Zilog - EZ80F93AZ020EC Datasheet

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	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	24
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K 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	100-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/ez80f93az020ec

Table 3. Register Map (Continued)

Address (hex)	Mnemonic	Name	Reset (hex)	CPU Access	Page No
00D1	UART1_IER	UART 1 Interrupt Enable Register	00	R/W	113
	UART1_BRG_H	UART 1 Baud Rate Generator Register—High Byte	00	R/W	111
00D2	UART1_IIR	UART 1 Interrupt Identification Register	01	R	114
	UART1_FCTL	UART 1 FIFO Control Register	00	W	115
00D3	UART1_LCTL	UART 1 Line Control Register	00	R/W	116
00D4	UART1_MCTL	UART 1 Modem Control Register	00	R/W	119
00D5	UART1_LSR	UART 1 Line Status Register	60	R/W	120
00D6	UART1_MSR	UART 1 Modem Status Register	XX	R/W	122
00D7	UART1_SPR	UART 1 Scratch Pad Register	00	R/W	123
Low-Power Control					
00DB	CLK_PPD1	Clock Peripheral Power-Down Register 1	00	R/W	37
00DC	CLK_PPD2	Clock Peripheral Power-Down Register 2	00	R/W	38
Real-Time Clock					
00E0	RTC_SEC	RTC Seconds Register[3]	XX	R/W	90
00E1	RTC_MIN	RTC Minutes Register[3]	XX	R/W	91
00E2	RTC_HRS	RTC Hours Register[3]	XX	R/W	92
00E3	RTC_DOW	RTC Day-of-the-Week Register[3]	0X	R/W	93
00E4	RTC_DOM	RTC Day-of-the-Month Register[3]	XX	R/W	94
00E5	RTC_MON	RTC Month Register[3]	XX	R/W	95
00E6	RTC_YR	RTC Year Register[3]	XX	R/W	96
00E7	RTC_CEN	RTC Century Register[3]	XX	R/W	97
00E8	RTC_ASEC	RTC Alarm Seconds Register	XX	R/W	98
00E9	RTC_AMIN	RTC Alarm Minutes Register	XX	R/W	99
00EA	RTC_AHRS	RTC Alarm Hours Register	XX	R/W	100
00EB	RTC_ADOW	RTC Alarm Day-of-the-Week Register	0X	R/W	101
00EC	RTC_ACTRL	RTC Alarm Control Register	00	R/W	102
00ED	RTC_CTRL	RTC Control Register[4]	x0xxx000b/ x0xxxx10b	R/W	103

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

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

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

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

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

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

A simplified block diagram of a GPIO port pin is displayed in Figure 5 on page 42.

an interrupt request signal to the CPU. Any time a port pin is configured for edge-triggered interrupt, writing a 1 to that pin's Port *x* Data register causes a reset of the edge-detected interrupt. The programmer must set the bit in the Port *x* Data register to 1 before entering either single or dual edge-triggered interrupt mode for that port pin.

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

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

GPIO Control Registers

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

Port *x* Data Registers

When the port pins are configured for one of the output modes, the data written to the Port *x* Data registers, listed in Table 7, are driven on the corresponding pins. In all modes, reading from the Port *x* Data registers always returns the current sampled value of the corresponding pins.

When the port pins are configured as edge-triggered interrupt sources, writing a 1 to the corresponding bit in the Port *x* Data register clears the interrupt signal that is sent to the CPU. When the port pins are configured for edge-selectable interrupts or level-sensitive interrupts, the value written to the Port *x* Data register bit selects the interrupt edge or interrupt level. See Table 6 on page 39 for more information.

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

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

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

Notes

1. Setting BUS_CYCLE to 1 in Intel™ bus mode causes the ALE pin to not function properly.
2. Use of the external $\overline{\text{WAIT}}$ input pin in Z80® mode requires that BUS_CYCLE is set to a value greater than 1.
3. BUS_CYCLE produces no effect in eZ80® mode.

