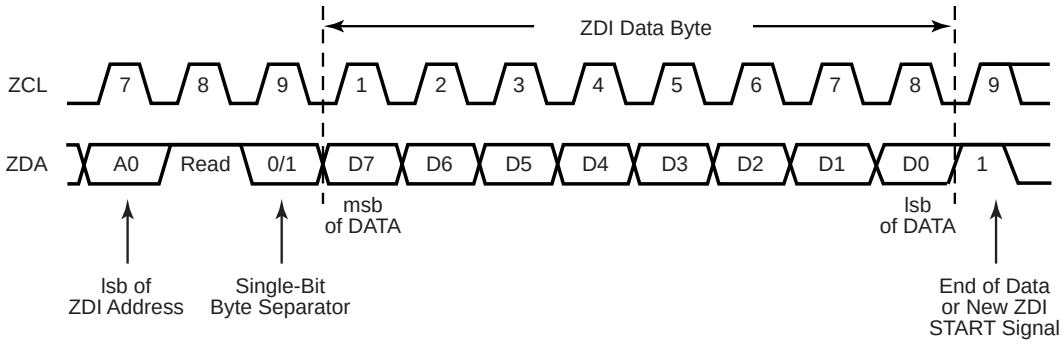


Figure 55. ZDI Single-Byte Data Read Timing

Note: *In ZDI single-byte read operations, after each read operation, the Program Counter (PC) address is incremented by two bytes. For example, if the current PC address is 0x00, then*

a read operation at 0x00 increments the PC to 0x02. To read the next byte, the PC must be decremented by one.

ZDI Block Read

A block Read operation is initiated in the same manner 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 56 displays the ZDI's block Read timing.

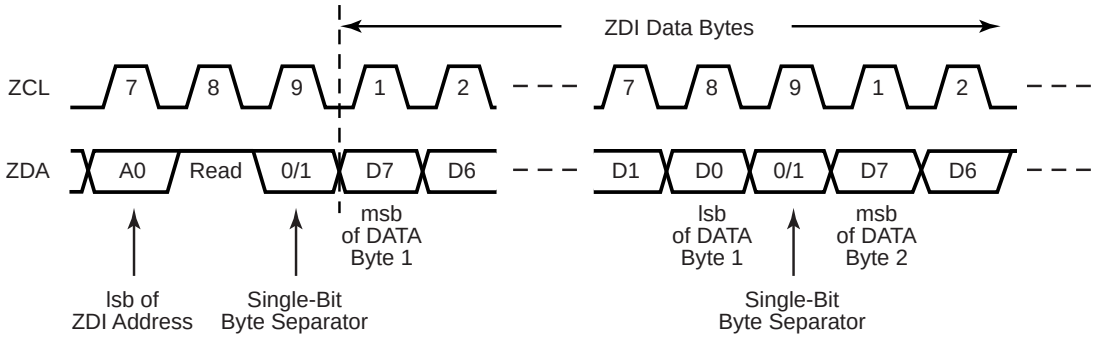


Figure 56. ZDI Block Data Read Timing

Operation of the eZ80F91 Device during ZDI Break Points

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

When using the ZDI interface, any Write or Read operations of peripheral registers in the I/O address space produces the same effect as Read or Write operations using the CPU. As 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 eZ80F91 device allows an external device to take control of the address and data bus while the eZ80F91 device is in DEBUG mode. ZDI_BUSACK_EN causes ZDI to allow or prevent acknowledgement of bus requests by external peripherals. The bus acknowledge occurs only 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 acknowl-

edgement to be disabled. To allow bus acknowledgement, the ZDI_BUSACK_EN must be written.

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

Potential Hazards of Enabling Bus Requests During DEBUG Mode

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

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

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

ZDI Write Only Registers

Table 133 lists the ZDI Write Only registers. Many of the ZDI Write Only addresses are shared with ZDI Read Only registers.

Table 133. 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
06h	ZDI_ADDR1_U	Address Match 1 Upper Byte	XXh

Table 133. ZDI Write Only Registers (Continued)

ZDI Address	ZDI Register Name	ZDI Register Function	Reset Value
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 134 lists the ZDI Read Only registers. Many of the ZDI Read Only addresses are shared with ZDI Write Only registers.

Table 134. ZDI Read Only Registers

ZDI Address	ZDI Register Name	ZDI Register Function	Reset Value
00h	ZDI_ID_L	eZ80® Product ID Low Byte Register	08h
01h	ZDI_ID_H	eZ80 Product ID High Byte Register	00h
02h	ZDI_ID_REV	eZ80 Product ID Revision Register	XXh

Table 134. ZDI Read Only Registers (Continued)

ZDI Address	ZDI Register Name	ZDI Register Function	Reset Value
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 eZ80F91 address is compared with the 3-byte address set, {ZDI_ADDRx_U, ZDI_ADDRx_H, and 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 eZ80F91 device placing the CPU 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 are used in conjunction with each other to break on branching instructions. See Table 135 on page 241.

Table 135. ZDI Address Match Registers

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_addrx_l, zdi_addrx_h, or zdi_addrx_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.

Address Information for 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.

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 occurs only 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 136 on page 243.

Table 136. 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 uses 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 is also 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 occurs 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 occurs 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 eZ80F91 device. It is capable of forcing a RESET and waking up the eZ80F91 from the LOW-POWER modes (HALT or SLEEP). See Table 137.

Table 137. 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 eZ80F91. 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 138.

Table 138. 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 LSBs.

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 eZ80F91 device immediately performs the operation corresponding to the data value written as listed in Table 139. 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 139. For information on the CPU registers, refer to *eZ80® CPU User Manual (UM0077)* available on www.zilog.com.

Note: *The 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 CPU register set.*

Table 139. 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 AF 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

Table 139. 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
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
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

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 $\overline{\text{BUSACK}}$ signal. This register must only be written during ZDI DEBUG mode (that is, following a break). See Table 140.

Table 140. 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 $\overline{\text{BUSREQ}}$ pin are ignored. The bus acknowledge signal, $\overline{\text{BUSACK}}$, is not asserted in response to any bus requests.
	1	Bus requests by external peripherals using the $\overline{\text{BUSREQ}}$ pin are accepted. A bus acknowledge occurs at the end of the current ZDI operation. The bus acknowledge is indicated by asserting the $\overline{\text{BUSACK}}$ pin in response to a bus request.
6 ZDI_BUSAK	0	Deassert the bus acknowledge pin ($\overline{\text{BUSACK}}$) to return control of the address and data buses back to ZDI.
	1	Assert the bus acknowledge pin ($\overline{\text{BUSACK}}$) 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 are written with instruction data for direct execution by the CPU. When the ZDI_IS0 register is written, the eZ80F91 device exits the ZDI break state and executes a single instruction. The opcodes 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 CPU requires to execute the instruction must be stored in these registers. Some CPU 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 141.

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 141. 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 eZ80F91 device 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 142 on page 251.

Table 142. 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. See Table 143 and Table 144 on page 252.

Table 143. eZ80 Product ID Low Byte Register (ZDI_ID_L = 00h in the ZDI Register Read Only Address Space, ZDI_ID_L = 0000h in the I/O Register Address Space)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	1	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_l	08h	{ZDI_ID_H, ZDI_ID_L} = {00h, 08h} indicates the eZ80F91 product.

Table 144. eZ80® Product ID High Byte Register (ZDI_ID_H = 01h in the ZDI Register Read Only Address Space, ZDI_ID_H = 0001h in the I/O Register 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, 08h} indicates the eZ80F91 device.

eZ80 Product ID Revision Register

The eZ80 Product ID Revision register identifies the current revision of the eZ80F91 product. See Table 145.

Table 145. eZ80 Product ID Revision Register (ZDI_ID_REV = 02h in the ZDI Register Read Only Address Space, ZDI_ID_REV = 0002h in the I/O Register 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 eZ80F91 product.

ZDI Status Register

The ZDI Status register provides current information on the eZ80F91 device and the CPU. See Table 146.

Table 146. 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	The CPU is not currently in HALT or SLEEP mode.
	1	The CPU 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 Register 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 147.

Table 147. ZDI Read Register 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 148.

Table 148. 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 eZ80F91 device 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]. For more information on Z80 and ADL MEMORY modes, refer to the *eZ80® CPU User Manual (UM0077)* available on www.zilog.com. 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 reads any number of data bytes out of memory via the ZDI Read Memory register. See Table 149 on page 256.

Note that the delay between issuing a memory read request and the return of the corresponding data amount to multiple ZDI clock cycles. This delay is a function of the wait state configuration of the memory space being accessed as well as the relative frequencies of the ZDI clock and the system clock. If the ZDI master begins clocking the read data out of the eZ80F91 soon after issuing the memory read request, invalid data will be returned. Since no data-valid handshake mechanism exists in the ZDI protocol, the ZDI master must account for expected memory read delay in some way.

A technique exists to mask this delay in almost all situations. It always reads at least two consecutive bytes, starting one address lower than the address of interest. In this situation, the eZ80F91 internally prefetches the data from the second address while the ZDI master is sending the second read request. This allows enough time for the second ZDI memory read to return valid data. The first data byte returned to the ZDI master must be discarded since it is invalid. Memory reads of more than two consecutive bytes will also return correct data for all but the first address.

Table 149. 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 eZ80® CPU core enables powerful debugging features. The OCI provides run control, memory and register visibility, complex break points, 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 Joint Test Action Group (JTAG) port that conforms to IEEE Standard 1149.1 (Test Access Port and Boundary Scan Architecture)
- Complex break point trigger functions
- Break point enhancements, such as the ability to:
  - Define two break point addresses that form a range
  - Break on masked data values
  - Start or stop trace
  - Assert a trigger output signal
- Trace history buffer
- Software break point instruction

There are four sections to the OCI:

- JTAG interface
- ZDI debug control
- Trace buffer memory
- Complex triggers

This document contains information to activate the OCI for JTAG boundary scan register operations. For additional information regarding OCI features, or to order OCI debug tools, contact:

First Silicon Solutions, Inc.
www.fs2.com

1. On-Chip Instrumentation and OCI are trademarks of First Silicon Solutions, Inc.

OCI Activation

OCI features clock initialization circuitry so that external debug hardware is 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 shut down, ZDI is enabled directly and is accessed via the clock (TCK) and data (TDI) pins. For more information on ZDI, see Zilog Debug Interface on page 231.

OCI Interface

There are six dedicated pins on the eZ80F91 for the OCI interface. Four pins—TCK, TMS, TDI, and TDO—are required for IEEE Standard 1149.1-compliant JTAG ports. A fifth pin, TRSTn, is optional for IEEE 1149.1 and utilized by the eZ80F91 device. The TRIGOUT pin provides additional testability features. These six OCI pins are listed in Table 150.

Table 150. OCI Pins

Symbol	Name	Type	Description
TCK	Clock	Input	Asynchronous to the primary eZ80F91 system clock. The TCK period 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).
TRSTn	TAP Reset	Input	Active Low asynchronous reset for the Test Access Port state register. 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. 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. NORMAL mode, following RESET, configures TDI as an input.

Table 150. OCI Pins (Continued)

Symbol	Name	Type	Description
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 open-drain when no data is being driven out.

JTAG Boundary Scan

Introduction

This section describes coverage, implementation, and usage of the eZ80F91 boundary scan register based on the JTAG standard. A working knowledge of the IEEE 1149.1 specification, particularly Clause 11, is required.

Pin Coverage

All pins are included in the boundary scan chain, except the following:

- TCK
- TMS
- TDI
- TDO
- TRSTN
- V_{DD}
- V_{SS}
- PLL_V_{DD}
- PLL_V_{SS}
- RTC_V_{DD}
- X_{IN}
- X_{OUT}
- RTC_X_{IN}
- RTC_X_{OUT}
- LOOP_FILT

Boundary Scan Cell Functionality

The boundary scan cells implemented are analogous to cell BC_1, defined in the Standard VHDL Package STD_1149_1_2001.

All boundary scan cells are of the type *control-and-observe*; they provide both controllability and observability for the pins to which they are connected. For open-drain outputs and bidirectional pins, this type includes controllability and observability of output enables.

Chain Sequence and Length

When enabled to shift data, the boundary scan shift register is connected to TDI at the input line for TRIGOUT and to TDO at PD0. The shift register is arranged so that data is shifted via the pins starting to the left of the OCI interface pins and proceeding clockwise around the chip. If a pin features multiple scannable bits (example: bidirectional pins or open-drain output pins), the data is shifted first into the input signal, then the output, then the output enable (OEN).

The boundary scan register is 213 bits wide. Table 151 lists the ordering of bits in the shift register, numbering them in clockwise order.

Table 151. Pin to Boundary Scan Cell Mapping

Pin	Direction	Scan Cell No	Pin	Direction	Scan Cell No
TRIGOUT	Input	0	MII_TxD2	Output	107
TRIGOUT	Output	1	MII_TxD3	Output	108
TRIGOUT	OEN	2	MII_COL	Input	109
HALT_SLP	Output	3	MII_CRS	Input	110
BUSACK	Output	4	PA7	Input	111
BUSREQ	Input	5	PA7	Output	112
NMI	Input	6	PA7	OEN	113
RESET	Input	7	PA6	Input	114
RESET_OUT	Output	8	PA6	Output	115
WAIT	Input	9	PA6	OEN	116
INSTRD	Output	10	PA5	Input	117
WR	Output	11	PA5	Output	118
WR	OEN	12	PA5	OEN	119
RD	Output	13	PA4	Input	120
MREQ	Input	14	PA4	Output	121

Table 151. Pin to Boundary Scan Cell Mapping (Continued)

Pin	Direction	Scan Cell No	Pin	Direction	Scan Cell No
MREQ	Output	15	PA4	OEN	122
IORQ	Input	16	PA3	Input	123
IORQ	Output	17	PA3	Output	124
D7	Input	18	PA3	OEN	125
D7	Output	19	PA2	Input	126
D6	Input	20	PA2	Output	127
D6	Output	21	PA2	OEN	128
D5	Input	22	PA1	Input	129
D5	Output	23	PA1	Output	130
D4	Input	24	PA1	OEN	131
D4	Output	25	PA0	Input	132
D3	Input	26	PA0	Output	133
D3	Output	27	PA0	OEN	134
D2	Input	28	PHI	Output	135
D2	Output	29	PHI	OEN	136
D1	Input	30	SCL	Input	137
D1	Output	31	SCL	Output	138
D0	Input	32	SDA	Input	139
D0	Output	33	SDA	Output	140
D0	OEN	34	PB7	Input	141
CS3	Output	35	PB7	Output	142
CS2	Output	36	PB7	OEN	143
CS1	Output	37	PB6	Input	144
CS0	Output	38	PB6	Output	145
A23	Input	39	PB6	OEN	146
A23	Output	40	PB5	Input	147
A22	Input	41	PB5	Output	148
A22	Output	42	PB5	OEN	149
A21	Input	43	PB4	Input	150

Table 151. Pin to Boundary Scan Cell Mapping (Continued)

Pin	Direction	Scan Cell No	Pin	Direction	Scan Cell No
A21	Output	44	PB4	Output	151
A20	Input	45	PB4	OEN	152
A20	Output	46	PB3	Input	153
A19	Input	47	PB3	Output	154
A19	Output	48	PB3	OEN	155
A18	Input	49	PB2	Input	156
A18	Output	50	PB2	Output	157
A17	Input	51	PB2	OEN	158
A17	Output	52	PB1	Input	159
A16	Input	53	PB1	Output	160
A16	Output	54	PB1	OEN	161
A16	OEN	55	PB0	Input	162
A15	Input	56	PB0	Output	163
A15	Output	57	PB0	OEN	164
A14	Input	58	PC7	Input	165
A14	Output	59	PC7	Output	166
A13	Input	60	PC7	OEN	167
A13	Output	61	PC6	Input	168
A12	Input	62	PC6	Output	169
A12	Output	63	PC6	OEN	170
A11	Input	64	PC5	Input	171
A11	Output	65	PC5	Output	172
A10	Input	66	PC5	OEN	173
A10	Output	67	PC4	Input	174
A9	Input	68	PC4	Output	175
A9	Output	69	PC4	OEN	176
A8	Input	70	PC3	Input	177
A8	Output	71	PC3	Output	178
A8	OEN	72	PC3	OEN	179

Table 151. Pin to Boundary Scan Cell Mapping (Continued)

Pin	Direction	Scan Cell No	Pin	Direction	Scan Cell No
A7	Input	73	PC2	Input	180
A7	Output	74	PC2	Output	181
A6	Input	75	PC2	OEN	182
A6	Output	76	PC1	Input	183
A5	Input	77	PC1	Output	184
A5	Output	78	PC1	OEN	185
A4	Input	79	PC0	Input	186
A4	Output	80	PC0	Output	187
A3	Input	81	PC0	OEN	188
A3	Output	82	PD7	Input	189
A2	Input	83	PD7	Output	190
A2	Output	84	PD7	OEN	191
A1	Input	85	PD6	Input	192
A1	Output	86	PD6	Output	193
A0	Input	87	PD6	OEN	194
A0	Output	88	PD5	Input	195
A0	OEN	89	PD5	Output	196
WP	Input	90	PD5	OEN	197
MII_MDIO	Input	91	PD4	Input	198
MII_MDIO	Output	92	PD4	Output	199
MII_MDIO	OEN	93	PD4	OEN	200
MII_MDC	Output	94	PD3	Input	201
MII_RxD3	Input	95	PD3	Output	202
MII_RxD2	Input	96	PD3	OEN	203
MII_RxD1	Input	97	PD2	Input	204
MII_RxD0	Input	98	PD2	Output	205
MII_Rx_DV	Input	99	PD2	OEN	206
MII_Rx_CLK	Input	100	PD1	Input	207
MII_Rx_ER	Input	101	PD1	Output	208

Table 151. Pin to Boundary Scan Cell Mapping (Continued)

Pin	Direction	Scan Cell No	Pin	Direction	Scan Cell No
MII_Tx_ER	Output	102	PD1	OEN	209
MII_Tx_CLK	Input	103	PD0	Input	210
MII_Tx_EN	Output	104	PD0	Output	211
MII_TxD0	Output	105	PD0	OEN	212
MII_TxD1	Output	106			

Notes
1. The address bits 0–7, 8–15, and 16–23 each share a single output enable. In this table, the output enables are associated with the LSb that they control.
2. Direction on the data bus is controlled by a single output enable. It is associated in this table with D[0].
3. MREQ, IORQ, INSTRDN, RD, and WR share an output enable; it is associated in this table with WR.

Usage

Boundary scan functionality is utilized by issuing the appropriate Test Access Port (TAP) instruction and shifting data accordingly. Both of these steps are accomplished using the JTAG interface. To activate the TAP (see OCI Activation on page 258), the TCK pin must be driven Low at least two CPU system clock cycles prior to the deassertion of the RESET pin. Otherwise the OCI-JTAG features are disabled.

As per the IEEE 1149.1 specification, the boundary scan cells capture system I/O on the rising edge of TCK during the CAPTURE_DR state. This captured data is shifted on the rising edge of TCK while in the SHIFT_DR state. Pins and logic receive shifted data only when enabled, and only on the falling edge of TCK during the UPDATE_DR state, after shifting is completed.

For more information about eZ80F91 boundary scan support, refer to *Using BSDL Files with eZ80® and eZ80Acclaim!® Devices (AN0114)*.

Boundary Scan Instructions

The eZ80F91 device's boundary scan architecture supports the following instructions:

- BYPASS (required)
- SAMPLE (required)
- EXTEST (required)
- PRELOAD (required)
- IDCODE (optional)

Phase-Locked Loop

Overview

The Phase-Locked-Loop (PLL) is a programmable frequency multiplier that satisfies the equation SCLK (Hz) = N * F_{OSC}(Hz). Figure 57 displays the PLL block diagram.

