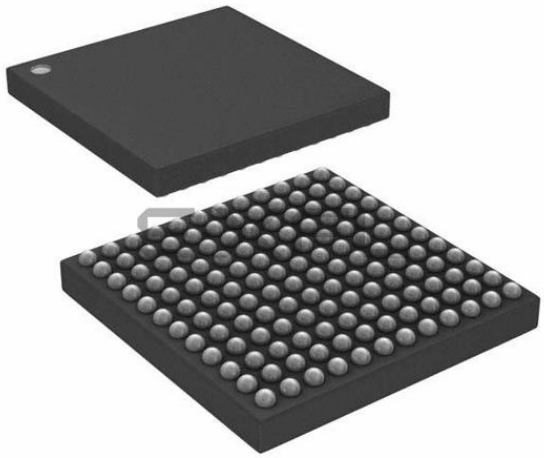

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	eZ80
Core Size	8-Bit
Speed	50MHz
Connectivity	Ethernet, I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	16K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LBGA
Supplier Device Package	144-BGA (13x13)
Purchase URL	https://www.e-xfl.com/product-detail/zilog/ez80f91na050sc

- 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.

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 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

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

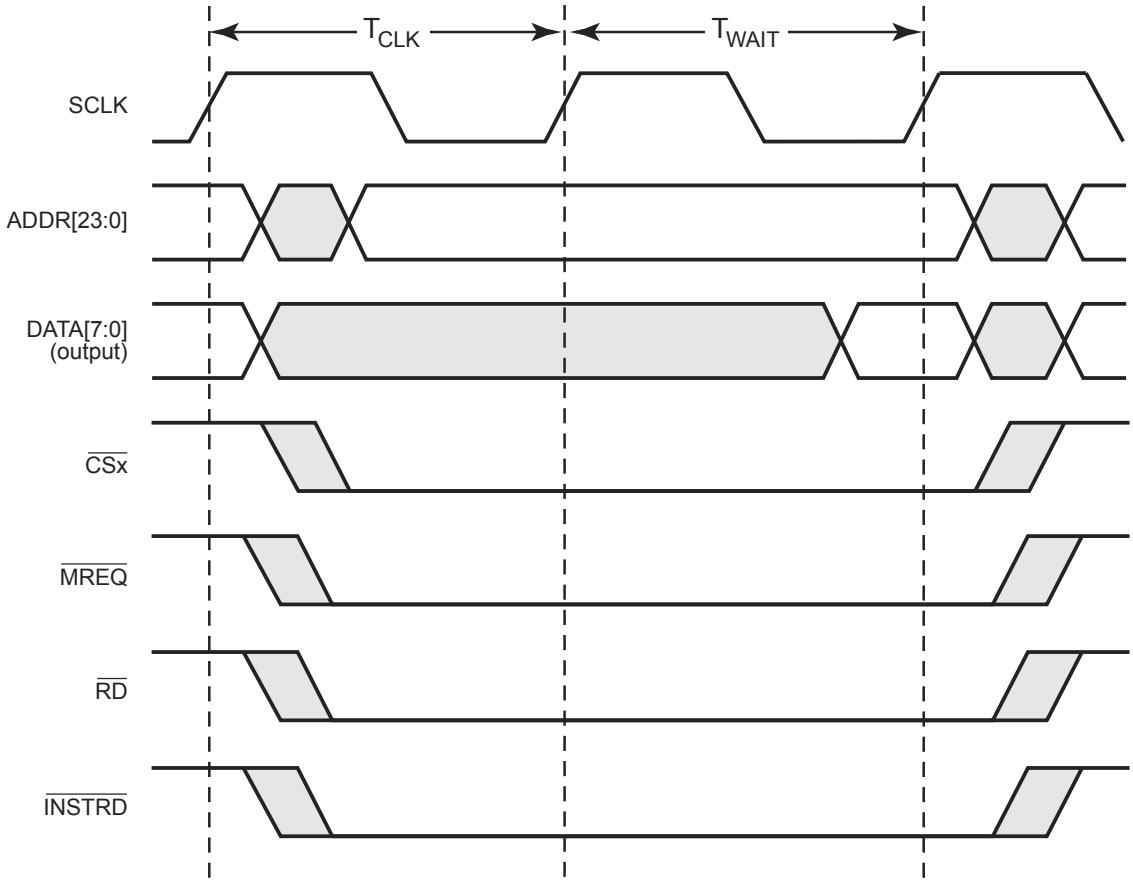

