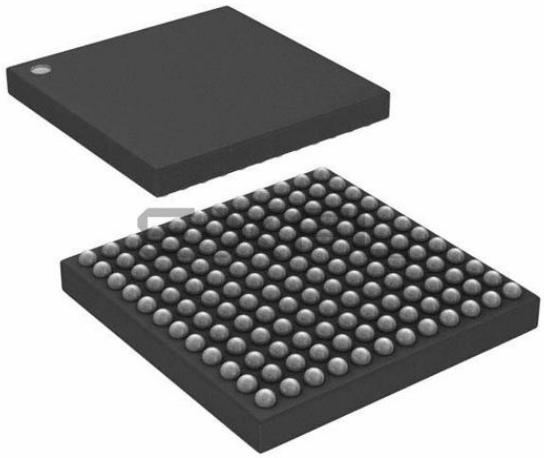


**Welcome to E-XFL.COM**

## 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 | -40°C ~ 105°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/ez80f91na050ec |

**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.

# 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).

**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<br>GPIO_D_OFF | 1 | System clock to GPIO Port D is powered down.<br>Port D alternate functions do not operate correctly. |
| | 0 | System clock to GPIO Port D is powered up. |
| 6<br>GPIO_C_OFF | 1 | System clock to GPIO Port C is powered down.<br>Port C alternate functions do not operate correctly. |
| | 0 | System clock to GPIO Port C is powered up. |
| 5<br>GPIO_B_OFF | 1 | System clock to GPIO Port B is powered down.<br>Port B alternate functions do not operate correctly. |
| | 0 | System clock to GPIO Port B is powered up. |
| 4<br>GPIO_A_OFF | 1 | System clock to GPIO Port A is powered down.<br>Port A alternate functions do not operate correctly. |
| | 0 | System clock to GPIO Port A is powered up. |
| 3<br>SPI_OFF | 1 | System clock to SPI is powered down. |
| | 0 | System clock to SPI is powered up. |
| 2<br>I2C_OFF | 1 | System clock to $I^2C$ is powered down. |
| | 0 | System clock to $I^2C$ is powered up. |
| 1<br>UART1_OFF | 1 | System clock to UART1 is powered down. |
| | 0 | System clock to UART1 is powered up. |
| 0<br>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 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.

**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. |

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 | |

**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<br>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<br>MIIE | 0 | Modem interrupt on edge detect of status inputs is disabled. |
| | 1 | Modem interrupt on edge detect of status inputs is enabled. |
| 2<br>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<br>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<br>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. |

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

### 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.<br>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. |

**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.

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

| Bit Position | Value | Description |
|---|---|---|
| 2<br>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<br>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<br>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. |

**Table 166. Rotate and Shift Instructions (Continued)**

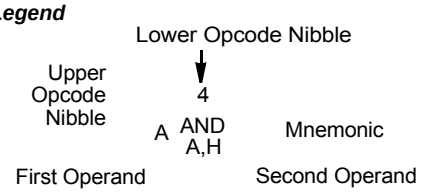
| Mnemonic | Instruction |
|---|---|
| RRA | Rotate Right–Accumulator |
| RRC | Rotate Right Circular |
| RRCA | Rotate Right Circular–Accumulator |
| RRD | Rotate Right Decimal |
| SLA | Shift Left Arithmetic |
| SRA | Shift Right Arithmetic |
| SRL | Shift Right Logical |

## 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 |

### 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 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<br>TFF | 1 | Transmit FIFO full |
| | 0 | Transmit FIFO not full |
| 6 | 0 | Reserved |
| 5<br>TFAE | 1 | Transmit FIFO almost empty |
| | 0 | Transmit FIFO not almost empty |
| 4<br>TFE | 1 | Transmit FIFO empty |
| | 0 | Transmit FIFO not empty |
| 3<br>RFF | 1 | Receive FIFO full |
| | 0 | Receive FIFO not full |
| 2<br>RFAF | 1 | Receive FIFO almost full |
| | 0 | Receive FIFO not almost full |
| 1<br>RFAE | 1 | Receive FIFO almost empty |
| | 0 | Receive FIFO not almost empty |
| 0<br>RFE | 1 | Receive FIFO empty |
| | 0 | Receive FIFO not empty |

**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).

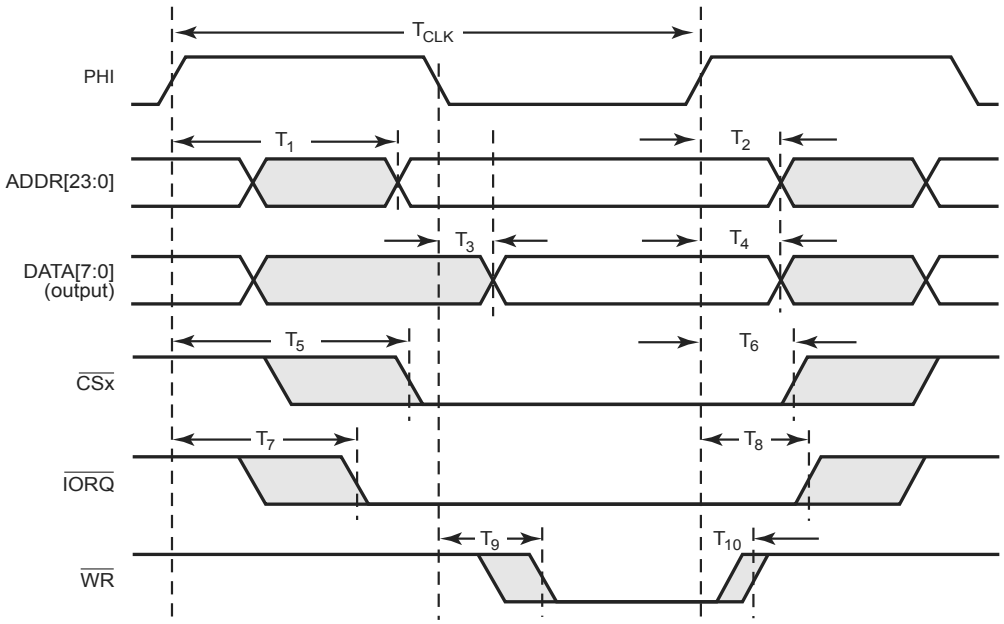



**Figure 71. External I/O Write Timing**

## V



## W



## X



## Z