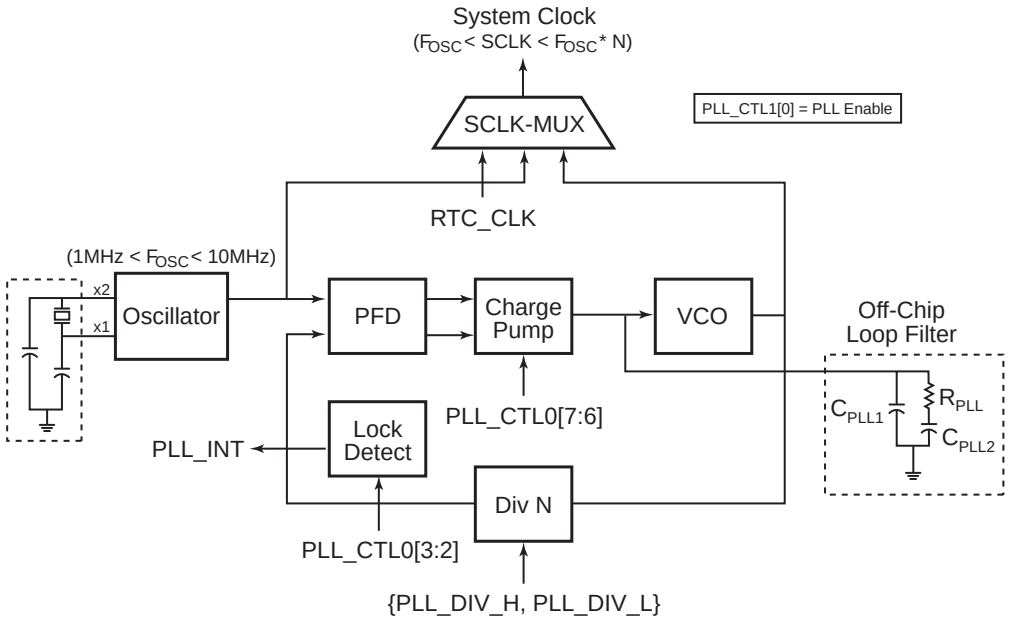


Figure 57. Phase-Locked Loop Block Diagram

PLL includes seven main blocks as listed below:

- Phase Frequency Detector
- Charge Pump
- Voltage Controlled Oscillator
- Loop Filter
- Divider
- MUX/CLK Sync
- Lock Detect

Phase Frequency Detector

The Phase Frequency Detector (PFD) is a digital block. The two inputs are the reference clock (XTAL oscillator; see On-Chip Oscillators on page 335) and the PLL divider output. The two outputs drive the internal charge pump and represent the error (or difference) between the falling edges of the PFD inputs.

Charge Pump

The Charge Pump is an analog block that is driven by two digital inputs from the PFD that control its programmable current sources. The internal current source contains four programmable values: 1.5 mA, 1 mA, 500 µA, and 100 µA. These values are selected by PLL_CTRL1[7:6]. The selected current drive is sinked/sourced onto the loop-filter node according to the error (or difference) between the falling edges of the PFD inputs. Ideally, when the PLL is locked, there are no errors (error = 0) and no current is sourced/sinked onto the loop-filter node.

Voltage Controlled Oscillator

The Voltage Controlled Oscillator (VCO) is an analog block that exhibits an output frequency proportional to its input voltage. The VCO input is driven from the charge pump and filtered via the off-chip loop filter.

Loop Filter

The Loop Filter comprises off-chip passive components (usually 1 resistor and 2 capacitors) that filter/integrate charge from the internal charge pump. The filtered node also drives the VCO input, which creates a proportional frequency output. When PLL is not used, the Loop Filter pin must not be connected.

Divider

The Divider is a digital, programmable downcounter. The divider input is driven by the VCO. The divider output drives the PFD. The function of the Divider is to divide the frequency of its input signal by a programmable factor *N* and supply the result in its output.

MUX/CLK Sync

The MUX/CLK Sync is a digital, software-controllable multiplexer that selects between PLL or the XTAL oscillator as the system clock (SCLK). A PLL source is selected only after the PLL is *locked* (via the lock detect block) to allow glitch-free clock switching.

Lock Detect

The Lock Detect digital block analyzes the PFD output for a locked condition. The PLL block of the eZ80F91 device is considered locked when the error (or difference) between the reference clock and divided-down VCO is less than the minimum timing lock criteria

for the number of consecutive reference clock cycles. The lock criteria is selected in the PLL Control Register, PLL_CTL0[LDS_CTL]. When the locked condition is met, this block outputs a logic High signal (lock) that interrupts the CPU.

PLL Normal Operation

By default (after system reset) the PLL is disabled and SCLK = XTAL oscillator. Ensuring proper loop filter, supply voltages and external oscillator are correctly configured, the PLL is enabled. The SCLK/Timer cannot choose the PLL as its source until the PLL is locked, as determined by the lock detect block. By forcing the PLL to be locked prior to enabling the PLL as a SCLK/Timer source, it is assured to be stable and accurate.

Figure 58 displays the programming flow for normal PLL operation.

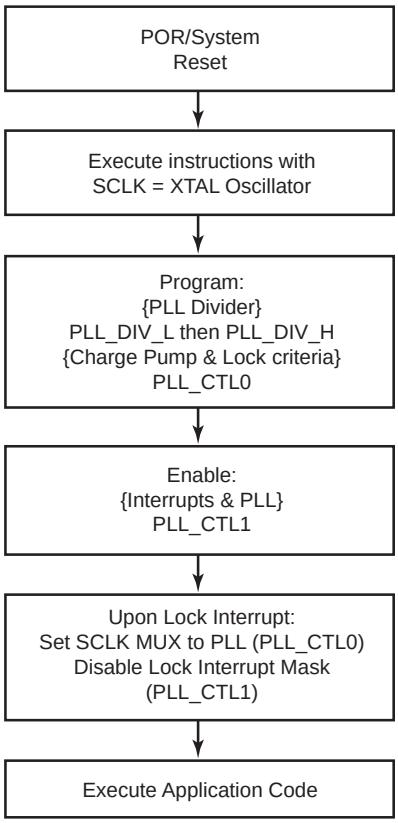


Figure 58. Normal PLL Programming Flow

Power Requirement to the Phase-Locked Loop Function

Regardless of whether or not you chooses to use the PLL module block as a clock source for the eZ80F91 device, the PLL_V_{DD} (pin 87) must be connected to a V_{DD} supply and the PLL_V_{SS} (pin 84) must be connected to a V_{SS} supply for proper operation of the eZ80F91 using any system clock source.

PLL Registers

PLL Divider Control Register—Low and High Bytes

This register is designed such that the 11 bit divider value is loaded into the divider module whenever the PLL_DIV_H register is written. Therefore, the procedure must be to load the PLL_DIV_L register, followed by the PLL_DIV_H register, for the divider to receive the appropriate value.

The divider is designed such that any divider value less than two is ignored; a value of two is used in its place.

The LSB of PLL divider N is set via the corresponding bits in the PLL_DIV_L register. See Table 152 and Table 153 on page 269.

Note: *The PLL divider register are written only when the PLL is disabled. A read-back of the PLL Divider registers returns 0.*

Table 152. PLL Divider Register—Low Bytes (PLL_DIV_L = 005Ch)

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

Note: W = Write only.

Bit Position	Value	Description
[7:0] PLL_DIV_L	00h–FFh	These bits represent the Low byte of the 11 bit PLL divider value. The complete PLL divider value is returned by {PLL_DIV_H, PLL_DIV_L}.

Table 153. PLL Divider Register—High Bytes (PLL_DIV_H = 005Dh)

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

Bit Position	Value	Description
[7:3]	00h	Reserved
[2:0] PLL_DIV_H	0h–7h	These bits represent the High byte of the 11 bit PLL divider value. The complete PLL divider value is returned by {PLL_DIV_H, PLL_DIV_L}.

PLL Control Register 0

The charge pump program, lock detect sensitivity, and system clock source selections are set using this register. A brief description of each of these PLL Control Register 0 attributes is listed below, and further listed in Table 154.

Charge Pump Program (CHRP_CTL)—Selects one of four values of charge pump current.

Lock Detect Sensitivity (LDS_CTL)—Determines the lock criteria for the PLL.

System Clock Source (CLK_MUX)—Selects the system clock source from a choice of the external crystal oscillator (XTAL), PLL, or Real-Time Clock crystal oscillator.

Table 154. PLL Control Register 0 (PLL_CTL0 = 005Eh)

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

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

Bit Position	Value	Description
[7:6] CHRP_CTL1	00	Charge pump current = 100 µA
	01	Charge pump current = 500 µA
	10	Charge pump current = 1.0 mA
	11	Charge pump current = 1.5 mA

Bit Position	Value	Description
[5:4]	00	Reserved
[3:2] LDS_CTL1	00	Lock criteria—8 consecutive cycles of 20 ns
	01	Lock criteria—16 consecutive cycles of 20 ns
	10	Lock criteria—8 consecutive cycles of 400 ns
	11	Lock criteria—16 consecutive cycles of 400 ns
[1:0] CLK_MUX	00	System clock source is the external crystal oscillator
	01	System clock source is the PLL[2]
	10	System clock source is the Real-Time Clock crystal oscillator
	11	Reserved (previous select is preserved)

Notes
1. Bits are programmed only when the PLL is disabled. The PLL is disabled when PLL_CTL1 bit 0 is equal to 0.
2. PLL cannot be selected when disabled or *out of lock*.

PLL Control Register 1

The PLL is enabled using this register. PLL lock-detect status, the PLL interrupt signals and the PLL interrupt enables are accessed via this register. A brief description of each of these PLL Control Register 1 attributes is listed below, and further listed in Table 155 on page 271.

Lock Status (LCK_STATUS)—The current lock bit out of the PLL is synchronized and read via this bit.

Interrupt Lock (INT_LOCK)—This signal feeds the interrupt line out of the CLKGEN module and indicates that a rising edge on the lock signal out of the PLL has been observed.

Interrupt Unlock (INT_UNLOCK)—This signal feeds the interrupt line out of the clkgen module and indicates that a falling edge on the lock signal out of the PLL has been observed.

Interrupt Lock Enable (INT_LOCK_EN)—This signal enables the interrupt lock bit.

Interrupt Unlock Enable (INT_UNLOCK_EN)—This signal enables the interrupt unlock bit.

PLL Enable (PLL_ENABLE)—Enables/disables the PLL.

Table 155. PLL Control Register 1 (PLL_CTL1 = 005Fh)

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
[7:6]	00	Reserved.
5 LCK_STATUS	0	PLL is currently out of lock.
	1	PLL is currently locked.
4 INT_LOCK	0	Lock signal from PLL has not risen since last time register was read.
	1	Interrupt generated when PLL enters LOCK mode. Held until register is read.
3 INT_UNLOCK	0	Lock signal from PLL has not fallen since last time register was read.
	1	Interrupt generated when PLL goes out of lock. Held until register is read.
2 INT_LOCK_EN	0	Interrupt generation for PLL locked condition (Bit 4) is disabled.
	1	Interrupt generation for PLL locked condition is enabled.
1 INT_UNLOCK_EN	0	Interrupt generation for PLL unlocked condition (Bit 3) is disabled.
	1	Interrupt generation for PLL unlocked condition is enabled.
0 PLL_ENABLE	0	PLL is disabled.[1]
	1	PLL is enabled.

Note

1. PLL cannot be disabled if the CLK_MUX bit of PLL_CTL0[1:0] is set to 01, because the PLL is selected as the clock source.

PLL Characteristics

The operating and testing characteristics for the PLL are listed in Table 156.

Note: *Not all conditions are tested in production test. The values in Table 156 are for design and characterization only.*

Table 156. PLL Characteristics

Symbol	Parameter	Test Condition	Min	Typ	Max	Units
I_{oHCP_OUT}	High level output current for CP_OUT pin (programmed value ± 42%)	3.0 < V_{DD} < 3.6 0.6 < PD_OUT < V_{DD} – 0.6 PLL_CTL0[7:6] = 11	–0.86	–1.50	–2.13	mA
I_{oLCP_OUT}	Low level output current for CP_OUT pin (programmed value ± 42%)	3.0 < V_{DD} <3.6 0.6 < PD_OUT < V_{DD} – 0.6 PLL_CTL0[7:6] = 11	0.86	1.50	2.13	mA
I_{oHCP_OUT}	High level output current for CP_OUT pin (programmed value ± 42%)	3.0 < V_{DD} <3.6 0.6 < PD_OUT < V_{DD} – 0.6 PLL_CTL0[7:6] = 10	–0.42	–1.0	–1.42	mA
I_{oLCP_OUT}	Low level output current for CP_OUT pin (programmed value ± 42%)	3.0 < V_{DD} <3.6 0.6 < PD_OUT <V_{DD} – 0.6 PLL_CTL0[7:6] = 10	0.42	1.0	1.42	mA
I_{oHCP_OUT}	High level output current for CP_OUT pin (programmed value ± 42%)	3.0 < V_{DD} <3.6 0.6 < PD_OUT <V_{DD} – 0.6 PLL_CTL0[7:6] = 01	–210	–500	–710	µA
I_{oLCP_OUT}	Low level output current for CP_OUT pin (programmed value ± 42%)	3.0 < V_{DD} <3.6 0.6 < PD_OUT <V_{DD} – 0.6 PLL_CTL0[7:6] = 01	210	500	710	µA
I_{oHCP_OUT}	High level output current for CP_OUT pin (programmed value ± 42%)	3.0 < V_{DD} <3.6 0.6 < PD_OUT <V_{DD} – 0.6 PLL_CTL0[7:6] = 00	–42	–100	–142	µA
I_{oLCP_OUT}	Low level output current for CP_OUT pin (programmed value ± 42%)	3.0 < V_{DD} <3.6 0.6 < PD_OUT <V_{DD} – 0.6 PLL_CTL0[7:6] = 00	42	100	142	µA
Match	I_{OHCP_OUT}–I_{OLCP_OUT} current match	3.0 < V_{DD} <3.6 0.6 < CP_OUT <V_{DD} – 0.6 PLL_CTL0[7:6] = XX	–15		+15	%
I_{LCP_OUT}	Tristate leakage on CP_OUT output pin	CP_OUT tristated	–1		1	µA
F_{osc}	Crystal oscillator frequency	PLL_CTL0[5:4] = 01	1 M		10 M	Hz

Table 156. PLL Characteristics (Continued)

Symbol	Parameter	Test Condition	Min	Typ	Max	Units
F_{vco}	VCO frequency	Recommended operating conditions		50		MHz
G_{vco}	VCO Gain	Recommended operating conditions	36		120	MHz/V
D1	SCLK Duty Cycle from PLL or XTALOSC source	Recommended operating conditions	45	50	55	%
T1A	PLL Clock Jitter	F_{VCO} = 50 MHz. XTALOSC = 10 MHz		350	500	ps
Lock2	PLL Lock-Time	F_{VCO} = 50 MHz. XTALOSC = 3.579 MHz C_{pll1} = 220 pF, R_{pll} = 499 ¾, C_{pll2} = 0.056 µF				s
I_{oH1} (XTL)	High-level Output Current for XTAL2 pin	$V_{oH} = V_{DD}$–0.4 V PLL_CTL0[5:4] = 01	–0.3			mA
I_{oL1} (XTL)	Low-level Output Current for XTAL2 pin	V_{oL} = 0.4 V PLL_CTL0[5:4] = 01	0.6			mA
I_{oH2} (XTL)	High-level Output Current for XTAL2 pin	$V_{oH} = V_{DD}$–0.4 V PLL_CTL0[5:4] = 11				mA
I_{oL2} (XTL)	Low-level Output Current for XTAL2 pin	V_{oL} = 0.4 V PLL_CTL0[5:4] = 11				mA
V_{PP3M} (XTL)	Peak-to-peak voltage under oscillator conditions for XTAL2 pin	F_{OSC} = 3.579 MHz Cx1 = 10 pF Cx2 = 10 pF				V
V_{PP10M} (XTL)	Peak-to-peak voltage under oscillator conditions for XTAL2 pin	F_{OSC} = 10 MHz Cx1 = 10 pF Cx2 = 10 pF				V
C_{xtal1} (package type)	Capacitance measured from XTAL1 pin to GND	T = 25 ºC				pF
C_{xtal2} (package type)	Capacitance measured from XTAL2 pin to GND	T = 25 ºC				pF
C_{loop} (package type)	Capacitance measured from loop filter pin to GND	T = 25 ºC				pF

eZ80® CPU Instruction Set

Table 157 through Table 166 on page 278 lists the CPU instructions available for use with the eZ80F91 device. The instructions are grouped by class. For more information, refer to *eZ80® CPU User Manual (UM0077)*.

Table 157. 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 158. Bit Manipulation Instructions

Mnemonic	Instruction
BIT	Bit Test
RES	Reset Bit
SET	Set Bit

Table 159. 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 160. Exchange Instructions

Mnemonic	Instruction
EX	Exchange registers
EXX	Exchange CPU Multibyte register banks

Table 161. 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/0 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)

Table 161. Input/Output Instructions (Continued)

Mnemonic	Instruction
OUTI2 (OTI2R)	Output to I/O and Increment (with Repeat)
TSTIO	Test I/O

Table 162. Load Instructions

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

Table 163. Logical Instructions

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

Table 164. Processor Control Instructions

Mnemonic	Instruction
CCF	Complement Carry Flag
DI	Disable Interrupts
EI	Enable Interrupts
HALT	Halt
IM	Interrupt Mode
NOP	No Operation

Table 164. Processor Control Instructions (Continued)

Mnemonic	Instruction
RSMIX	Reset Mixed-Memory Mode Flag
SCF	Set Carry Flag
SLP	Sleep
STMIX	Set Mixed-Memory Mode Flag

Table 165. 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 166. 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

Table 166. Rotate and Shift Instructions (Continued)

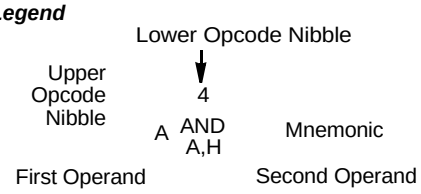
Mnemonic	Instruction
RRA	Rotate Right–Accumulator
RRC	Rotate Right Circular
RRCA	Rotate Right Circular–Accumulator
RRD	Rotate Right Decimal
SLA	Shift Left Arithmetic
SRA	Shift Right Arithmetic
SRL	Shift Right Logical

Opcode Map

Table 167 through Table 173 on page 286 list the hex values for each of the eZ80® instructions.

Table 167. Opcode Map—First Opcode

Legend



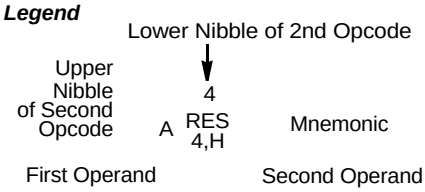
Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LD BC, Mmn	LD (BC),A	INC BC	INC B	DEC B	LD B,n	RLCA	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	RRCA
1	DJNZ d	LD DE, Mmn	LD (DE),A	INC DE	INC D	DEC D	LD D,n	RLA	JR d	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	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	See Table 168	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	See Table 169	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	See Table 170	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	See Table 171	CP A,n	RST 38h

Table 167. Opcode Map—First Opcode (Continued)

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

Table 168. Opcode Map—Second Opcode after 0CBh

Legend



Upper Nibble (Hex) \ Lower 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

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

Table 169. Opcode Map—Second Opcode After 0DDh

Legend

Lower Nibble of 2nd Opcode

Upper Nibble of Second Opcode

9

F LD SP,IX

Mnemonic

First Operand

Second Operand

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, (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 172				
D																
E		POP IX		EX (SP),IX		PUSH IX				JP (IX)						
F										LD SP,IX						

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

Table 170. Opcode Map—Second Opcode After 0EDh

Legend

Lower Nibble of 2nd Opcode

Upper Nibble of Second Opcode

2

4 SBC HL,BC

Mnemonic

First Operand

Second Operand

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				LD I,HL			INDRX	OTDRX				
D								LD HL,I								
E																
F																

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

Table 171. Opcode Map—Second Opcode After 0FDh

Legend

Lower Nibble of 2nd Opcode

Upper Nibble of Second Opcode

9

F LD SP,IY

Mnemonic

First Operand

Second Operand

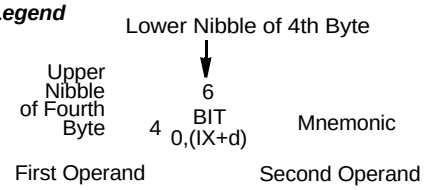
Lower Nibble (Hex)

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 173				
D																
E		POP IY		EX (SP),IY		PUSH IY				JP (IY)						
F										LD SP,IY						

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

Table 172. Opcode Map—Fourth Byte After 0DDh, 0CBh, and dd

Legend

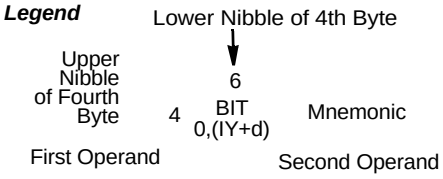


Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0							RLC (IX+d)								RRC (IX+d)	
1							RL (IX+d)								RR (IX+d)	
2							SLA (IX+d)								SRA (IX+d)	
3															SRL (IX+d)	
4							BIT 0, (IX+d)								BIT 1, (IX+d)	
5							BIT 2, (IX+d)								BIT 3, (IX+d)	
6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

Note: d = 8-bit two's-complement displacement

Table 173. Opcode Map—Fourth Byte After 0FDh, 0CBh, and dd

Legend



Upper Nibble (Hex) \ Lower 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)	

Note: d = 8-bit two's-complement displacement

Ethernet Media Access Controller

The Ethernet Media Access Controller (EMAC) is a full-function 10/100 Mbps media access control module with a Media-Independent Interface (MII). When communicating with an external PHY device, the eZ80F91 MCU uses the MII to gain access to the Ethernet network.

