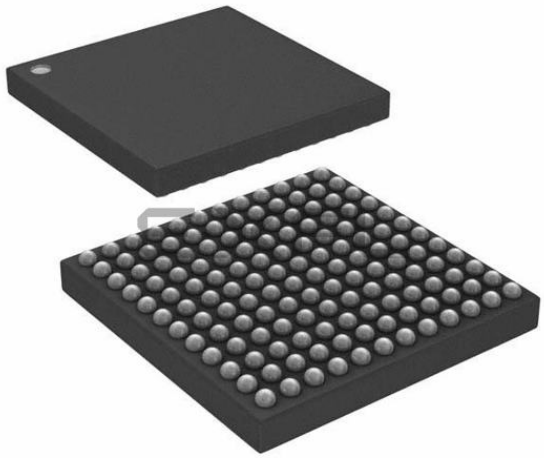

Welcome to E-XFL.COM

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/ez80f91na050sg

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
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.

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).

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.

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.

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

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.

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.

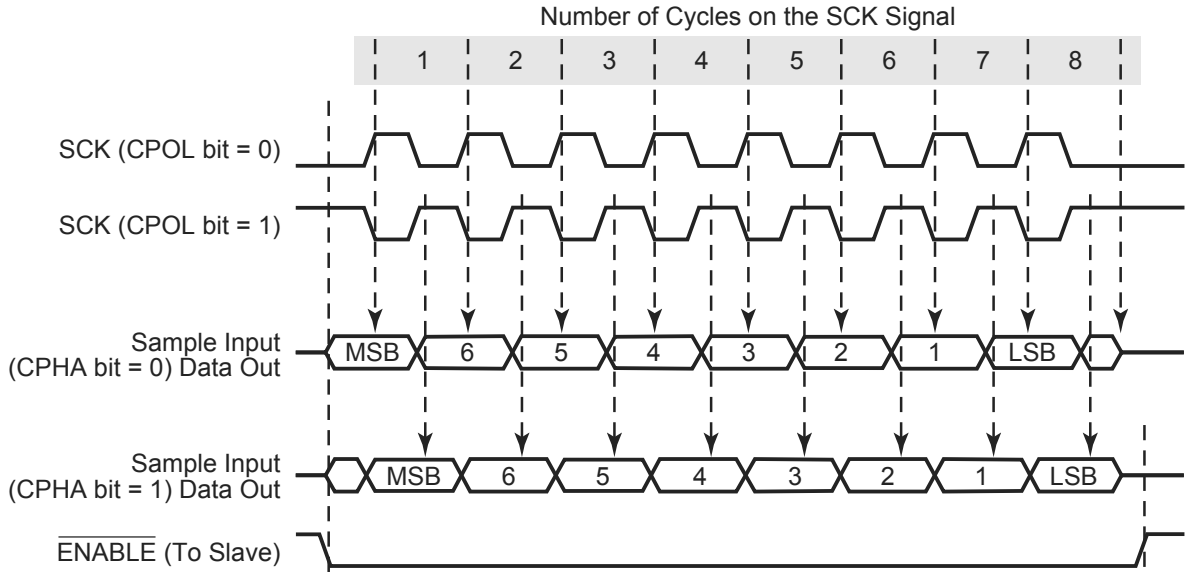

Figure 42. SPI Timing

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.

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.

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.

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.

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.

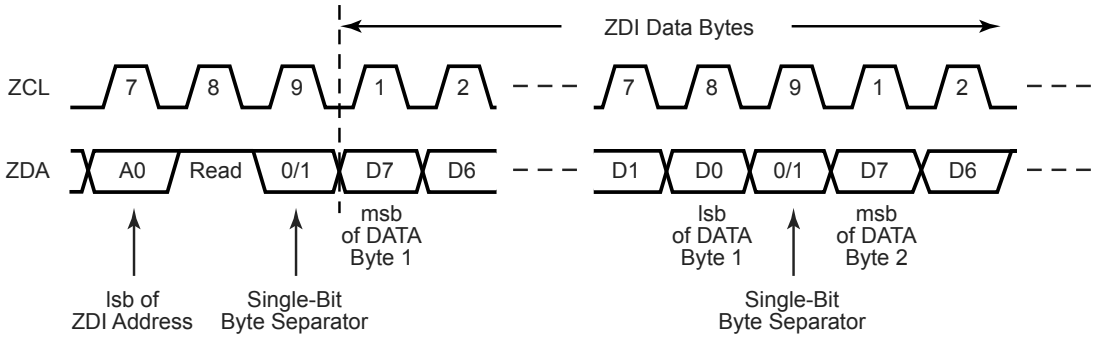

Figure 56. ZDI Block Data Read Timing

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-

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.

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

Upper Nibble (Hex) \ Lower 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 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

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.

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.

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.