Watchdog Timer Operation

Enabling and Disabling the WDT

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

Time-Out Period Selection

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

Table 26. Watchdog Timer Approximate Time-Out Delays

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

Note: *WDT time-out values should be sufficiently long to allow Flash operations to complete.

RESET Or NMI Generation

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

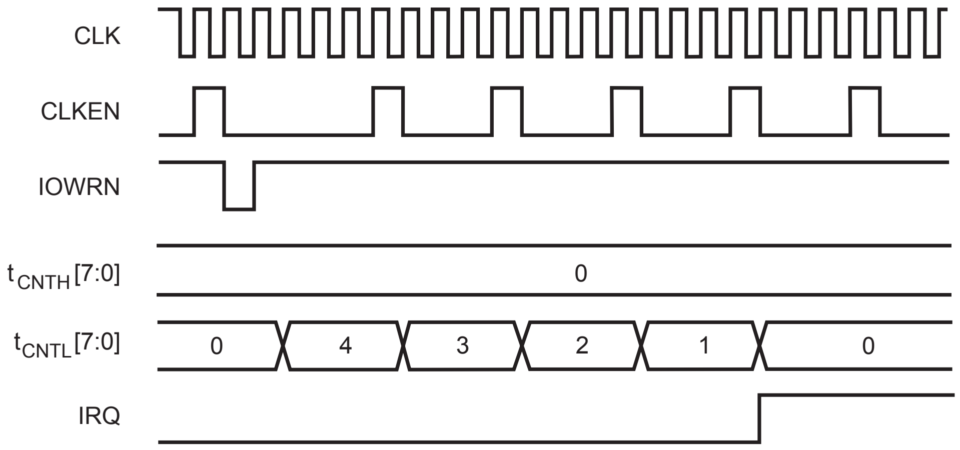

Figure 21.PRT SINGLE PASS Mode Operation Example

Table 29. PRT SINGLE PASS Mode Operation Example

Parameter	Control Register(s)	Value
PRT Enabled	TMRx_CTL[0]	1
Reload and Restart Enabled	TMRx_CTL[1]	1
PRT Clock Divider = 4	TMRx_CTL[3:2]	00b
SINGLE PASS Mode	TMRx_CTL[4]	0
PRT Interrupt Enabled	TMRx_CTL[6]	1
PRT Reload Value	{TMRx_RR_H, TMRx_RR_L}	0004h

Continuous Mode

In CONTINUOUS mode, when the end-of-count value, `0000h`, is reached, the timer automatically reloads the 16-bit start value from the Timer Reload registers, TMRx_RR_H and TMRx_RR_L. Downcounting continues on the next clock edge. In CONTINUOUS mode, the PRT continues to count until disabled. An example of a PRT operating in CONTINUOUS mode is displayed in Figure 22 on page 79. Timer register information is listed in Table 30 on page 79.

Timer Reload Register—High Byte

The Timer Reload Register—High Byte, listed in Table 36, stores the most-significant byte (MSB) of the 2-byte timer reload value. In CONTINUOUS mode, the timer reload value is reloaded into the timer upon end-of-count. When RST_EN (TMRx_CTL[1]) is set to 1 to enable the automatic reload and restart function, the timer reload value is written to the timer on the next rising edge of the clock.

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

Table 36. Timer Reload Register—High Byte(TMR0_RR_H = 0082h, TMR1_RR_H = 0085h, TMR2_RR_H = 0088h, TMR3_RR_H = 008Bh, TMR4_RR_H = 008Eh, or TMR5_RR_H = 0091h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	0	0
CPU Access	W	W	W	W	W	W	W	W
Note: W = Write only.								