Figure 59 displays the EMAC block diagram.

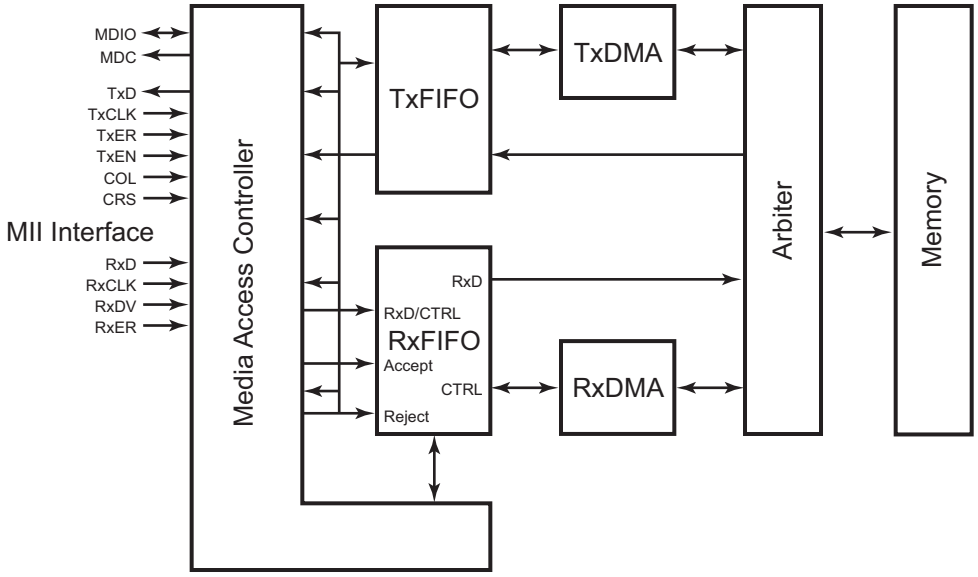


Figure 59. EMAC Block Diagram

Note: *For additional information about the Ethernet protocol and using it with the eZ80F91 MCU, refer to the IEEE 802.3 specification, 1998 edition, Section 22. The eZ80F91 MCU supports the IEEE 802.3 protocol with the following exception:*

The eZ80F91 MCU does not support the Giga Media Independent Interface (GMII) referred to in the following sections of the IEEE 802.3 1998 version: section 22.1.5, section 22.2.4, section 22.2.4.1.2, section 22.2.4.1.5, and section 22.2.4.1.6.

The EMAC is used for many different applications, including network interface, ethernet switching, and test equipment designs. The EMAC includes the following blocks:

- Central clock and reset module (not shown in the block diagram).
- Host memory interface and transmit/receiver arbiter.

- FIFO buffer and DMA control blocks for transmit and receive.
- 802.3x media access control block.
- MII interface management.

The media access control block implements 802.3x flow control functions for both transmit and receive.

The MII management module provides a two-wire control/status path to the MII PHY. Read and Write communication to and from registers within the PHY is accomplished via the host interface.

Note: *MII PHY is a Physical Layer transceiver device; PHY does not refer to the eZ80F91 system clock output pin, PHI.*

The MII management module provides a two-wire control/status path to the MII. Read and Write communication to and from registers within the PHY is accomplished via the host interface.

EMAC Functional Description

The EMAC block implements memory, arbiter, and transmit and receive direct memory access functions, and offers four communication modes: HALF-DUPLEX, FULL-DUPLEX, NIBBLE, and ENDEC. In HALF-DUPLEX and FULL-DUPLEX modes, throughput occurs at both 10 Mbps and 100 Mbps speeds. Throughput in ENDEC and NIBBLE modes occurs at 10 Mbps. A brief description of these four modes are as follows:

10/100 Mbps HALF-DUPLEX Mode— In this mode, data are transferred only in one direction at a time; that is, one can either transmit or receive, but both cannot occur simultaneously.

10/100 Mbps FULL-DUPLEX Mode— In this mode, data are transmitted and received at the same time.

10 Mbps ENDEC Mode— This mode affects the MII interface between the PHY and the MAC. In ENDEC mode, the RxCLK and TxCLK clocks are bit clocks instead of the normal nibble clock. In NIBBLE mode, 4 bits are transferred on each clock. In ENDEC mode, 1 bit is transferred per clock.

For more information on throughput, see EMAC and the System Clock on page 296.

Memory

EMAC memory is the shared Ethernet memory location of the Transmit and Receive buffers. This memory is broken into two parts: the Tx buffer and the Rx buffer. The Transmit Lower Boundary Pointer Register, EmacTLBP, is the register that holds the starting address of the Tx buffer. The Boundary Pointer Register, EmacBP, points to the start of the Rx buffer (end of Tx buffer + 1). The Receive High Boundary Pointer Register,

EmacRHBP, points to the end of the Rx buffer + 1. The Tx and Receive buffers are divided into packet buffers of either 256, 128, 64, or 32 bytes. These buffer sizes are selected by EmacBufSize register bits 7 and 6.

The EmacBlksLeft register contains the number of Receive packet buffers remaining in the Rx buffer. This buffer is used for software flow control. If the Block_Level is nonzero (bits 5:0 of the EmacBufSize register), hardware flow control is enabled. If in FULL-DUPLEX mode, the EMAC transmits a pause control frame when the EmacBlksLeft register is less than the Block_Level. In HALF-DUPLEX mode, the EMAC continually transmits a nibble pattern of hexadecimal 5's to jam the channel.

Four pointers are defined for reading and writing the Tx and Rx buffers. The Transmit Write Pointer, TWP, is a software pointer that points to the next available packet buffer. The TWP is reset to the value stored in EmacTLBP. The Transmit Read Pointer, TRP, is a hardware pointer in the Transmit Direct Memory Access Register, TxDMA, that contains the address of the next packet to be transmitted. It is automatically reset to the EmacTLBP. The Receive Write Pointer, RWP, is a hardware pointer in the Receive Direct Memory Access Register, RxDMA, which contains the storage address of the incoming packet. The RWP pointer is automatically initialized to the Boundary Pointer registers. The Receive Read Pointer, RRP, is a software pointer to where the next packet must be read from. The RRP pointer must be initialized to the Boundary Pointer registers. For the hardware flow control to function properly, the software must update the hardware RRP (EmacRrp) pointer whenever the software version is updated. The RxDMA uses RWP and the RRP to determine how many packet buffers remain in the Rx buffer.

Arbiter

The arbiter controls access to EMAC memory. It prioritizes the requests for memory access between the CPU, the TxDMA, and the RxDMA. The TxDMA offers two levels of priority: a high priority when the TxFIFO is less than half full and a Low priority when the TxFIFO is more than half full. Similarly, the RxDMA offers two levels of priority: a high priority when the RxFIFO is more than half full and a Low priority when the RxFIFO is less than half full.

The arbiter determines resolution between the CPU, the RxDMA, and the TxDMA requests to access EMAC memory. Post writing for CPU Writes results in Zero-Wait-state write access timing when the CPU assumes the highest priority. CPU Reads require a minimum of 1 Wait state and takes more when the CPU does not hold the highest priority. The CPU Read Wait state is not a user-controllable operation, because it is controlled by the arbiter. The RxDMA and TxDMA requests are not allowed to occur back-to-back. Therefore, the maximum throughput rate for the two Direct Memory Access (DMA) ports is 25 Mbps each (one byte every 2 clocks) when the system clock is running at 50 MHz. The rate is reduced to 20 MBps for a 40 MHz system clock. The arbiter uses the internal WAIT signal to add Wait states to CPU access when required. See Table 174 on page 290.

Table 174. Arbiter Priority

Priority Level	Device Serviced	Flags
0	RxDMA High	RxFIFO > half full (FAF)
1	TxDMA High	TxFIFO < half full (FAE)
2	eZ80® CPU	
3	RxDMA Low	RxFIFO < half full (FAE)
4	TxDMA Low	TxFIFO > half full (FAF)

TxDMA

The TxDMA module moves the next packet to be transmitted from EMAC memory into the TxFIFO. Whenever the polling timer expires, the TxDMA reads the High status byte from the Tx descriptor table pointed to by the Transmit Read Pointer, TRP. Polling continues until the High status Read reaches bit 7, when the Emac_Owns ownership semaphore, bit 15 of the descriptor table (see Table 178 on page 295) is set to 1. The TxDMA then initializes the packet length counter with the size of the packet from descriptor table bytes 3 and 4. The TxDMA moves the data into the TxFIFO until the packet length counter downcounts to zero. The TxDMA then waits for Transmission Complete signal to be asserted to indicate that the packet is sent and that the Transmit status from the EMAC is valid. The TxDMA updates the descriptor table status and resets the ownership semaphore, bit 15. Finally, the Tx_DONE_STAT bit of the EMAC Interrupt Status Register is set to 1, the address field, DMA_Address, is updated from the descriptor table next pointer, NP (see Figure 62 on page 294). The High byte of the status is read to determine if the next packet is ready to be transmitted.

While the TxDMA is filling the TxFIFO, it monitors two signals from the Transmit FIFO State Machine (TxFifoSM) to detect error conditions and to determine if the packet is to be retransmitted (TxDMA_Retry asserted) or the packet is aborted (TxDMA_Abort asserted). If the packet is aborted, the TxDMA updates the descriptor status and moves to the next packet. If the packet is to be retried, the DMA_Address is reset to the start of the packet, the packet length counter is reloaded from the descriptor table, bytes 3 and 4, and the packet is moved into the TxFIFO again. When an abort or retry event occurs, the TxDMA asserts the appropriate signal to reset the TxFIFO Read and Write pointers which clears out any data that is in the FIFO. The TxFifoSM negates the TxDMA_Abort or TxDMA_Retry signal(s) or both when the TxFCWP signal is High. This handshaking maintains synchronization between the TxDMA and the TxFifoSM.

RxDMA

The RxDMA reads the data from the RxFIFO and stores it in the EMAC memory Receive buffer. When the end of the packet is detected, the RxDMA reads the next two bytes from

the RxFIFO and writes them into the Rx descriptor status LSB and MSB. The packet-length counter is stored into the descriptor table's Packet Length field, and the descriptor table's next pointer is written into the Rx descriptor table. Additionally, the Rx_DONE_STAT bit in the EMAC Interrupt Status Register is set to 1.

Signal Termination

When the EMAC interface is not used, the MII signals must be terminated as listed in Table 175. Terminated pins are either left unconnected (float) or tied to ground.

MDIO is controlled by the MDC output signal. When the EMAC is not being used, these two pins are not driven. The RX_DV, RX_ER, and RXD[3:0] inputs are controlled by the rising edge of the RX_CLK input signal. When RX_CLK is tied to Ground, these pins do not affect the EMAC. The TX_EN, TX_ER, and TXD[3:0] outputs are controlled by the rising edge of the TX_CLK input signal. When TX_CLK is tied to Ground, these pins do not affect the EMAC. The CRS and COL input pins have no relationship to the clock, and therefore must be placed into nonactive states and tied to Ground.

Table 175. MII Signal Termination When EMAC is Not Used

Signal	Pin Type	Termination Direction
MDIO	Bidirectional	Float
MDC	Output pin	Float
RX_DV	Input pin	Float
CRS	Input pin	Ground
RX_CLK	Input pin	Ground
RX_ER	Input pin	Float
RXD[3:0]	Input pins	Float
COL	Input pin	Ground
TX_CLK	Input pin	Ground
TX_EN	Output pin	Float
TXD[3:0]	Output pins	Float
TX_ER	Output pin	Float

EMAC Interrupts

Eight different sources of interrupts from the EMAC are listed in Table 176.

Table 176. EMAC Interrupts

Interrupt	Description
EMAC System Interrupts	
Transmit State Machine Error	Bit 7 (TxFSMERR_STAT) of the EMAC Interrupt Status Register (EMAC_ISTAT). A Transmit State Machine Error must not occur. However, if this bit is set, the entire transmitter module must be reset.
MIIMGT Done	Bit 6 (MGTDONE_STAT) of the Interrupt Status Register (EMAC_ISTAT). This bit is set when communicating to the PHY over the MII during a Read or Write operation.
Receive Overrun	Bit 2 (Rx_OVR_STAT) of the Interrupt Status Register (EMAC_ISTAT). If this bit is set, all incoming packets are ignored until this bit is cleared by software.
EMAC Transmitter Interrupts	
Transmit Control Frame	Transmit Control Frame = Bit 1 (Tx_CF_STAT) of the Interrupt Status Register (EMAC_ISTAT). Denotes when control frame transmission is complete.
Transmit Done	Bit 0 (Tx_DONE_STAT) of the Interrupt Status Register (EMAC_ISTAT). Denotes when packet transmission is complete.
EMAC Receiver Interrupts	
Receive Packet	Bit 5 (Rx_CF_STAT) of the Interrupt Status Register (EMAC_ISTAT). Denotes when packet reception is complete.
Receive Pause Packet	Bit 4 (Rx_PCF_STAT) of the Interrupt Status Register (EMAC_ISTAT). Denotes when pause packet reception is complete.
Receive Done	Bit 3 (Rx_DONE_STAT) of the Interrupt Status Register (EMAC_ISTAT). Denotes when packet reception is complete.

EMAC Shared Memory Organization

Internal Ethernet SRAM shares memory with the CPU. This memory is divided into the Transmit buffer and the Receive buffer by defining three registers, as listed below.

- Transmit Lower Boundary Pointer (TLBP)—this register points to the start of the Transmit buffer in the internal Ethernet shared memory space.
- Boundary Pointer (BP)—this register points to the start of the Receive buffer.

- Receive High Boundary Pointer (RHBP)—this register points to the end of the Receive buffer + 1.

Figure 60 displays the internal Ethernet shared memory.

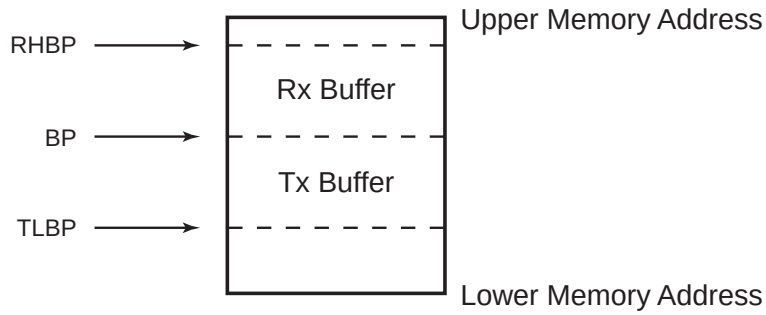


Figure 60. Internal Ethernet Shared Memory

The Transmit and Receive buffers are subdivided into packet buffers of 32, 64, 128, or 256 bytes in size. The packet buffer size is set in bits 7 and 6 of the EmacBufSize register. An Ethernet packet accommodate multiple packet buffers. First, however, a brief listing of the contents of a typical Ethernet packet is in order. See Table 177.

Table 177. Ethernet Packet Contents

Byte Range	Contents
Bytes 0–5	MAC destination address.
Bytes 6–11	MAC source address.
Bytes 12–13	Length/Type field.
Bytes 14–n	MAC Client Data.
Bytes (n+1)–(n+4)	Frame Check Sequence.

At the start of each packet is a descriptor table that describes the packet. Each actual Ethernet packet follows the descriptor table as displayed in Figure 61 on page 294.

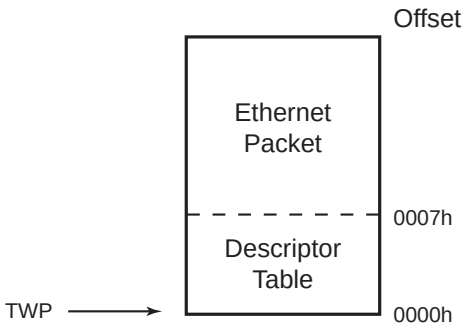


Figure 61. Descriptor Table

Note: *For an official description of an Ethernet packet, refer to IEEE 802.3 specification, Figure 3-1.*

The descriptor table contains three entries: the next pointer (NP), the packet size (Pkt_Size), and the packet status (Stat), as displayed in Figure 62.

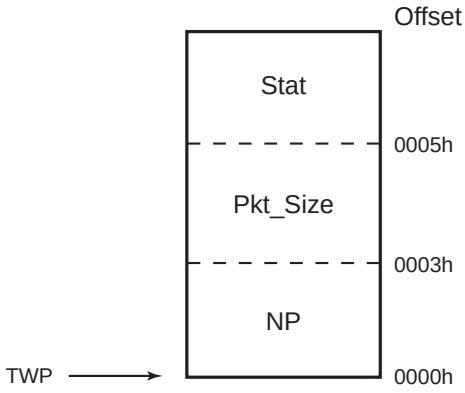


Figure 62. Descriptor Table Entries

NP is a 24-bit pointer to the start of the next packet. Pkt_Size contains the number of bytes of data in the Ethernet packet, including the four CRC bytes, but does not contain the seven descriptor table bytes. Stat contains the status of the packet. Stat differs for Transmit and Receive packets. See Table 178 on page 295 and Table 179 on page 295.

Table 178. Transmit Descriptor Status

Bit	Name	Description
15	TxOwner	0 = Host (eZ80®) owns, 1 = EMAC owns.
14	TxAbort	1 = Packet aborted (not transmitted).
13	TxBPA	1 = Back pressure applied.
12	TxHuge	1 = Packet size is very large (Pkt_Size > EmacMaxf).
11	TxLOOR	1 = Type/Length field is out of range (larger than 1518 bytes).
10	TxLCError	1 = Type/Length field is not a Type field and it does not match the actual data byte length of the Ethernet packet. The data byte length is the number of bytes of data in the Ethernet packet between the Type/Length field and the FCS.
9	TxCrcError	1 = The packet contains an invalid FCS (CRC). This flag is set when CRCEN = 0 and the last 4 bytes of the packet are not the valid FCS.
8	TxPktDeferred	1 = Packet is deferred.
7	TxXsDfr	1 = Packet is excessively deferred. (> 6071 nibble times in 100BaseT or 24,287 bit times in 10BaseT).
6	TxFifoUnderRun	1 = TxFIFO experiences underrun. Check the TxAbort bit to see if the packet is aborted or retried.
5	TxLateCol	1 = A late collision occurs. Collision is detected at a byte count > EmacCfg2[5:0]. Collisions detected before the byte count reaches EmacCfg2[5:0] are early collisions and retried.
4	TxMaxCol	1 = The maximum number of collisions occurs. #Collisions > EmacCfg3[3:0]. These packets are aborted.
[3:0]	TxNumberOfCollisions	This field contains the number of collisions that occur while transmitting the packet.

Table 179. Receive Descriptor Status

Bit	Name	Description
15	RxOK	1 = Packet received intact.
14	RxAlignError	1 = An odd number of nibbles is received.
13	RxCrcError	1 = The CRC (FCS) is in error.