Figure 9. Example: Wait State Read Operation

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

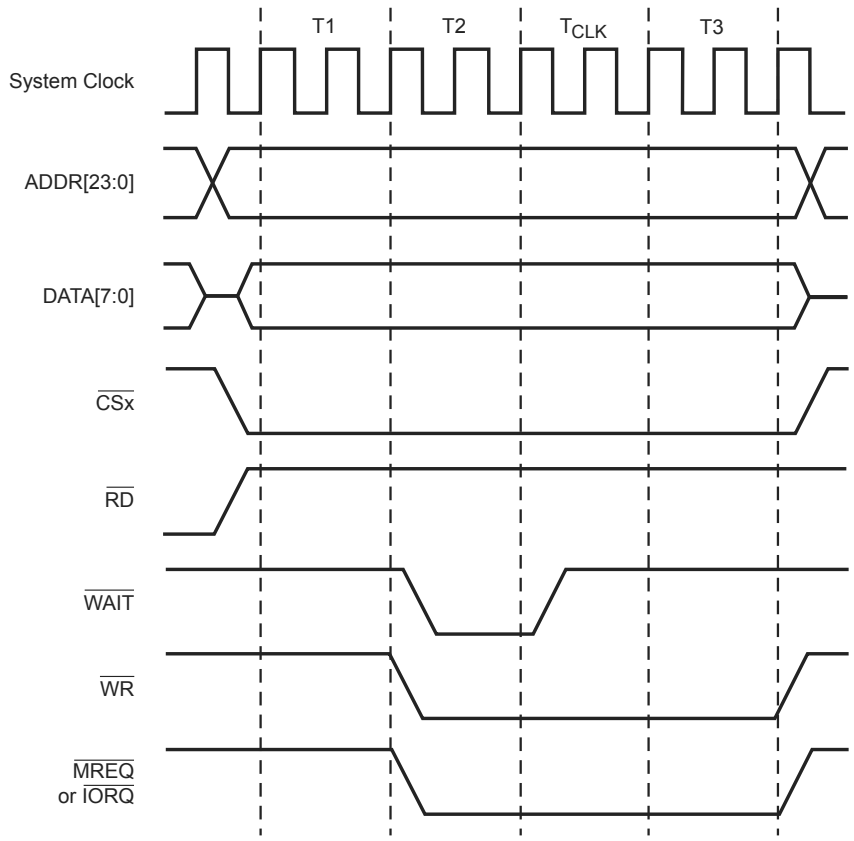

Figure 11. Example: Z80® Bus Mode Write Timing

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.

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.

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.

[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.

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.*

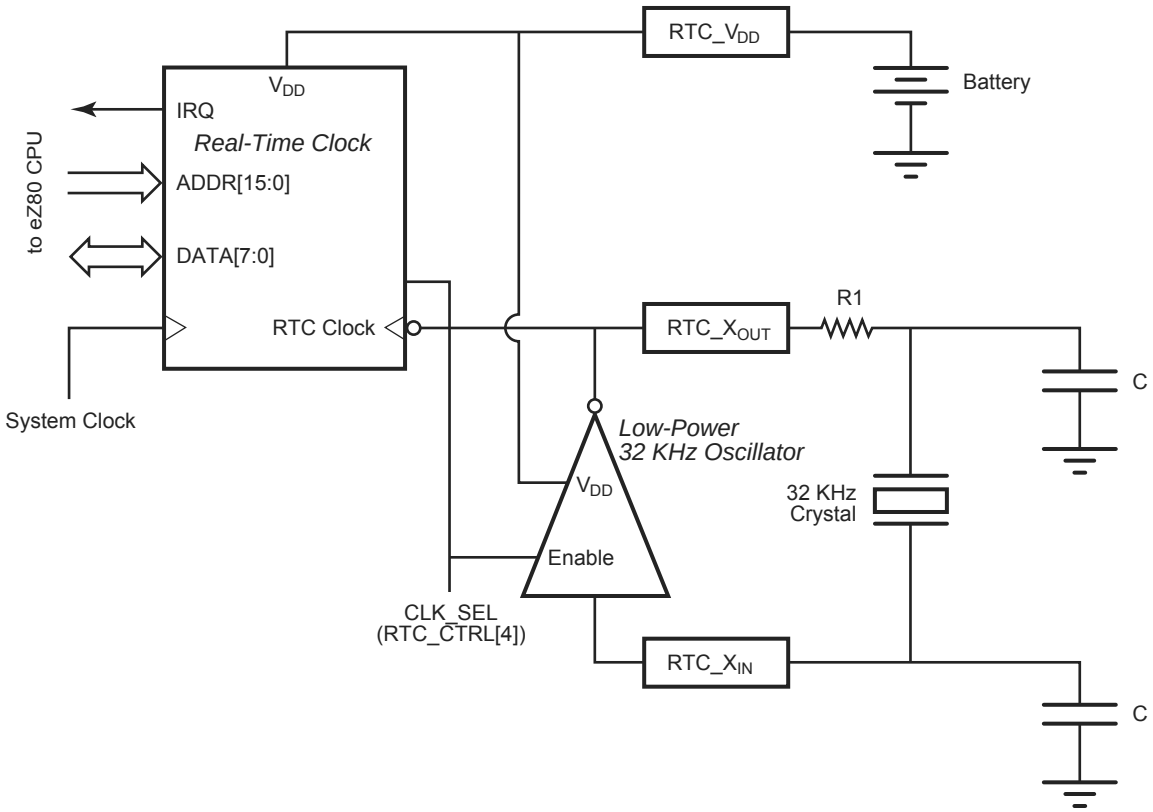

Figure 35. Real-Time Clock and 32 kHz Oscillator Block Diagram

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.