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

Timer Input Source Select Register

The Timer Input Source Select register, listed in Table 37 on page 86, sets the input source for Programmable Reload Timer 0–3 (TMR0, TMR1, TMR2, TMR3). Event frequency must be less than one-half of the system clock frequency. When configured for event inputs through the port pins, the Timers decrement on the fifth system clock rising edge following the rising edge of the port pin. The timer event input can arrive from the GPIO port, the real-time clock, or the system clock. The value of the clock divider in the Timer Control Register is ignored when the timer event input is either from the GPIO port pin or the real-time clock source.

SPI Transmit Shift Register

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

The SPI Transmit Shift Write Only register shares the same address space as the SPI Receive Buffer Read Only register. See Table 76.

Table 76. SPI Transmit Shift Register(SPI_TSR = 00BCh)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	X	X	X	X	X	X	X	X
CPU Access	W	W	W	W	W	W	W	W
Note: W = Write only.								

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

SPI Receive Buffer Register

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

The SPI Receive Buffer Read Only register shares the same address space as the SPI Transmit Shift Write Only register. See Table 77.

Table 77. SPI Receive Buffer Register(SPI_RBR = 00BCh)

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

Data Validity

The data on the SDA line must be stable during the High period of the clock. The High or Low state of the data line can only change when the clock signal on the SCL line is Low as displayed in Figure 32.

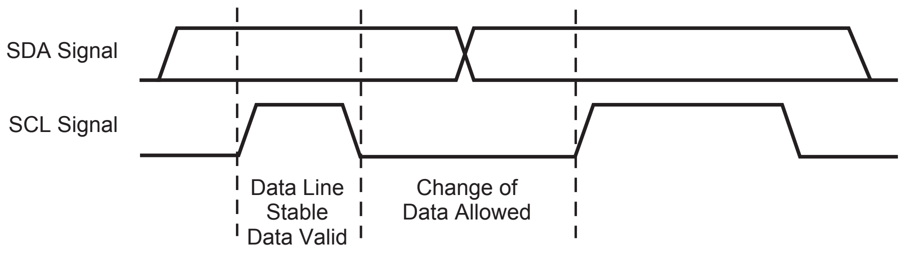

Figure 32. I^{2}C Clock and Data Relationship

START and STOP Conditions

Within the I^{2}C bus protocol, unique situations arise which are defined as START and STOP conditions. See Figure 33. A High-to-Low transition on the SDA line while SCL is High indicates a START condition. A Low-to-High transition on the SDA line while SCL is High defines a STOP condition.

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

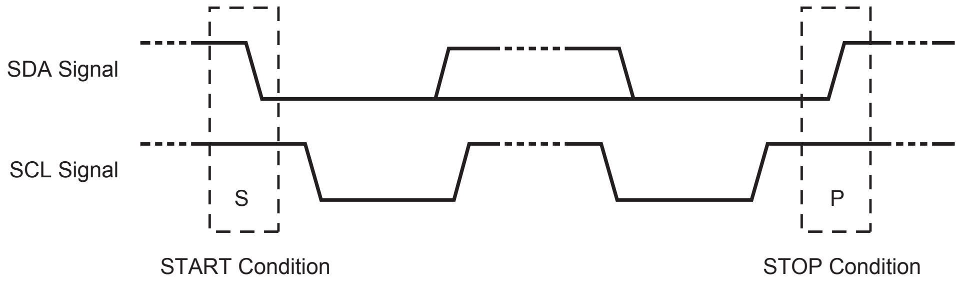

Figure 33. START and STOP Conditions In I^{2}C Protocol

The slave-transmitter must release the data line to allow the master to generate a STOP or a repeated START condition.