Table 179. Receive Descriptor Status (Continued)

Bit	Name	Description
12	RxLongEvent	1 = A Long or Dropped Event occurs. A Long Event is when a packet over 50,000 bit times occurs. A Dropped Packet occurs if the minimum interpacket gap is not met, the preamble is not pure, and the EmacCfg3[PUREP] bit is set, or if a preamble over 11 bytes in length is detected and the EmacCfg3[LONGP] bit is set to 1.
11	RxPCF	1 = The packet is a pause control frame.
10	RxCF	1 = The packet is a control frame.
9	RxMcPkt	1 = The packet contains a multicast address.
8	RxBcPkt	1 = The packet contains a broadcast address.
7	RxVLAN	1 = The packet is a VLAN packet.
6	RxUOpCode	1 = An unsupported opcode is indicated in the opcode field of the Ethernet packet.
5	RxLOOR	1 = The Type/Length field is out of range (larger than 1518 bytes).
4	RxLCError	1 = Type/Length field is not a Type field and it does not match the actual data byte length of the Ethernet packet. The data byte length is the number of bytes of data in the Ethernet packet between the Type/Length field and the FCS.
3	RxCodeV	1 = A code violation is detected. The PHY asserts Rx error (RxER).
2	RxCEvent	1 = A carrier event is previously seen. This event is defined as Rx error RxER = 1, receive data valid (RxDV) = 0 and receive data (RxD) = Eh.
1	RxDvEvent	1 = A receive data (RxDV) event is previously seen. Indicates that the last Receive event is not long enough to be a valid packet.
0	RxOVR	1 = A Receive overrun occurs in this packet. An overrun occurs when all of the EMAC Receive buffers are in use and the Receive FIFO is full. The hardware ignores all incoming packets until the EmacIStat Register [Rx_Ovr] bit is cleared by the software. There is no indication as to how many packets are ignored.

EMAC and the System Clock

Effective Ethernet throughput in any given system is dependent upon factors such as system clock speed, network protocol overhead, application complexity, and network traffic conditions at any given moment. The following information provides a general guideline about the effects of system clock speed on Ethernet operation.

The eZ80F91 MCU's EMAC block performs a synchronous function that is designed to operate over a wide range of system clock frequencies. To understand its maximum data

transfer capabilities at certain system operating frequencies, you must first understand the internal data bus bandwidth that is required under ideal conditions.

For 10BaseT Ethernet connectivity, the data rate is 10 Mbps, which equates to 1.25 Mbps. If the eZ80F91 MCU is operating in FULL-DUPLEX mode over 10BaseT, the data rate for RX data and TX data is 1.25Mbps. Because raw data transfers at this rate consume a certain amount of CPU bandwidth, the CPU must support traffic from both directions as well as operate at a minimum clock frequency of (1.25 + 1.25) * 2 = 5MHz while transferring Ethernet packets to and from the physical layer.

Similarly, for 100BaseT Ethernet, the data rate is 100 Mbps, which equates to 12.5 Mbps. If the eZ80F91 MCU is operating in FULL-DUPLEX mode over 100BaseT, the data rate for RX data and TX data is 12.5Mbps. Because raw data transfers at this rate consume a certain amount of CPU bandwidth, the CPU must support traffic from both directions as well as operate at a minimum clock frequency of (12.5 + 12.5) x 2 = 50 MHz while transferring Ethernet packets to and from the physical layer. Consequently, 50 MHz is the minimum system clock speed that the eZ80® CPU requires to sustain EMAC data transfers while not including any software overhead or additional eZ80 tasks.

The FIFO functionality of the EMAC operates at any frequency as long as the user application avoids overrun and underrun errors via higher-level flow control. Actual application requirements will dictate Ethernet modes of operation (FULL-DUPLEX, HALF-DUPLEX, etc.). Because each user and application is different, it becomes your responsibility to control the data flow with these parameters. Under ideal conditions, the system clock will operate somewhere between 5 MHz and 50 MHz to handle the EMAC data rates.

EMAC Operation in HALT Modes

When the CPU is in HALT mode, the eZ80F91 device's EMAC block cannot be disabled as other peripherals. Upon receipt of an Ethernet packet, a maskable Receive interrupt is generated by the EMAC block, just as it would be in a non-halt mode. Accordingly, the processor wakes up and continues with the user-defined application.

EMAC Registers

After a system reset, all EMAC registers are set to their default values. Any Writes to unused registers or register bits are ignored and reads return a value of 0. For compatibility with future revisions, unused bits within a register must always be written with a value of 0. Read/Write attributes, reset conditions, and bit descriptions of all of the EMAC registers are provided in this section.

EMAC Test Register

The EMAC Test Register allows test functionality of the EMAC block. Available test modes are defined for bits [6:0]. See Table 180.

Table 180. EMAC Test Register (EMAC_TEST = 0020h)

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

Bit Position	Value	Description
7	0	Reserved.
6 TEST_FIFO	0	FIFO test mode disabled—Normal operation.
	1	FIFO test mode enabled.
5 TxRx_SEL	0	Select the Receive FIFO when FIFO test mode is enabled.
	1	Select the Transmit FIFO when FIFO test mode is enabled.
4 SSTC	0	Normal operation.
	1	Short Cut Slot Timer Counter. Slot time is shortened to speed up simulation.
3 SIMR	0	Normal operation.
	1	Simulation Reset.
2 FRC_OVR_ERR	0	Normal operation.
	1	Force Overrun error in Receive FIFO.
1 FRC_UND_ERR	0	Normal operation.
	1	Force Underrun error in Transmit FIFO.
0 LPBK	0	Normal operation.
	1	EMAC Transmit interface is looped back into EMAC Receive interface.

EMAC Configuration Register 1

The EMAC Configuration Register 1 allows control of the padding, autodetection, cyclic redundancy checking (CRC) control, full-duplex, field length checking, maximum packet ignores, and proprietary header options. See Table 181.

Table 181. EMAC Configuration Register 1 (EMAC_CFG1 = 0021h)

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 PADEN	0	No padding. Assume all frames presented to EMAC have proper length.
	1	EMAC pads all short frames by adding zeroes to the end of the data field. This bit is used in conjunction with ADPADN and VLPAD.
6 ADPADN	0	Disable autodetection.
	1	Enable frame detection by comparing the two bytes following the source address with 0x8100 (VLAN Protocol ID) and pad accordingly. This bit is ignored if PADEN is cleared to 0.
5 VLPAD	0	Do not pad all short frames.
	1	EMAC pads all short frames to 64 bytes and append a valid CRC. This bit is ignored if PADEN is cleared to 0.
4 CRCEN	0	Do not append CRC.
	1	Append CRC to every frame regardless of padding options.
3 FULLD	0	HALF-DUPLEX mode. CSMA/CD is enabled.
	1	Enable FULL-DUPLEX mode. CSMA/CD is disabled.
2 FLCHK	0	Ignore the length field within Transmit/Receive frames.
	1	Both Transmit and Receive frame lengths are compared to the length/type field. If the length/type field represents a length then the frame length check is performed.
1 HUGEN	0	Limit the Receive frame-size to the number of bytes specified in the MAXF[15:0] field.
	1	Allow unlimited sized frames to be received. Ignore the MAXF[15:0] field.

Bit Position	Value	Description
0 DCRCC	0	No proprietary header. Normal operation.
	1	Four bytes of proprietary header, ignored by CRC, exists on the front of IEEE 802.3 frames.

Table 182 lists the results of different settings for bits [7:4] of EMAC Configuration Register 1.

Table 182. CRC/PAD Features of EMAC Configuration Register

ADPADN	VLPADN	PADEN	CRCEN	Result
0	0	0	0	No pad or CRC appended.
0	0	0	1	CRC appended.
0	0	1	0	Pad to 60 bytes if necessary; append CRC (min. size = 64).
0	0	1	1	Pad to 60 bytes if necessary; append CRC (min. size = 64).
0	1	0	0	No pad or CRC appended.
0	1	0	1	CRC appended.
0	1	1	0	Pad to 64 bytes if necessary, append CRC (min. size = 68).
0	1	1	1	Pad to 64 bytes if necessary, append CRC (min. size = 68).
1	0	0	0	No pad or CRC appended.
1	0	0	1	CRC appended.
1	0	1	0	If VLAN not detected, pad to 60, add CRC. If VLAN detected, pad to 64, add CRC.
1	0	1	1	If VLAN not detected, pad to 60, add CRC. If VLAN detected, pad to 64, add CRC.
1	1	0	0	No pad or CRC appended.
1	1	0	1	CRC appended.
1	1	1	0	If VLAN not detected, pad to 60, add CRC. If VLAN detected, pad to 64, add CRC.
1	1	1	1	If VLAN not detected, pad to 60, add CRC. If VLAN detected, pad to 64, add CRC.

EMAC Configuration Register 2

The EMAC Configuration Register 2 controls the behavior of the back pressure and late collision data from the Descriptor table. See Table 183.

Table 183. EMAC Configuration Register 2 (EMAC_CFG2 = 0022h)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	1	1	0	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.

Bit Position	Value	Description
7 BPNB	0	Use normal back-off algorithm prior to transmitting packet. No back pressure applied.
	1	After incidentally causing a collision during back pressure, the EMAC immediately (that is, no back-off) retransmits the packet without back-off, which reduces the chance of further collisions and ensures that the Transmit packets are sent.
6 NOBO	0	Enable exponential back-off.
	1	The EMAC immediately retransmits following a collision rather than use the binary exponential backfill algorithm, as specified in the IEEE 802.3 specification.
[5:0] LCOL	00h–3Fh	Sets the number of bytes after Start Frame Delimiter (SFD) for which a late collision occurs. By default, all late collisions are aborted.

EMAC Configuration Register 3

The EMAC Configuration Register 3 controls preamble length and value, excessive deferment, and the number of retransmission tries. See Table 184.

Table 184. EMAC Configuration Register 3 (EMAC_CFG3 = 0023h)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	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.

Bit Position	Value	Description
7 LONGP	0	The EMAC allows any preamble length as per the IEEE 802.3 specification.*
	1	The EMAC only allows Receive packets that contain preamble fields less than 12 bytes in length.*
6 PUREP	0	No preamble error checking is performed.
	1	The EMAC verifies the content of the preamble to ensure that it contains a value of 55h and that it is error-free. Packets containing an errored preamble are discarded.
5 XSDFR	0	The EMAC aborts when the excessive deferral limit is reached.
	1	The EMAC defers to the carrier indefinitely as per the IEEE 802.3 specification.
4 BITMD	0	Disable 10 Mbps ENDEC mode.
	1	Enable 10 Mbps ENDEC mode.
[3:0] RETRY	0h–Fh	A programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions.

Note: IEEE 802.3 specifies a minimum of 56 bits of preamble. A maximum number of bits is not defined. For details, see the IEEE 802.3 Specification, Section 7.2.3.2.

EMAC Configuration Register 4

The EMAC Configuration Register 4 controls pause control frame behavior, back pressure, and receive frame acceptance. See Table 185.

Table 185. EMAC Configuration Register 4 (EMAC_CFG4 = 0024h)

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	0	Reserved.
6 TPCF	0	Do not transmit a pause control frame.
	1	Transmit pause control frame (FULL-DUPLEX mode). TPCF continually sends pause control frames until negated.
5 THDF	0	Disable back pressure.
	1	EMAC asserts back pressure on the link. Back pressure causes preamble to be transmitted, raising carrier sense (HALF-DUPLEX mode).
4 PARF	0	Only accept frames that meet preset criteria (that is, address, CRC, length, etc.).
	1	All frames are received regardless of address, CRC, length, etc.
3 RxFC	0	EMAC ignores received pause control frames.
	1	EMAC acts upon pause control frames received.
2 TxFC	0	PAUSE control frames are not allowed to be transmitted.
	1	PAUSE control frames are allowed to be transmitted.
1 TPAUSE	0	Do not force a pause condition.
	1	Force a pause condition while this bit is asserted.
0 RxEN	0	EMAC receiver disabled.
	1	EMAC receiver enabled.

EMAC Station Address Register

The EMAC Station Address register is used for two functions. In the address recognition logic for Receive frames, EMAC_STAD_0–EMAC_STAD_5 are matched against the sixth byte Destination Address (DA) field of the Receive frame. EMAC_STAD_0 is matched against the first byte of the Receive frame, and EMAC_STAD_5 is matched against the sixth byte of the Receive frame. Bit 0 of EMAC_STAD_0 (STAD[40]) is matched against the first bit (Unicast/Multicast bit) of the first byte of the Receive frame. This bit ordering is used to logically map the PE-MACMII station address as illustrated below.

EMAC_STAD0[7:0] contains STAD[47:40]

....

....

EMAC_STAD5[7:0] contains STAD[7:0]

The second function of the EMAC Station Address registers is to provide the Source Address (SA) field of Transmit Pause frames when these frames are transmitted by the EMAC. EMAC_STAD_0 provides the first byte of the 6 byte SA field and EMAC_STAD_5 provides the final byte of the SA field in order of transmission. The LSB is the first byte sent out. The EMAC Station Address register is listed in Table 186.

Table 186. EMAC Station Address Register (EMAC_STAD_0 = 0025h, EMAC_STAD_1 = 0026h, EMAC_STAD_2 = 0027h, EMAC_STAD_3 = 0028h, EMAC_STAD_4 = 0029h, EMAC_STAD_5 = 002Ah)

Bit	7	6	5	4	3	2	1	0
EMAC_STAD_0 Reset	0	0	0	0	0	0	0	0
EMAC_STAD_1 Reset	0	0	0	0	0	0	0	0
EMAC_STAD_2 Reset	0	0	0	0	0	0	0	0
EMAC_STAD_3 Reset	0	0	0	0	0	0	0	0
EMAC_STAD_4 Reset	0	0	0	0	0	0	0	0
EMAC_STAD_5 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] EMAC_STAD_x	00h–FFh	This 48-bit station address comprises {EMAC_STAD_5, EMAC_STAD_4, EMAC_STAD_3, EMAC_STAD_2, EMAC_STAD_1, EMAC_STAD_0}.

EMAC Transmit Pause Timer Value Register—Low and High Bytes

The Low and High bytes of the EMAC Transmit Pause Timer Value Register are inserted into outgoing pause control frames. See Table 187 and Table 188.

Table 187. EMAC Transmit Pause Timer Value Register—Low Byte (EMAC_TPTV_L = 002Bh)

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] EMAC_TPTV_L	00h–FFh	The 16-bit value, {EMAC_TPTV_H, EMAC_TPTV_L}, is inserted into outgoing pause control frames as the pause timer value upon asserting TPCF.

Table 188. EMAC Transmit Pause Timer Value Register—High Byte (EMAC_TPTV_H = 002Ch)

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] EMAC_TPTV_H	00h–FFh	The 16-bit value, {EMAC_TPTV_H, EMAC_TPTV_L}, is inserted into outgoing pause control frames as the pause timer value upon asserting TPCF.

EMAC Interpacket Gap

EMAC Interpacket Gap Overview

Interpacket Gap (IPG) is measured between the last nibble of the frame check sequence (FCS) and the first nibble of the preamble of the next packet. Three registers are available to fine tune the IPG, the EMAC_IPGT, EMAC_IPGR1, and the EMAC_IPGR2. The first register EMAC_IPGT determines the back-to-back Transmit IPG. The other two registers determine the non-back-to-back IPG in two parts. Table 189 lists the values for the EMAC_IPGT and the corresponding IPGs for both FULL-DUPLEX and HALF-DUPLEX modes.

Table 189. EMAC_IPGT Back-to-Back Settings for Full- and Half-Duplex Modes

MII, RMII/SMII, PMD (100 Mbps)			MII, RMII/SMII (10 Mbps)			ENDEC Mode (10 Mbps)		
Clock Period = 40 ns IPGT[6:0]			Clock Period = 400 ns IPGT[6:0]			Clock Period = 100 ns IPGT[6:0]		
Half Duplex	Full Duplex	Interpacket Gap	Half Duplex	Full Duplex	Interpacket Gap	Half Duplex	Full Duplex	Interpacket Gap
	0Dh	0.12 µs		00h	1.2 µs		10h	1.9 µs
	0Bh	0.44 µs		08h	4.4 µs		18h	2.7 µs
	0Ch	0.60 µs		0Ch	6.0 µs		20h	3.5 µs
	10h	0.76 µs		10h	7.5 µs		40h	6.7 µs
*12h	15h	0.96 µs	12h	15h	9.6 µs	5Ah	5Dh	9.6 µs
	20h	1.40 µs		20h	14.0 µs		20h	13.0 µs

Note: *The IEEE 802.3, 802.3(u) minimum values are shaded.

The equations for back-to-back Transmit IPG are determined by the following:

FULL-DUPLEX Mode (3 clocks + IPGT clocks) * clock period = IPG

HALF-DUPLEX Mode (6 clocks + IPGT clocks) * clock period = IPG

Table 190 on page 307 lists the IPGR2 settings for the non-back-to-back packets.

Table 190. EMAC_IPGT Non-Back-to-Back Settings for Full- /Half-Duplex Modes

MII, RMII/SMII, PMD (100 Mbps)		MII, RMII/SMII (10 Mbps)		ENDEC Mode (10 Mbps)	
Clock Period = 40 ns		Clock Period = 400 ns		Clock Period = 100 ns	
IPGR2[6:0]	Interpacket Gap	IPGR2[6:0]	Interpacket Gap	IPGR2[6:0]	Interpacket Gap
00h	0.24 µs	00h	2.4 µs	00h	0.6 µs
10h	0.88 µs	10h	8.8 µs	10h	2.2 µs
*12h	0.96 µs	12h	9.6 µs	20h	3.8 µs
20h	1.52 µs	20h	15.2 µs	40h	7.0 µs
40h	2.80 µs	40h	28.0 µs	5Ah	9.6 µs
7Fh	5.32 µs	7Fh	53.2 µs	7Fh	13.3 µs

Note: *The IEEE 802.3, 802.3(u) minimum values are shaded.

A non-back-to-back Transmit IPG is determined by the following formula:

(6 clocks + IPGR2 clocks) * clock period = IPG

The difference in values between Table 189 on page 306 and Table 190 is due to the asynchronous nature of the Carrier Sense (CRS). The CRS must undergo a 2-clock synchronization before the internal Tx state machine detects it. This synchronization equates to a 6-clock intrinsic delay between packets instead of the 3-clock intrinsic delay in the back-to-back packet mode. More information covering this topic is found in the IEEE 802.3/4.2.3.2.1 Carrier Deference section.

EMAC Interpacket Gap Register

The EMAC Interpacket Gap (IPG) is a programmable field representing the IPG between back-to-back packets. It is the IPG parameter used in FULL-DUPLEX and HALF-DUPLEX modes between back-to-back packets. Set this field to the appropriate number of IPG bytes. The default setting of 15h represents the minimum IPG of 0.96 µs (at 100 Mbps) or 9.6 µs (at 10 Mbps). See Table 191.

Table 191. EMAC Interpacket Gap Register (EMAC_IPGT = 002Dh)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	1	0	1	0	1
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	0	Reserved.
[6:0] IPGT	00h–7Fh	The number of bytes of IPG.

EMAC Non-Back-To-Back IPG Register—Part 1

Part 1 of the EMAC non-back-to-back IPG Register is a programmable field representing the optional carrier sense window referenced in IEEE 802.3/4.2.3.2.1 Carrier Deference. If a carrier is detected during the timing of IPGR1, the EMAC defers to the carrier. If, however, the carrier becomes active after IPGR1, the EMAC continues timing for IPGR2 and transmits, knowingly causing a collision. This collision acts to ensure fair access to the medium. Its range of values is 00h to IPGR2. See Table 192. The default setting of 0Ch represents the Carrier Sense Window Referencing depicted tin IEEE 802.3, Section 4.2.3.2.1.

Table 192. EMAC Non-Back-To-Back IPG Register—Part 1 (EMAC_IPGR1 = 002Eh)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	1	1	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	Reserved.
[6:0] IPGR 1	00h–7Fh	This is a programmable field representing the optional carrier sense window referenced in IEEE 802.3/4.2.3.2.1 Carrier Deference.