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

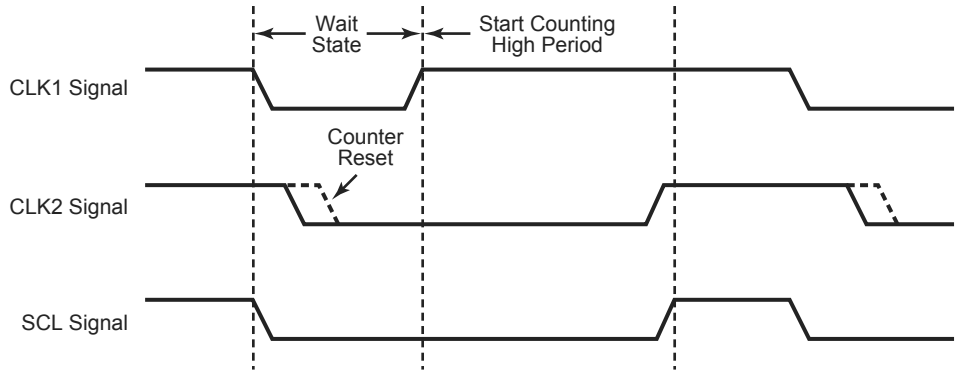

Figure 47. Clock Synchronization In I²C Protocol

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

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.

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.

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.

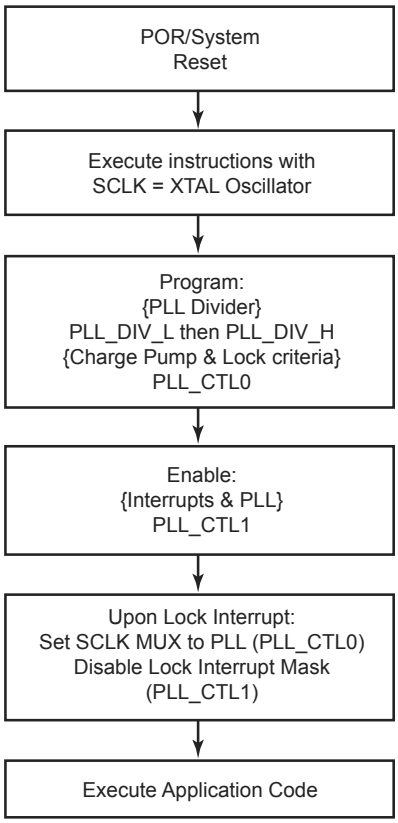

Figure 58. Normal PLL Programming Flow

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 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.

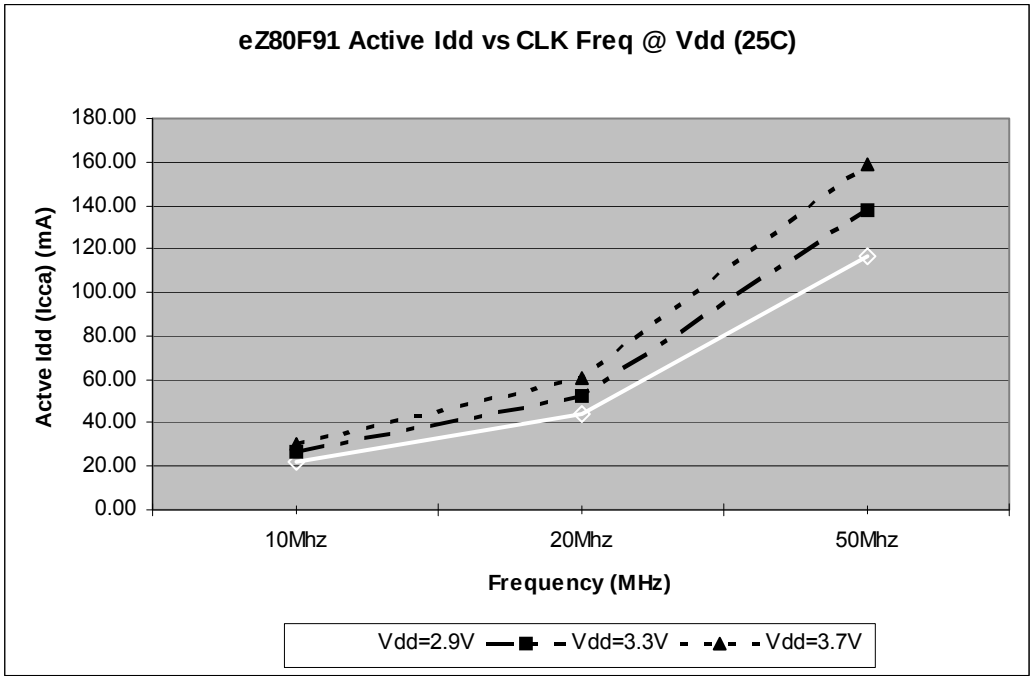

Figure 65. I_{CC} vs. System Clock Frequency During ACTIVE Mode