EMAC Non-Back-To-Back IPG Register—Part 2

Part 2 of the EMAC non-back-to-back IPG Register is a programmable field representing the non-back-to-back IPG. Its default is 12h, which represents the minimum IPG of 0.96 µs at 100 Mbps or 9.6 µs at 10 Mbps. SeeTable 193 on page 309.

Table 193. EMAC Non-Back-To-Back IPG Register—Part 2 (EMAC_IPGR2 = 002Fh)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	1	0	0	1	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	0	Reserved.
[6:0] IPGR2	00h–7Fh	This bit range is a programmable field representing the non-back-to-back interpacket gap.

EMAC Maximum Frame Length Register—Low and High Bytes

The 16-bit field resets to 0600h, which represents a maximum Receive frame of 1536 bytes. An untagged maximum size Ethernet frame (packet) is 1518 bytes. A tagged frame adds four bytes for a total of 1522 bytes. If a shorter maximum length restriction is more appropriate, program this field. See Table 194 and Table 195 on page 310.

Note: *The default value of 1536 bytes is large enough to cover the largest Ethernet packet, which contains 14 bytes of Ethernet header, 1500 bytes of MAC client data, plus 4 bytes of CRC for a total of 1518 maximum bytes. This value is also large enough to cover VLAN frames with prepended headers up to 18 bytes.*

VLAN frames have a proprietary header prepended to the Ethernet packet. Setting the DCRCC bit in EMAC_CFG1 will exclude the first 4 bytes—the proprietary header—from the CRC calculation. For VLAN packets, the maximum frame length is 1522, 4 more than for normal Ethernet packets due to the 4 byte prepended header. Normal packets feature a 12 byte header before the MAC client data. For more information about this topic, refer to Figure 3-1 of the IEEE 802.3 specification.

If a proprietary header is allowed, this field must be adjusted accordingly. For example, if 12 byte headers are prepended to frames, MAXF must be set to 1524 bytes to allow the maximum VLAN tagged frame plus the 12 byte header. The default value of 1536 is large enough to cover the largest Ethernet packet: 14 bytes of Ethernet header, 1500 bytes of MAC client data, plus 4 bytes of CRC for a total of 1518 bytes maximum. It is also large enough to cover VLAN packets with prepended headers up to 18 bytes. The following formulas illustrate:

Ethernet Packet— Maximum frame size = normal Ethernet packet – 14 (Ethernet header) + 1500 (MAC client data) + 4 (CRC) = 1518 bytes

VLAN Packet— Maximum frame size = VLAN with 4 byte header – 4 (VLAN header) + 14 (Ethernet header) + 1500 MAC client data) + 4 (CRC) = 1522 bytes.

Table 194. EMAC Maximum Frame Length Register—Low Byte (EMAC_MAXF_L = 0030h

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] EMAC_MAXF_L	00h–FFh	These bits represent the Low byte of the 2 byte MAXF value, {EMAC_MAXF_H, EMAC_MAXF_L}. Bit 7 is bit 7 of the 16-bit value. Bit 0 is bit 0 (lsb) of the 16-bit value.

Table 195. EMAC Maximum Frame Length Register—High Byte (EMAC_MAXF_H = 0031h)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	1	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] EMAC_MAXF_H	00h–FFh	These bits represent the High byte of the 2 byte MAXF value, {EMAC_MAXF_H, EMAC_MAXF_L}. Bit 7 is bit 15 (msb) of the 16-bit value. Bit 0 is bit 8 of the 16-bit value.

EMAC Address Filter Register

The EMAC Address Filter Register functions as a filter to control Promiscuous mode, and multicast and broadcast messaging. See Table 196.

Table 196. EMAC Address Filter Register (EMAC_AFR = 0032h)

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]	0h	Reserved.
3 PROM	1	Enable Promiscuous Mode. Receive all incoming packets regardless of station address. Disables station address filtering.
	0	Disable Promiscuous Mode.
2 MC	1	Accept any multicast message. A multicast packet is determined by the first bit in the destination address. If the first LSB is a 1, it is a group address and is globally or locally administered depending on the 2nd bit. For more information, see IEEE 802.3/3.2.3.
	0	Do not accept multicast messages of any type.
1 QMC	1	Accept only qualified multicast (QMC) messages as determined by the hash table.
	0	Do not accept QMC messages.
0 BC	1	Accept broadcast messages. Broadcast messages have the destination address set to FFFFFFFFFFFFh.
	0	Do not accept broadcast messages.

EMAC Hash Table Register

The EMAC Hash Table Register represents the 8x8 hash table matrix. This table is used as an option to select between different multicast addresses. If a multicast address is received, the first 6 bits of the CRC are decoded and added to a table that points to a single bit within the hash table matrix. If the selected bit = 1, the multicast packet is accepted. If the bit = 0, the multicast packet is rejected. See Table 197.

Table 197. EMAC Hash Table Register (EMAC_HTBL_0 = 0033h, EMAC_HTBL_1 = 0034h, EMAC_HTBL_2 = 0035h, EMAC_HTBL_3 = 0036h, EMAC_HTBL_4 = 0037h, EMAC_HTBL_5 = 0038h, EMAC_HTBL_6 = 0039h, EMAC_HTBL_7 = 003Ah)

Bit	7	6	5	4	3	2	1	0
EMAC_HTBL_0 Reset	0	0	0	0	0	0	0	0
EMAC_HTBL_1 Reset	0	0	0	0	0	0	0	0
EMAC_HTBL_2 Reset	0	0	0	0	0	0	0	0
EMAC_HTBL_3 Reset	0	0	0	0	0	0	0	0
EMAC_HTBL_4 Reset	0	0	0	0	0	0	0	0
EMAC_HTBL_5 Reset	0	0	0	0	0	0	0	0
EMAC_HTBL_6 Reset	0	0	0	0	0	0	0	0
EMAC_HTBL_7 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] EMAC_HTBL_x	00h–FFh	This field is the hash table. The 64 bit hash table is {EMAC_HTBL_7, EMAC_HTBL_6, EMAC_HTBL_5, EMAC_HTBL_4, EMAC_HTBL_3, EMAC_HTBL_2, EMAC_HTBL_1, EMAC_HTBL_0}.

EMAC MII Management Register

The EMAC MII Management Register is used to control the external PHY attached to the MII. See Table 198.

Table 198. EMAC MII Management Register (EMAC_MIIMGT = 003Bh)

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 LCTLD	1	Rising edge causes the CTLD control data to be transmitted to external PHY if MII is not busy. This bit is self clearing.
	0	No operation.
6 RSTAT	1	Rising edge causes status to be read from external PHY via PRSD[15:0] bus if MII is not busy. This bit is self clearing.
	0	No operation.
5 SCINC	1	Scan PHY address increments upon SCAN cycle. The SCAN bit must also be set for the PHY address to increment after each scan. The scanning starts at the EMAC_FIAD and increments up to 1Fh. It then returns to the EMAC_FIAD address.
	0	Normal operation.
4 SCAN	1	Perform continuous Read cycles via MII management. While in SCAN mode, the EMAC_ISTAT[MGTDONE] bit is set when the current PHY Read has completed. At this time, the EMAC_PRSD register holds the Read data and the EMAC_MIISTAT[4:0] holds the address of the PHY for which the EMAC_PRSD data pertains.
	0	Normal operation.
3 SPRE	1	Suppress the MDO preamble. MDO is management data output, an internal signal driven from the MDIO pin.
	0	Normal preamble.

Bit Position	Value	Description
[2:0] CLKS	Programmable divisor that produces MDC from SCLK. MDC is the management data clock pin, which clocks MDIO data to and from the PHY. Its frequency is SCLK divided by the MDC clock divider.	
	000	MDC = SCLK ÷ 4.
	001	MDC = SCLK ÷ 4.
	010	MDC = SCLK ÷ 6.
	011	MDC = SCLK ÷ 8.
	100	MDC = SCLK ÷ 10.
	101	MDC = SCLK ÷ 14.
	110	MDC = SCLK ÷ 20.
	111	MDC = SCLK ÷ 28.

EMAC PHY Configuration Data Register—Low and High Byte

The Low and High bytes of the EMAC PHY Configuration Data Register represents the configuration data written to the external PHY. The EMAC_CTLD_H and EMAC_CTLD_L registers form a 16-bit register. These registers are loaded with data to be sent via the MDIO pin to the PHY. The PHY is selected by setting the EMAC_FIAD. The register inside the PHY is selected by setting EMAC_RGAD. See Table 199 and Table 200 on page 315.

Table 199. EMAC PHY Configuration Data Register—Low Byte (EMAC_CTLD_L = 003Ch)

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] EMAC_CTLD_L	00h–FFh	These bits represent the Low byte of the 2 byte PHY configuration data value, {EMAC_CTLD_H, EMAC_CTLD_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 16 bit value.

Table 200. EMAC PHY Configuration Data Register—High Byte (EMAC_CTLD_H = 003Dh)

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] EMAC_CTLD_H	00h–FFh	These bits represent the High byte of the 2 byte PHY configuration data value, {EMAC_CTLD_H, EMAC_CTLD_L}. Bit 7 is bit 15 (msb) of the 16 bit value. Bit 0 is bit 8 of the 16 bit value.

EMAC PHY Address Register

The EMAC PHY Address Register allows access to the external PHY registers. See Table 201.

Table 201. EMAC PHY Address Register (EMAC_RGAD = 003Eh)

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]	000	Reserved.
[4:0] RGAD	00h–1Fh	Programmable 5 bit value which selects address within the selected external PHY.

EMAC PHY Unit Select Address Register

The EMAC PHY Unit Select Address Register allows the selection of multiple connected external PHY devices. See Table 202.

Table 202. EMAC PHY Unit Select Address Register (EMAC_FIAD = 003Fh)

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]	000	Reserved.
[4:0] FIAD	00h–1Fh	Programmable 5-bit value that selects an external PHY.

EMAC Transmit Polling Timer Register

This register sets the Transmit Polling Period in increments of TPTMR = SYSCLK ÷ 256. Whenever this register is written, the status of the Transmit Buffer Descriptor is checked to determine if the EMAC owns the Transmit buffer. It then rechecks this status every TPTMR (calculated by TPTMR x EMAC_PTMR[7:0]). The Transmit Polling Timer is disabled if this register is set to 00h (which also disables the transmitting of packets). If a transmission is in progress when EMAC_PTMR is set to 00h, the transmission will complete. See Table 203.

Table 203. EMAC Transmit Polling Timer Register (EMAC_PTMR = 0040h)

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] EMAC_PTMR	00h–FFh	The Transmit polling period.

EMAC Reset Control Register

The bit values in the EMAC Reset Control Register are not self-clearing bits. You are responsible for controlling their state. See Table 204.

Table 204. EMAC Reset Control Register (EMAC_RST = 0041h)

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

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

Bit Position	Value	Description
[7:6]	00	Reserved
5 SRST	1	Software Reset Active—resets Receive, Transmit, EMAC Control and EMAC MII_MGT functions
	0	Normal operation
4 HRTFN	1	Reset Transmit function
	0	Normal operation
3 HRRFN	1	Reset Receive function
	0	Normal operation
2 HRTMC	1	Reset EMAC Transmit Control function
	0	Normal operation
1 HRRMC	1	Reset EMAC Receive Control function
	0	Normal operation
0 HRMGT	1	Reset EMAC Management function
	0	Normal operation

EMAC Transmit Lower Boundary Pointer Register—Low and High Bytes

The EMAC Transmit Lower Boundary Pointer is set to the start of the Transmit buffer in EMAC shared memory. See Table 205 and Table 206.

Table 205. EMAC Transmit Lower Boundary Pointer Register—Low Byte (EMAC_TLBP_L = 0042h)

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	R	R	R

Note: R/W = Read/Write.

Bit Position	Value	Description
[7:0] EMAC_TLBP_L	00h–FFh	These bits represent the Low byte of the 2 byte Transmit Lower Boundary Pointer value, {EMAC_TLBP_H, EMAC_TLBP_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 16 bit value.

Table 206. EMAC Transmit Lower Boundary Pointer Register—High Byte (EMAC_TLBP_H = 0043h)*

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

Note: R/W = Read/Write.

Bit Position	Value	Description
[7:0] EMAC_TLBP_H	00h–FFh	These bits represent the High byte of the 2 byte Transmit Lower Boundary Pointer value, {EMAC_TLBP_H, EMAC_TLBP_L}. Bit 7 is bit 15 (msb) of the 16 bit value. Bits 7:5 default to 000 on reset; bit 0 is bit 8 of the 16-bit value.

Note: *Bits 7:5 are not used by the EMAC; these bits return 000.

EMAC Boundary Pointer Register—Low and High Bytes

The Boundary Pointer is set to the start of the Receive buffer (end of Transmit buffer +1) in EMAC shared memory. This pointer is 24 bits and determined by {RAM_ADDR_U, EMAC_BP_H, EMAC_BP_L}. The upper 3 bits of the EMAC_BP_H register are hard-wired inside the eZ80F91 device to locate the base of EMAC shared memory. The last 5 bits of the EMAC_BP_L register value are hard-wired to keep the addressing aligned to a 32 byte boundary. See Table 207 and Table 208.

Table 207. EMAC Boundary Pointer Register—Low Byte (EMAC_BP_L = 0044h)

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	R	R	R

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

Bit Position	Value	Description
[7:0] EMAC_BP_L	00h–FFh	These bits represent the Low byte of the 3 byte EMAC Boundary Pointer value, {EMAC_BP_U, EMAC_BP_H, EMAC_BP_L}. Bit 7 is bit 7 of the 24 bit value. Bit 0 is bit 0 of the 24 bit value.

Table 208. EMAC Boundary Pointer Register—High Byte (EMAC_BP_H = 0045h)

Bit	15:13			12:8				
Reset	1	1	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
[7:0] EMAC_BP_H	00h–FFh	These bits represent the High byte of the 3 byte EMAC Boundary Pointer value, {EMAC_BP_U, EMAC_BP_H, EMAC_BP_L}. Bit 7 is bit 15 of the 24 bit value. Bit 0 is bit 8 of the 24 bit value.

EMAC Boundary Pointer Register—Upper Byte

The EMAC Boundary Pointer Register maps directly to the RAM_ADDR_U register within the eZ80F91 device. This register value is Read Only. See Table 209 on page 320.

Table 209. EMAC Boundary Pointer Register—Upper Byte (EMAC_BP_U = 0046h)

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

Note: R = Read Only.

Bit Position	Value	Description
[7:0] EMAC_BP_U	00h–FFh	These bits represent the upper byte of the 3 byte EMAC Boundary Pointer value, {EMAC_BP_U, EMAC_BP_H, EMAC_BP_L}. Bit 7 is bit 23 of the 24 bit value. Bit 0 is bit 16 of the 24 bit value.

EMAC Receive High Boundary Pointer Register—Low and High Bytes

The Receive High Boundary Pointer Register must be set to the end of the Receive buffer +1 in EMAC shared memory. This RHBP uses the same RAM_ADDR_U as the EMAC_BP_U pointer above. See Table 210 and Table 211 on page 321.

Table 210. EMAC Receive High Boundary Pointer Register—Low Byte (EMAC_RHBP_L = 0047h)

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	R	R	R

Note: R = Read Only, R/W = Read/Write

Bit Position	Value	Description
[7:0] EMAC_RHBP_L	00h–E0h	These bits represent the Low byte of the 2 byte EMAC Receive High Boundary Pointer value, {EMAC_RHBP_H, EMAC_RHBP_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 16 bit value.

Table 211. EMAC Receive High Boundary Pointer Register—High Byte (EMAC_RHBP_H = 0048h)

Bit	7	6	5	4	3	2	1	0
Reset	1	1	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
[7:0] EMAC_RHBP_H	00h–FFh	These bits represent the High byte of the 2 byte EMAC Receive High Boundary Pointer value, {EMAC_RHBP_H, EMAC_RHBP_L}. Bit 7 is bit 15 (msb) of the 16 bit value. Bit 0 is bit 8 of the 16 bit value.

Note: *Bits 7:5 are not used by the EMAC; these bits return 000 upon reset.

EMAC Receive Read Pointer Register—Low and High Bytes

The Receive Read Pointer Register must be initialized to the EMAC_BP value (start of the Receive buffer). This register points to where the next Receive packet is read from. The EMAC_BP[12:5] is loaded into this register whenever the EMAC_RST [(HRRFN) is set to 1. The RxDMA block uses the Emac_Rrp[12:5] to compare to EmacRwp[12:5] for determining how many buffers remain. The result equates to the EmacBlksLeft register. See Table 212 and Table 213 on page 322.

Table 212. EMAC Receive Read Pointer Register—Low Byte (EMAC_RRP_L = 0049h)

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	R	R	R

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

Bit Position	Value	Description
[7:0] EMAC_RRP_L	00h–FFh	These bits represent the Low byte of the 2 byte EMAC Receive Read Pointer value, {EMAC_RRP_H, EMAC_RRP_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 16 bit value.

Table 213. EMAC Receive Read Pointer Register—High Byte (EMAC_RRP_H = 004Ah)

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
[7:0] EMAC_RRP_H	00h–FFh	These bits represent the High byte of the 2-byte EMAC Receive Read Pointer value, {EMAC_RRP_H, EMAC_RRP_L}. Bit 7 is bit 15 (msb) of the 16-bit value. Bits 7:5 default to 000 on reset; bit 0 is bit 8 of the 16-bit value.

EMAC Buffer Size Register

The lower six bits of this register set the level at which the EMAC either transmits a pause control frame or jams the Ethernet bus, depending on the mode selected. When each of these bits contain a zero, this feature is disabled.

In FULL-DUPLEX mode, a Pause Control Frame is transmitted as a One-shot operation. The software must free up a number of Rx buffers so that the number of buffers remaining, EmacBlksLeft, is greater than TCPF_LEV.

In HALF-DUPLEX mode, the EMAC jams the Ethernet by sending a continuous stream of hexadecimal 5s (`5fh`). When the software frees up the Rx buffers and the number of buffers remaining, EmacBlksLeft, is greater than TCPF_LEV, the EMAC stops jamming.

Table 214. EMAC Buffer Size Register (EMAC_BUFSZ = 004Bh)

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] BUFSZ	00	Set EMAC Rx/Tx buffer size to 256 bytes.
	01	Set EMAC Rx/Tx buffer size to 128 bytes.
	10	Set EMAC Rx/Tx buffer size to 64 bytes.
	11	Set EMAC Rx/Tx buffer size to 32 bytes.
[5:0] TPCF_LEV	00h–3Fh	Transmit Pause Control Frame level. 00h disables the hardware generated transmit pause control frame.

EMAC Interrupt Enable Register

Enabling the Receive Overrun interrupt allows software to detect an overrun condition as soon as it occurs. If this interrupt is not set, then an overrun cannot be detected until the software processes the Receive packet with the overrun and checks the Receive status in the Rx descriptor table. Because the receiver is disabled by an overrun error until the Rx_OVR bit is cleared in the EMAC_ISTAT register, this packet is the final packet in the Receive buffer. To re-enable the receiver before all of the Receive packets are processed and the Receive buffer is empty, software enables this interrupt to detect the overrun condition early. As it processes the Receive packets, it re-enables the receiver when the number of free buffers is greater than the number of minimum buffers. See Table 215 on page 324.

Table 215. EMAC Interrupt Enable Register (EMAC_IEN = 004Ch)

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 TxFSMERR	1	Enable Transmit State Machine Error Interrupt (system interrupt).
	0	Disable Transmit State Machine Error Interrupt (system interrupt).
6 MGTDONE	1	Enable MII Management. Done Interrupt (system Interrupt).
	0	Disable MII Management. Done Interrupt (system Interrupt).
5 Rx_CF	1	Enable Receive Control Frame Interrupt (Receive interrupt).
	0	Disable Receive Control Frame Interrupt (Receive interrupt).
4 Rx_PCF	1	Enable Receive Pause Control Frame interrupt (Receive interrupt).
	0	Disable Receive Pause Control Frame interrupt (Receive interrupt).
3 Rx_DONE	1	Enable Receive Done interrupt (Receive interrupt).
	0	Disable Receive Done interrupt (Receive interrupt).
2 Rx_OVR	1	Enable Receive Overrun interrupt (System interrupt).
	0	Disable Receive Overrun interrupt (System interrupt).
1 Tx_CF	1	Enable Transmit Control Frame Interrupt (Transmit interrupt).
	0	Disable Transmit Control Frame Interrupt (Transmit interrupt).
0 Tx_DONE	1	Enable Transmit Done interrupt (Transmit interrupt).
	0	Disable Transmit Done Interrupt (Transmit interrupt).

EMAC Interrupt Status Register

When a Receive overrun occurs, all incoming packets are ignored until the Rx_OVR_STAT status bit is cleared by software. Consequently, software controls when the receiver is re-enabled after an overrun. Enable the Rx_OVR interrupt to detect overrun conditions when they occur. Clear this condition when the Rx buffers are freed to avoid additional overrun errors. See Table 216.

Note: *Status bits are not self-clearing. Each status bit is cleared by writing a 1 into the selected bit.*

Table 216. EMAC Interrupt Status Register (EMAC_ISTAT = 004Dh)

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 TxFSMERR_STAT	1	An internal error occurs in the EMAC Transmit path. The Transmit path must be reset to reset this error condition.
	0	Normal operation—no Transmit state machine errors.
6 MGTDONE_STAT	1	The MII Management interrupt has completed a Read (RSTAT or SCAN) or a Write (LDCTLD) access to the PHY.
	0	The MII Management interrupt does not occur.
5 Rx_CF_STAT	1	Receive Control Frame interrupt (Receive Interrupt) occurs.
	0	Receive Control Frame interrupt does not occur.
4 Rx_PCF_STAT	1	Receive Pause Control Frame interrupt (Receive Interrupt) occurs.
	0	Disable Receive Pause Control Frame interrupt (Receive Interrupt) does not occur.
3 Rx_DONE_STAT	1	Receive Done interrupt (Receive Interrupt) occurs.
	0	Disable Receive Done interrupt (Receive Interrupt) does not occur.
2 Rx_OVR_STAT	1	Receive Overrun interrupt (System Interrupt) occurs.
	0	Receive Overrun interrupt (System Interrupt) does not occur.

Bit Position	Value	Description
1 Tx_CF_STAT	1	Transmit Control Frame Interrupt (Transmit Interrupt) occurs.
	0	Transmit Control Frame Interrupt (Transmit Interrupt) does not occur.
0 Tx_DONE_STAT	1	Transmit Done interrupt (Transmit Interrupt) occurs.
	0	Transmit Done interrupt (Transmit Interrupt) does not occur.

EMAC PHY Read Status Data Register—Low and High Bytes

The PHY MII Management Data Register is where the data Read from the PHY is stored. See Table 217 and Table 218 on page 327.

Table 217. EMAC PHY Read Status Data Register—Low Byte (EMAC_PRSD_L = 004Eh)

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] EMAC_PRSD_L	00h–FFh	These bits represent the Low byte of the 2 byte EMAC PHY Read Status Data value, {EMAC_PRSD_H, EMAC_PRSD_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 16 bit value.

Table 218. EMAC PHY Read Status Data Register—High Byte (EMAC_PRSD_H = 004Fh)

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] EMAC_PRSD_H	00h–FFh	These bits represent the High byte of the 2-byte EMAC PHY Read Status Data value, {EMAC_PRSD_H, EMAC_PRSD_L}. Bit 7 is bit 15 (msb) of the 16-bit value. Bit 0 is bit 8 of the 16-bit value.

EMAC MII Status Register

The EMAC MII Status Register is used to determine the current state of the external PHY device. See Table 219.

Table 219. EMAC MII Status Register (EMAC_MIISTAT = 0050h)

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 BUSY	1	MII management operation in progress—Busy. This status bit goes busy whenever the LCTLD (PHY Write) or the RSTAT (PHY Read) is set in the EMAC_MIIMGT register. It is negated when the Write or Read operation to the PHY has completed. In SCAN mode, the BUSY will be asserted until the SCAN is disabled. Use the EmacIStat[MGTDONE] interrupt status bit to determine when the data is valid.
	0	Not Busy.
6 MIILF	1	Local copy of PHY Link fail bit.
	0	PHY Link OK.

5 NVALID	1	MII Scan result is not valid Emac_PRSD is invalid
	0	Emac_PRSD is valid.
[4:0] RDADR	00h–1Fh	Denotes PHY addressed in current scan cycle.

EMAC Receive Write Pointer Register—Low Byte

The Read Only Receive-Write-Pointer register reports the current RxDMA Receive Write pointer. This pointer gets initialized to EmacTLBP whenever Emac_RST bits SRST or HRRTN are set. Because the size of the packet is limited to a minimum of 32 bytes, the last five bits are always zero. See Table 220 and Table 221 on page 329.

Table 220. EMAC Receive Write Pointer Register—Low Byte (EMAC_RWP_L = 0051h)

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] EMAC_RWP_L	00h–E0h	These bits represent the Low byte of the 2 byte EMAC RxDMA Receive Write Pointer value, {EMAC_RWP_H, EMAC_RWP_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 16 bit value.

EMAC Receive Write Pointer Register—High Byte

Because of the size of the EMAC's 8 KB SRAM, the upper three bits of the EMAC Receive Write Pointer Register are always zero.

Table 221. EMAC Receive Write Pointer Register—High Byte (EMAC_RWP_H = 0052h)

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] EMAC_RWP_H	00h–1Fh	These bits represent the High byte of the 2 byte EMAC RxDMA Receive Write Pointer value, {EMAC_RWP_H, EMAC_RWP_L}. Bit 7 is bit 15 (msb) of the 16 bit value. Bit 0 is bit 8 of the 16 bit value.

EMAC Transmit Read Pointer Register—Low Byte

The Low byte of the Transmit Read Pointer register reports the current TxDMA Transmit Read pointer.This pointer is initialized to EmacTLBP whenever Emac_RST bits SRST or HRRTN are set. Because the size of the packet is limited to a minimum of 32 bytes, the last five bits are always zero. See Table 222.

Table 222. EMAC Transmit Read Pointer Register—Low Byte (EMAC_TRP_L = 0053h)

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] EMAC_TRP_L	00h–E0h	These bits represent the Low byte of the 2 byte EMAC TxDMA Transmit Read Pointer value, {EMAC_TRP_H, EMAC_TRP_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 16 bit value.

EMAC Transmit Read Pointer Register—High Byte

Because of the size of the EMAC's 8 KB SRAM, the upper three bits of the EMAC Transmit Read Pointer Register are always zero. See Table 223.

Table 223. EMAC Transmit Read Pointer Register—High Byte (EMAC_TRP_H = 0054h)

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

Note: R/W = Read/Write.

Bit Position	Value	Description
[7:0] EMAC_TRP_H	00h–1Fh	These bits represent the High byte of the 2 byte EMAC TxDMA Transmit Read Pointer value, {EMAC_TRP_H, EMAC_TRP_L}. Bit 7 is bit 15 (msb) of the 16 bit value. Bit 0 is bit 8 of the 16 bit value.

EMAC Receive Blocks Left Register—Low and High Bytes

This register reports the number of buffers left in the Receive EMAC shared memory. The hardware uses this information along with the block-level set in the EMAC_BUFSZ register to determine when to transmit a pause control frame. Software uses this information to determine when it must request that a pause control frame be transmitted (by setting bit 6 of the EMAC_CFG4 register). For the BlksLeft logic to operate properly, the Receive buffer must contain at least one more packet buffer than the number of packet buffers required for the largest packet. That is, one packet cannot fill the entire Receive buffer. Otherwise, the BlksLeft will be in error. See Table 224 and Table 225 on page 331.

Table 224. EMAC Receive Blocks Left Register—Low Byte (EMAC_BLKSLFT_L = 0055h)

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] EMAC_BLKSLFT_L	00h–FFh	These bits represent the Low byte of the 2 byte EMAC Receive Blocks Left value, {EMAC_BLKSLFT_H, EMAC_BLKSLFT_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 16 bit value.

Table 225. EMAC Receive Blocks Left Register—High Byte (EMAC_BLKSLFT_H = 0056h)

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] EMAC_BLKSLFT_H	00h–FFh	These bits represent the High byte of the 2 byte EMAC Receive Blocks Left value, {EMAC_BLKSLFT_H, EMAC_BLKSLFT_L}. Bit 7 is bit 15 (msb) of the 16 bit value. Bit 0 is bit 8 of the 16 bit value.

EMAC FIFO Data Register—Low and High Bytes

The FIFO Read/Write Test Access Data Register allows writing and reading the FIFO selected by the EMAC_TEST TxRx_SEL bit when the EMAC_TEST register TEST_FIFO bit is set. See Table 226 and Table 227.

Table 226. EMAC FIFO Data Register—Low Byte (EMAC_FDATA_L = 0057h)

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: R/W = Read/Write.

Bit Position	Value	Description
[7:0] EMAC_FDATA_L	00h–FFh	These bits represent the Low byte of the 10 bit EMAC FIFO data value, {EMAC_FDATA_H[1:0], EMAC_FDATA_L}. Bit 7 is bit 7 of the 16 bit value. Bit 0 is bit 0 (lsb) of the 10 bit value.

Table 227. EMAC FIFO Data Register—High Byte (EMAC_FDATA_H = 0058h)

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

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

Bit Position	Value	Description
[7:2]	00h	Reserved.
[1:0] EMAC_FDATA_H	0h–3h	These bits represent the upper two bits of the 10 bit EMAC FIFO data value, {EMAC_FDATA_H[1:0], EMAC_FDATA_L}. Bit 1 is bit 9 (msb) of the 16 bit value. Bit 0 is bit 8 of the 10 bit value.

EMAC FIFO Flags Register

The FIFO Flags value is set in the EMAC hardware to *half full*, or 16 bytes. See Table 228.

Table 228. EMAC FIFO Flags Register (EMAC_FFLAGS = 0059h)

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

Note: R = Read Only.

Bit Position	Value	Description
7 TFF	1	Transmit FIFO full
	0	Transmit FIFO not full
6	0	Reserved
5 TFAE	1	Transmit FIFO almost empty
	0	Transmit FIFO not almost empty
4 TFE	1	Transmit FIFO empty
	0	Transmit FIFO not empty
3 RFF	1	Receive FIFO full
	0	Receive FIFO not full
2 RFAF	1	Receive FIFO almost full
	0	Receive FIFO not almost full
1 RFAE	1	Receive FIFO almost empty
	0	Receive FIFO not almost empty
0 RFE	1	Receive FIFO empty
	0	Receive FIFO not empty

On-Chip Oscillators

The eZ80F91 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 also accepts a CMOS-level clock input signal. If an external clock generator is used, the X_{OUT} pin must be left unconnected. The secondary oscillator drives a 32 kHz crystal to generate the time-base for the Real-Time Clock.

Primary Crystal Oscillator Operation

Figure 63 on page 336 displays a recommended configuration for connection with an external 50 MHz, 3rd-overtone, parallel-resonant crystal. Recommended crystal specifications are provided in Table 229 on page 336 and Table 230 on page 337. Printed circuit board layout must add not more than 4 pF of stray capacitance to either the X_{IN} or X_{OUT} pins. If oscillation does not occur, try removing C1 for testing and decreasing the value of C_2 by the estimated stray capacitance to decrease loading.

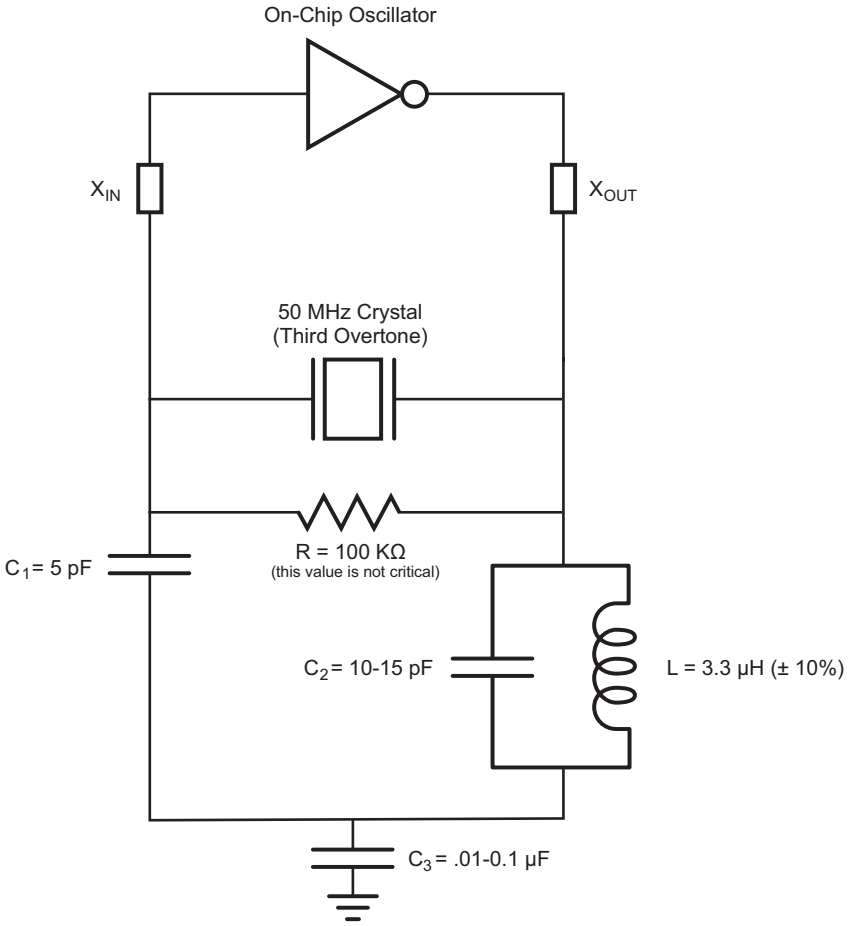


Figure 63. Recommended Crystal Oscillator Configuration—50 MHz Operation

Table 229. Recommended Crystal Oscillator Specifications—1 MHz Operation

Parameter	Frequency Dependent Value	Units	Comments
Frequency	1	MHz	
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	750	Ohms	Maximum
Load Capacitance (C_L)	13	pF	Maximum

Table 229. Recommended Crystal Oscillator Specifications—1 MHz Operation (Continued)

Parameter	Frequency Dependent Value	Units	Comments
Shunt Capacitance (C_0)	7	pF	Maximum
Drive Level	1	mW	Maximum

Table 230. Recommended Crystal Oscillator Specifications—10 MHz Operation

Parameter	Frequency Dependent Value	Units	Comments
Frequency	10	MHz	
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	35	Ohms	Maximum
Load Capacitance (C_L)	30	pF	Maximum
Shunt Capacitance (C_0)	7	pF	Maximum
Drive Level	1	mW	Maximum

32 kHz Real-Time Clock Crystal Oscillator Operation

Figure 64 on page 338 displays 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 231on page 338. A printed circuit board layout must not add 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.

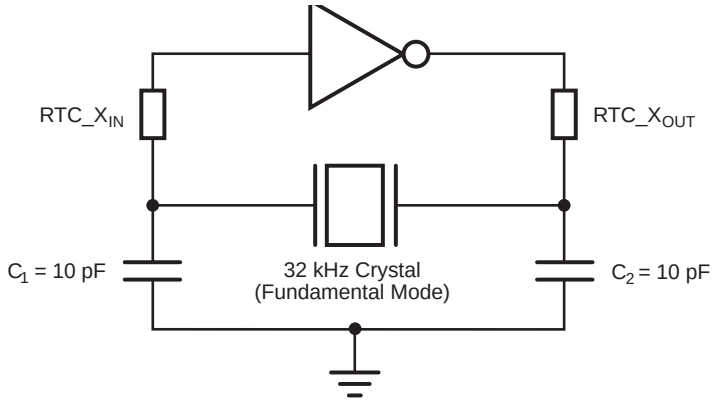


Figure 64. Recommended Crystal Oscillator Configuration—32 kHz Operation

Table 231. Recommended Crystal Oscillator Specifications—32 kHz Operation

Parameter	Value	Units	Comments
Frequency	32	kHz	32768 Hz
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	50	kΩ	Maximum
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 232 causes 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 affects device reliability. For improved reliability, unused inputs must be tied to one of the supply voltages (V_{DD} or V_{SS}).

Table 232. Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units	Notes
Ambient temperature under bias (ºC)	–40	+105	ºC	1
Storage temperature (ºC)	–65	+150	C	
Voltage on any pin with respect to V_{SS}	–0.3	+5.5	V	2
Voltage on V_{DD} pin with respect to V_{SS}	–0.3	+3.6	V	
Total power dissipation		830	mW	
Maximum current out of V_{SS}		230	mA	
Maximum current into V_{DD}		230	mA	
Maximum current on input and/or inactive output pin	–15	+15	µA	
Maximum output current from active output pin	–8	+8	mA	
Flash memory Writes to Same Single Address	—	2	—	3
Flash Memory Data Retention	100	—	Years	
Flash Memory Write/Erase Endurance	10,000	—	Cycles	4

Notes
1. Operating temperature is specified in DC Characteristics.
2. This voltage applies to all pins except X_{IN} and X_{OUT}.
3. Before next erase operation.
4. Write cycles.

DC Characteristics

Table 233 on page 340 lists the DC characteristics of the eZ80F91 device.

Table 233. DC Characteristics

		T_A = 0 ºC to 70 ºC			T_A = –40 ºC to 105 ºC				
Symbol	**Parameter**	**Minimum**	**Typ[2]**	**Maximum**	**Minimum**	**Typ[2]**	**Maximum**	**Units**	**Conditions**
V_{DD}	Supply Voltage	3.0	3.3	3.6	3.0	3.3	3.6	V	
V_{IL}	Low Level Input Voltage	–0.3		0.3 x V_{DD}	–0.3		0.3 x V_{DD}	V	
V_{IH}	High Level Input Voltage	0.7xV_{DD}		5.5	0.7xV_{DD}		5.5	V	
V_{OL}	Low Level Output Voltage			0.4			0.4	V	V_{DD} = 3.0 V; I_{OL} = 1 mA
V_{OH}	High Level Output Voltage	2.4			2.4			V	V_{DD} = 3.0 V; I_{OH} = –1 mA
V_{RTC}	RTC Supply Voltage	2.0		3.6	2.0		3.6	V	
I_{IL}	Input Leakage Current	–10		+10	–10		+10	µA	V_{DD} = 3.6 V; V_{IN} = V_{DD} or V_{SS}[1]
I_{TL}	Open-drain Leakage Current	–10		+10	–10		+10	µA	V_{DD} = 3.6 V
I_{CCa}	Active Current					26	40	mA	@ 10 MHz
						52	80	mA	@ 20 MHz
						137	190	mA	@ 50 MHz
I_{CCh}	HALT Mode Current					15	20	mA	@ 10 MHz
						27	40	mA	@ 20 MHz
						75	100	mA	@ 50 MHz
I_{CCs}	SLEEP Mode Current		2.5	20		2.5	95	µA	VBO_OFF=1 (VBO disabled)
I_{RTC}	RTC Supply Current		2.5	10		2.5	10	µA	Supply current into V_{RTC}

[1]This condition excludes all pins with on-chip pull-ups when driven Low.
[2]Values in Typical column are for Vdd = 3.3 V and T_A = 25 ºC.

POR and VBO Electrical Characteristics

Table 234 on page 341 lists the Power-On Reset and Voltage Brownout characteristics of the eZ80F91 device.

Table 234. POR and VBO Electrical Characteristics

		T_A = –40 ºC to 105 ºC				
Symbol	**Parameter**	**Minimum**	**Typ**	**Maximum**	**Unit**	**Conditions**
V_{VBO}	VBO Voltage Threshold	2.45	2.65	2.90	V	$V_{CC} = V_{VBO}$
V_{POR}	POR Voltage Threshold	2.55	2.75	2.95	V	$V_{CC} = V_{POR}$
V_{HYST}	POR/VBO Hysteresis	30	100	120	mV	
T_{ANA}	POR/VBO analog RESET duration	40		100	µs	
T_{VBO_MIN}	VBO pulse reject period		10		µs	
I_{POR_VBO}	POR/VBO DC current consumption		40	50	µA	
IS_{POR_VBO}	POR/VBO DC SLEEP mode current consumption		120	150	µA	VBO_OFF=0 (VBO enabled)
VCC_{RAMP}	V_{CC} ramp rate requirements to guarantee proper RESET occurs	0.1		100	V/ms	

Flash Memory Characteristics

Table 235 lists the Flash memory characteristics of the eZ80F91 device. For Flash programming and erase timing information, see Flash Memory on page 97.

Table 235. Flash Memory Electrical Characteristics and Timing

	V_{DD} = 3.0 V to 3.6 V; T_A = –40 ºC to 105 ºC				
Symbol	**Minimum**	**Typical**	**Maximum**	**Units**	**Notes**
Flash Byte Read Cycle Time	78	—	—	ns	

Current Consumption Under Various Operating Conditions

Figure 65 on page 342 displays the typical current consumption of the eZ80F91 device versus Vdd while operating at 25 ºC, with zero Wait states, and with either a 10 MHz, 20 MHz, or 50 MHz system clock.

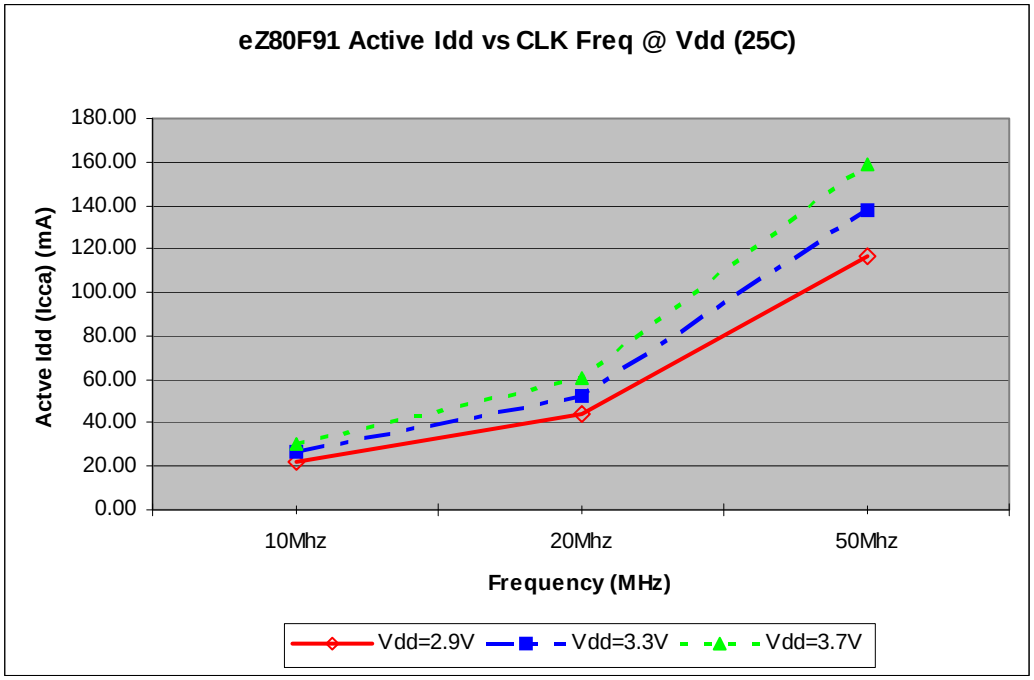


Figure 65. I_{CC} vs. System Clock Frequency During ACTIVE Mode

Figure 66 displays the typical current consumption of the eZ80F91 device versus system clock frequency while operating in HALT mode.

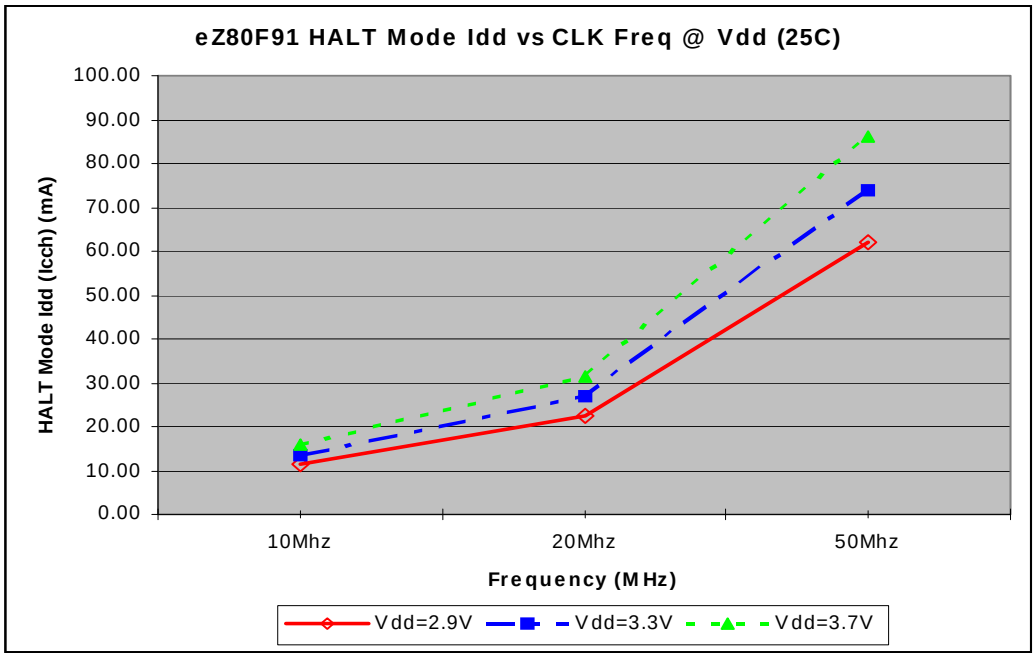


Figure 66. I_{CC} vs. System Clock Frequency During HALT Mode

Figure 67 displays the typical current consumption of the eZ80F91 device versus Vdd while operating in SLEEP mode (units in microamps, 10^{-6}A); all peripherals off, and VBO disabled.

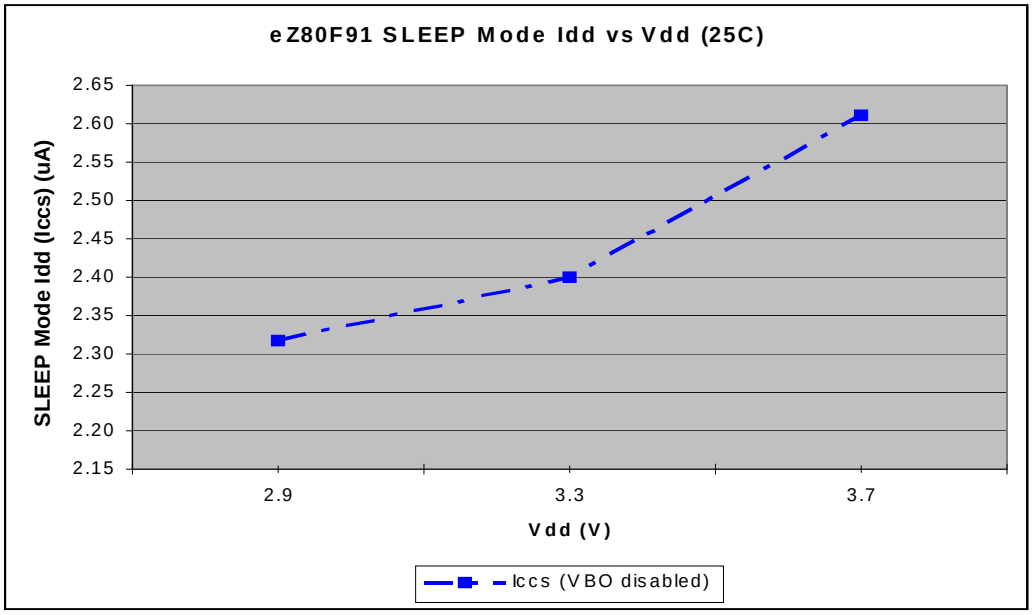


Figure 67. I_{CC} vs. Vdd During SLEEP Mode

AC Characteristics

This section provides information about the AC characteristics and timing of the eZ80F91 device. All AC timing information assumes a standard load of 50 pF on all outputs. See Table 236.

Table 236. AC Characteristics

		T_A = 0 °C to 70 °C		T_A = –40 °C to 105 °C			
Symbol	**Parameter**	**Minimum**	**Maximum**	**Minimum**	**Maximum**	**Units**	**Conditions**
T_{XIN}	System Clock Cycle Time	20	1000	20	1000	ns	V_{DD} = 3.0–3.6 V
T_{XINH}	System Clock High Time	8		8		ns	V_{DD} = 3.0–3.6 V; T_{CLK} = 20 ns
T_{XINL}	System Clock Low Time	8		8		ns	V_{DD} = 3.0–3.6 V; T_{CLK} = 20 ns

Table 236. AC Characteristics (Continued)

		T_A = 0 ºC to 70 ºC		T_A = –40 ºC to 105 ºC			
Symbol	**Parameter**	**Minimum**	**Maximum**	**Minimum**	**Maximum**	**Units**	**Conditions**
T_{XINR}	System Clock Rise Time		3		3	ns	V_{DD} = 3.0–3.6 V; T_{CLK} = 20 ns
T_{XINF}	System Clock Fall Time		3		3	ns	V_{DD} = 3.0–3.6 V; T_{CLK} = 20 ns
C_{IN}	Input capacitance	10 typical		10 typical		pF	

Table 237 lists simulated inductance, capacitance, and resistance results for the 144-pin LQFP package at 100 MHz operating frequency.

Table 237. Typical 144-LQFP Package Electrical Characteristics

Lead	Inductance (nH)	Capacitance (pF)	Resistance (mohm)
Longest	6.430	1.100	62.9
Shortest	4.230	1.070	52.6

Note: *Package vendor-supplied; 100 MHz operating frequency*

External Memory Read Timing

Figure 68 and Table 238 display the timing for external memory reads.

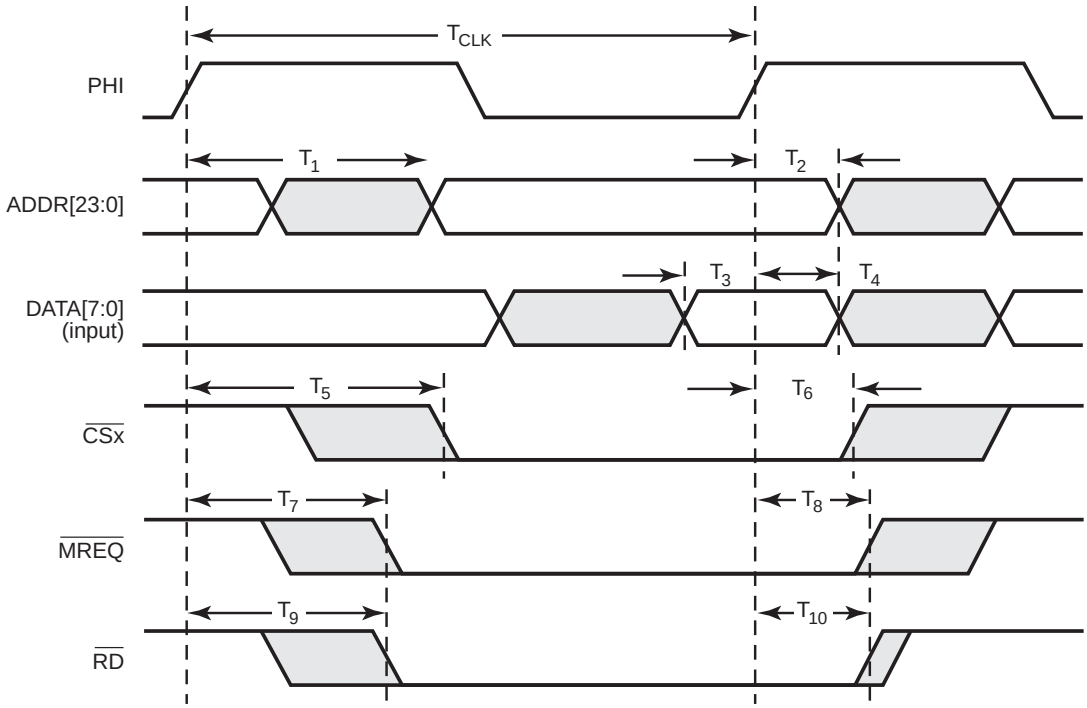


Figure 68. External Memory Read Timing

Table 238. External Memory Read Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	PHI Clock Rise to ADDR Valid Delay	—	8.5
T_2	PHI Clock Rise to ADDR Hold Time	1.0	—
T_3	DATA Valid to PHI Clock Rise Setup Time	0.5	—
T_4	PHI Clock Rise to DATA Hold Time	0.5	—
T_5	PHI Clock Rise to $\overline{CSx}$ Assertion Delay	2.6	8.0
T_6	PHI Clock Rise to $\overline{CSx}$ Deassertion Delay	0.0	6.0
T_7	PHI Clock Rise to $\overline{MREQ}$ Assertion Delay	2.6	7.0
T_8	PHI Clock Rise to $\overline{MREQ}$ Deassertion Delay	1.0	6.3

Table 238. External Memory Read Timing (Continued)

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_9	PHI Clock Rise to $\overline{RD}$ Assertion Delay	2.7	7.0
T_{10}	PHI Clock Rise to $\overline{RD}$ Deassertion Delay	1.0	6.3

External Memory Write Timing

Figure 69 and Table 239 on page 348 display the timing for external memory Writes.

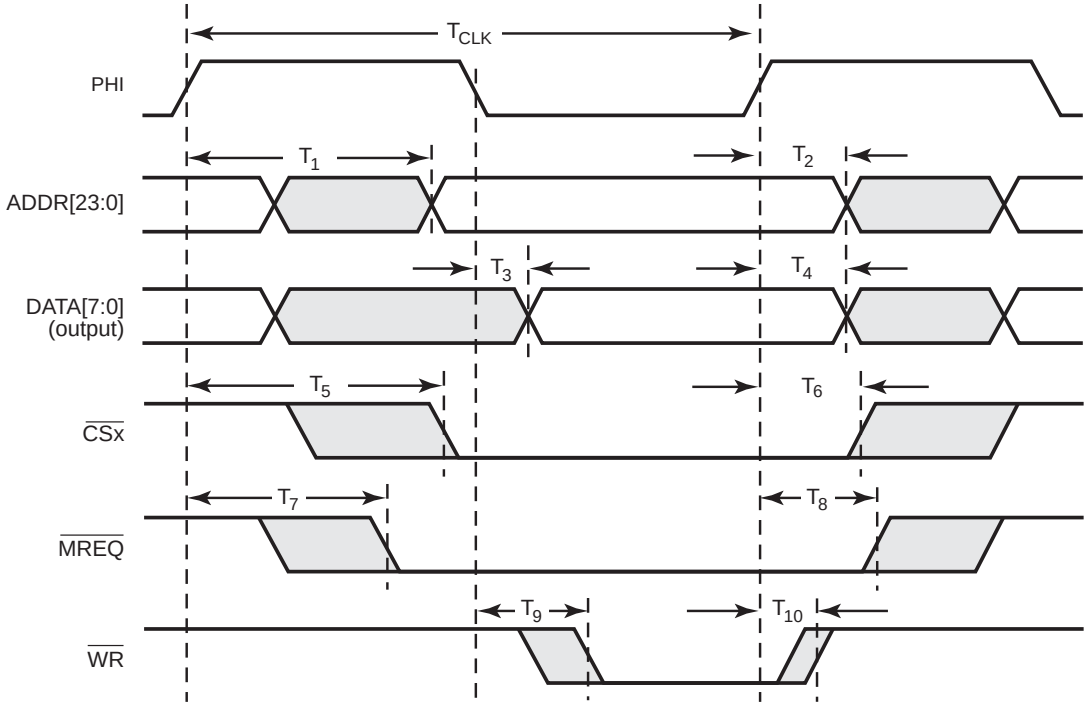


Figure 69. External Memory Write Timing

Table 239. External Memory Write Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	PHI Clock Rise to ADDR Valid Delay	—	8.5
T_2	PHI Clock Rise to ADDR Hold Time	1	—
T_3	PHI Clock Fall to DATA Valid	—	2.5
T_4	PHI Clock Rise to DATA Hold Time	1.0	—
T_5	PHI Clock Rise to $\overline{CSx}$ Assertion Delay	2.3	10.8
T_6	PHI Clock Rise to $\overline{CSx}$ Deassertion Delay	0.0	6.0
T_7	PHI Clock Rise to $\overline{MREQ}$ Assertion Delay	2.3	7.0
T_8	PHI Clock Rise to $\overline{MREQ}$ Deassertion Delay	2.3	6.5
T_9	PHI Clock Fall to $\overline{WR}$ Assertion Delay	—	1.0
T_{10}	PHI Clock Rise to $\overline{WR}$ Deassertion Delay*	0.0	5.0
	$\overline{WR}$ Deassertion to ADDR Hold Time	0.4	—
	$\overline{WR}$ Deassertion to DATA Hold Time	0.5	—
	$\overline{WR}$ Deassertion to $\overline{CSx}$ Hold Time	1.2	—
	$\overline{WR}$ Deassertion to $\overline{MREQ}$ Hold Time	0.5	—

*At the conclusion of a Write cycle, deassertion of $\overline{WR}$ always occurs before any change to ADDR, DATA, $\overline{CSx}$, or $\overline{MREQ}$.

External I/O Read Timing

Figure 70 and Table 240 display the timing for external I/O reads. PHI clock rise/fall to signal transition timing is independent of the particular bus mode employed (eZ80®, Z80®, Intel, or Motorola).

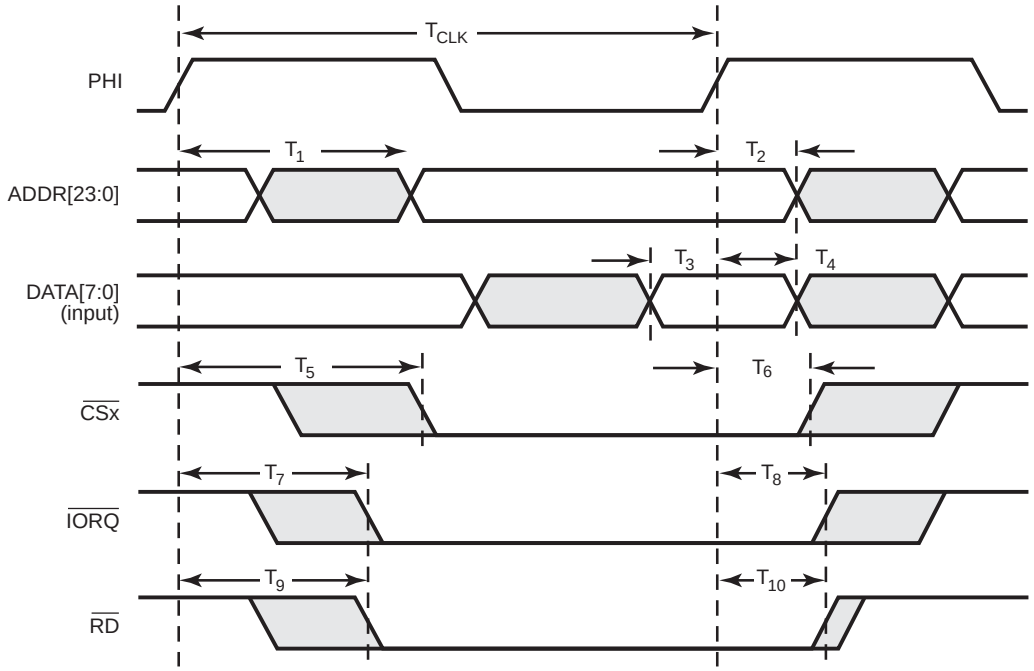


Figure 70. External I/O Read Timing

Table 240. External I/O Read Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	PHI Clock Rise to ADDR Valid Delay	—	7.3
T_2	PHI Clock Rise to ADDR Hold Time	1.0	—
T_3	DATA Valid to PHI Clock Rise Setup Time	0.5	—
T_4	PHI Clock Rise to DATA Hold Time	0.0	—
T_5	PHI Clock Rise to $\overline{CSx}$ Assertion Delay	2.0	8.5
T_6	PHI Clock Rise to $\overline{CSx}$ Deassertion Delay	0.0	6.0
T_7	PHI Clock Rise to $\overline{IORQ}$ Assertion Delay	2.6	7.0

Table 240. External I/O Read Timing (Continued)

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_8	PHI Clock Rise to $\overline{IORQ}$ Deassertion Delay	1.0	6.3
T_9	PHI Clock Rise to $\overline{RD}$ Assertion Delay	2.7	7.0
T_{10}	PHI Clock Rise to $\overline{RD}$ Deassertion Delay	0.5	6.3

External I/O Write Timing

Figure 71 and Table 241 on page 351 display the timing for external I/O Writes. PHI clock rise/fall to signal transition timing is independent of the particular bus mode employed (eZ80®, Z80®, Intel, or Motorola).

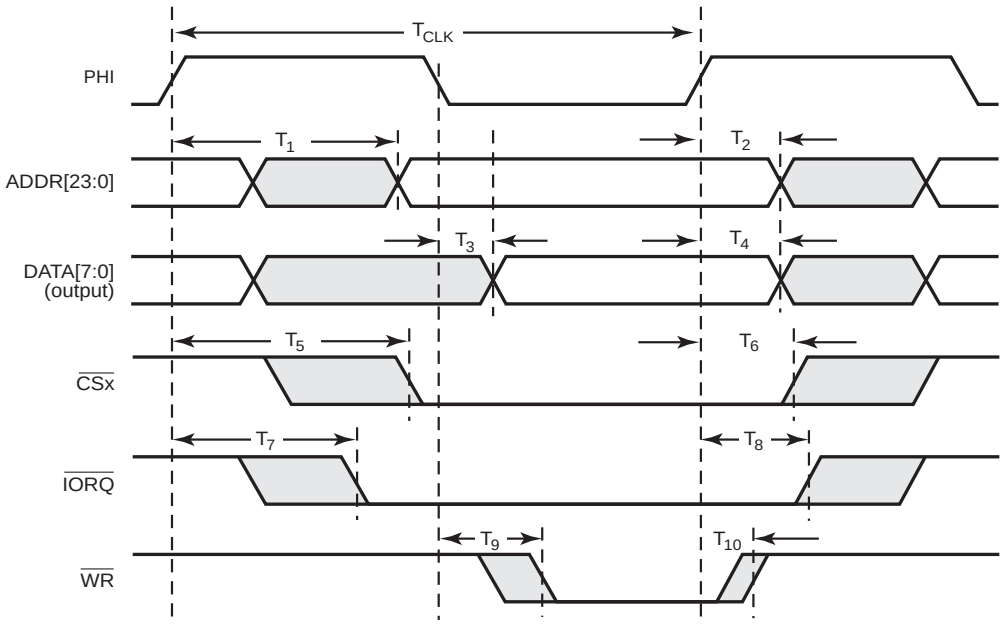


Figure 71. External I/O Write Timing

Table 241. External I/O Write Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
T_1	PHI Clock Rise to ADDR Valid Delay	—	7.3
T_2	PHI Clock Rise to ADDR Hold Time	1.0	—
T_3	PHI Clock Fall to DATA Valid	—	2.5
T_4	PHI Clock Rise to DATA Hold Time	1.0	—
T_5	PHI Clock Rise to $\overline{CSx}$ Assertion Delay	2.3	10.8
T_6	PHI Clock Rise to $\overline{CSx}$ Deassertion Delay	1.0	6.0
T_7	PHI Clock Rise to $\overline{IORQ}$ Assertion Delay	2.4	7.0
T_8	PHI Clock Rise to $\overline{IORQ}$ Deassertion Delay	1.0	6.3
T_9	PHI Clock Fall to $\overline{WR}$ Assertion Delay	—	1.0
T_{10}	PHI Clock Rise to $\overline{WR}$ Deassertion Delay*	0.0	5.0
	$\overline{WR}$ Deassertion to ADDR Hold Time	0.4	—
	$\overline{WR}$ Deassertion to DATA Hold Time	0.5	—
	$\overline{WR}$ Deassertion to $\overline{CSx}$ Hold Time	1.2	—
	$\overline{WR}$ Deassertion to $\overline{IORQ}$ Hold Time	0.5	—

*At the conclusion of a Write cycle, deassertion of $\overline{WR}$ always occurs before any change to ADDR, DATA, $\overline{CSx}$, or $\overline{IORQ}$.

Wait State Timing for Read Operations

Figure 72 displays 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.

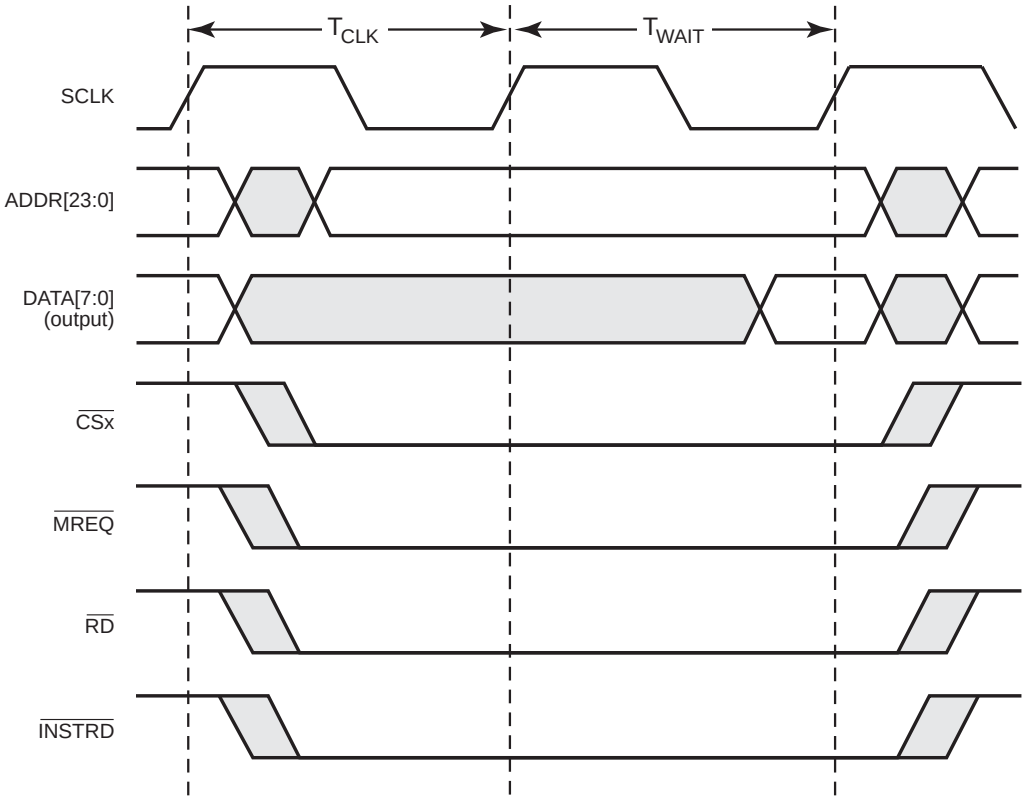


Figure 72. Wait State Timing for Read Operations

Wait State Timing for Write Operations

Figure 73 displays 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.

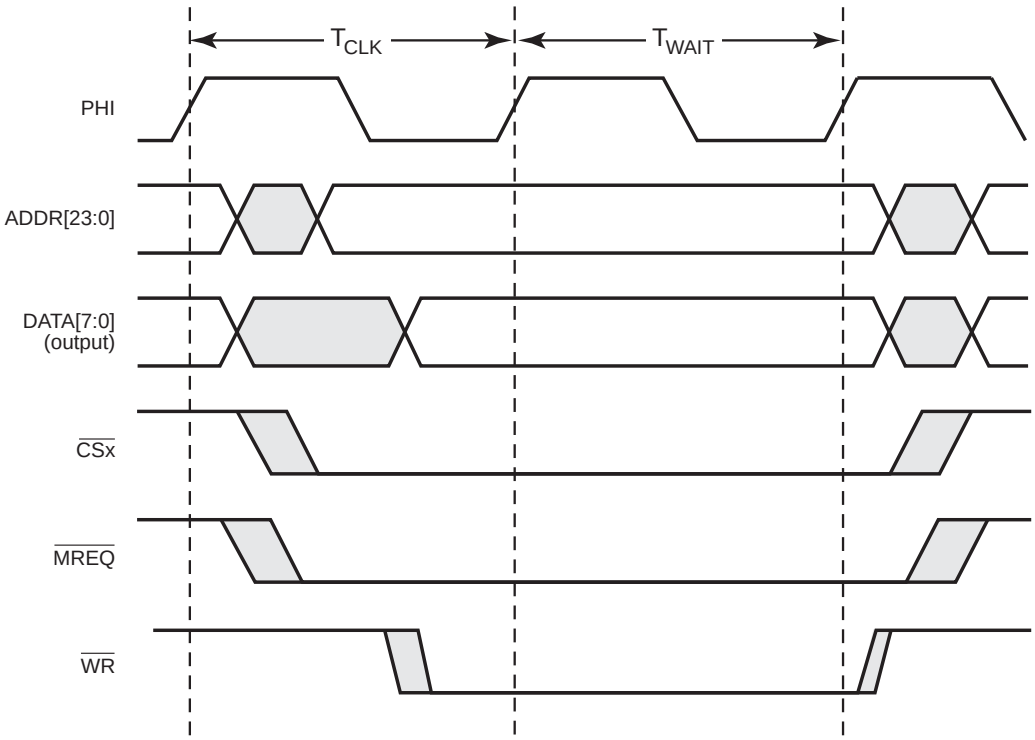


Figure 73. Wait State Timing for Write Operations

General-Purpose Input/Output Port Input Sample Timing

Figure 74 displays 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.

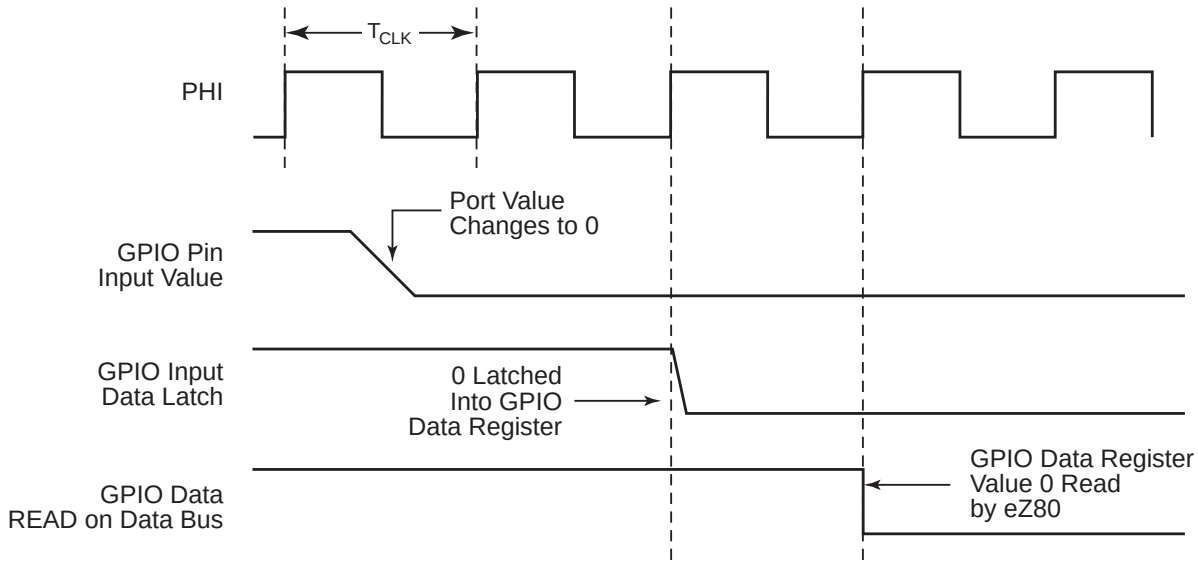


Figure 74. Port Input Sample Timing

General-Purpose I/O Port Output Timing

Figure 75 and Table 242 on page 355 display timing information for GPIO port pins.

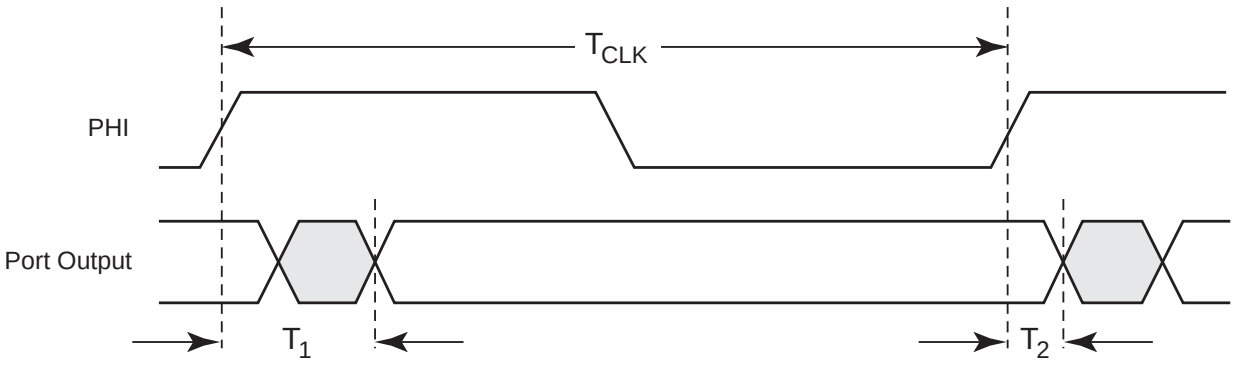


Figure 75. GPIO Port Output Timing

Table 242. GPIO Port Output Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	PHI Clock Rise to Port Output Valid Delay	—	5
T_2	PHI Clock Rise to Port Output Hold Time	1.0	—

External Bus Acknowledge Timing

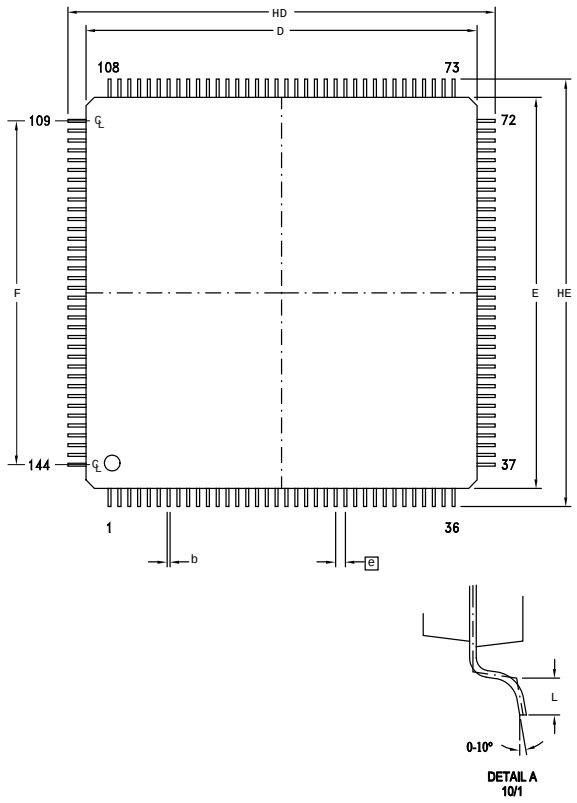
Table 243 lists information on the bus acknowledge timing.

Table 243. Bus Acknowledge Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	PHI Clock Rise to $\overline{\text{BUSACK}}$ Assertion Delay	2.8	7.1
T_2	PHI Clock Rise to $\overline{\text{BUSACK}}$ Deassertion Delay	1.5	6.5

Packaging

Figure 76 displays the 144-pin low-profile quad flat package (LQFP) for the eZ80F91 device.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	1.40	1.60	0.055	0.063
A1	0.05	0.15	0.002	0.006
A2	1.35	1.45	0.053	0.057
b	0.17	0.27	0.007	0.011
c	0.09	0.20	0.004	0.008
HD	21.75	22.25	0.856	0.876
D	19.90	20.10	0.783	0.791
HE	21.75	22.25	0.856	0.876
E	19.90	20.10	0.783	0.791
e	0.50 BSC		0.020 BSC	
F	17.5		0.689	
L	0.35	0.65	0.014	0.026
LE	1.00 REF		.039 REF	

CONTROLLING DIMENSIONS : MM
MAX COPLANARITY : .10mm
.004"

Figure 76. 144-Lead Plastic Low-Profile Quad Flat Package (LQFP)

Figure 77 displays the 144-pin chip array ball grid array (BGA) package for the eZ80F91 device.

Figure 77. 144-Lead Chip Array Ball Grid Array (BGA)

Ordering Information

Table 244 lists part name, a product specification index code, and a brief description of each part. Order the eZ80F91 microcontroller from Zilog®, using the following part numbers. For more information on ordering, please consult your local Zilog sales office. The Zilog website (www.zilog.com) lists all regional offices and provides additional eZ80F91 microcontroller product information.

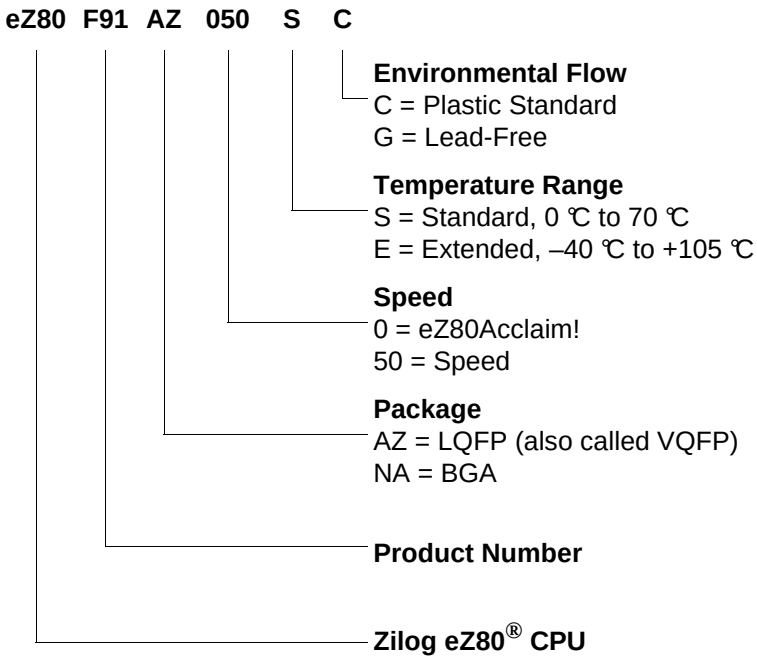
Table 244. Ordering Information

Part	PSI	Description
eZ80F91	eZ80F91AZ050SG*	144-pin LQFP, 256 KB Flash memory, 8 KB SRAM, 50 MHz, Standard Temperature
	eZ80F91AZ050EG*	144-pin LQFP, 256 KB Flash memory, 8 KB SRAM, 50 MHz, Extended Temperature
	eZ80F91NA050SG*	144-pin BGA, 256 KB Flash memory, 8 KB SRAM, 50 MHz, Standard Temperature
	eZ80F91NA050EG*	144-pin BGA, 256 KB Flash memory, 8 KB SRAM, 50 MHz, Extended Temperature
	eZ80F910200ZC0G	eZ80F91 Acclaim! Development Kit
	eZ80F910100KITG	eZ80F91 Acclaim! Modular Development Kit
	eZ80F9105050MODG	Ethernet Module
	eZ80F9105005MODG	Mini Ethernet Module
	ZUSBSC00100ZACG	USB Smart Cable
	ZENETSC0100ZACG	Ethernet Smart Cable

*Denotes parts not recommended for new designs.

Part Number Description

Zilog® part numbers consists of number of components as described below:



Example: Part number eZ80F91AZ050SC is an eZ80F91 Acclaim! product in a LQFP package, operating with a 50 MHz external clock frequency over a 0 ºC to +70 ºC temperature range and built using the Plastic Standard environmental flow.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Customer Support

For answers to technical questions about the product, documentation, or any other issues with Zilog's offerings, visit Zilog's Knowledge Base at http://www.zilog.com/kb.

For any comments, detail technical questions, or reporting problems, visit Zilog's Technical Support at http://support.zilog.com.