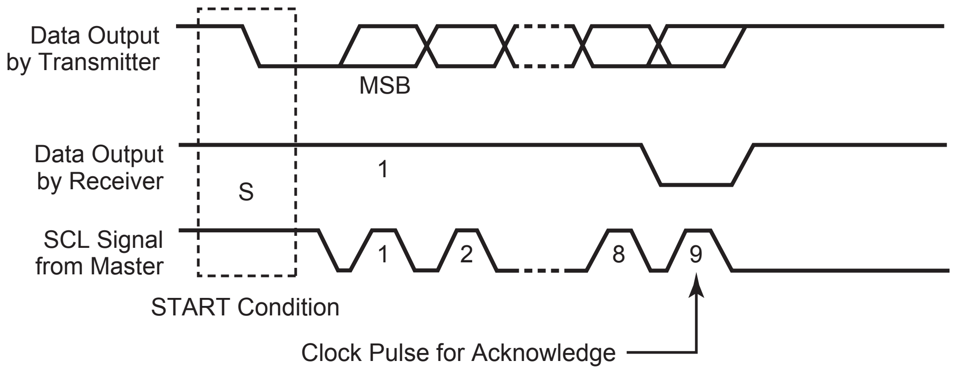

Figure 35.I^{2}C Acknowledge

Clock Synchronization

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

Clock synchronization is performed using the wired AND connection of the I^{2}C interfaces to the SCL line, meaning that a High-to-Low transition on the SCL line causes the relevant devices to start counting from their Low period. When a device clock goes Low, it holds the SCL line in that state until the clock High state is reached. See Figure 36 on page 145. The Low-to-High transition of this clock, however, may not change the state of the SCL line if another clock is still within its Low period. The SCL line is held Low by the device with the longest Low period. Devices with shorter Low periods enter a High wait-state during this time.

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

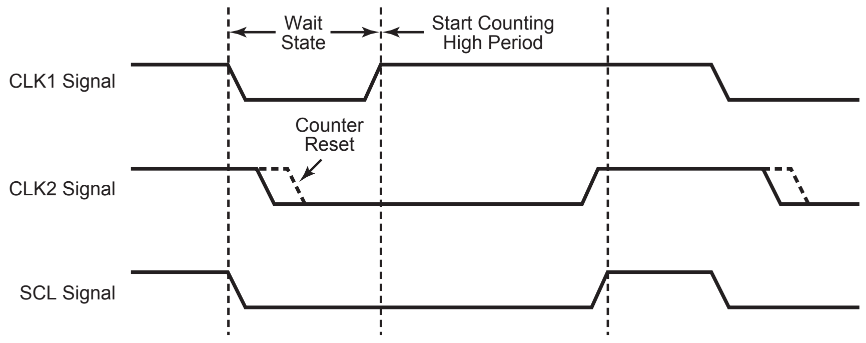

Figure 36.Clock Synchronization In I^2C Protocol

Arbitration

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

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

If a master also incorporates a slave function and it loses arbitration during the addressing stage, it's possible that the winning master is trying to address it. The losing master must switch over immediately to its slave-receiver mode. Figure 36 displays the arbitration procedure for two masters. Of course, more may be involved (depending on how many masters are connected to the bus). The moment there is a difference between the internal data level of the master generating DATA 1 and the actual level on the SDA line, its data output is switched off, which means that a High output level is then connected to the bus. As a result, the data transfer initiated by the winning master is not affected. Because control of the I^2C bus is decided solely on the address and data sent by competing masters, there is no central master, nor any order of priority on the bus.

Special attention must be paid if, during a serial transfer, the arbitration procedure is still in progress at the moment when a repeated START condition or a STOP condition is transmitted to the I^2C bus. If it is possible for such a situation to occur, the masters involved must send this repeated START condition or STOP condition at the same position in the format frame.

ZDI Bus Control Register

The ZDI Bus Control register controls bus requests during DEBUG mode. It enables or disables bus acknowledge in ZDI DEBUG mode and allows ZDI to force assertion of the $\overline{\text{BUSACK}}$ signal. This register should only be written during ZDI DEBUG mode (that is, following a BREAK). See Table 100.

Table 100. ZDI Bus Control Register(ZDI_BUS_CTL = 17h in the ZDI Register Write Only Address Space)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	0	0
CPU Access	W	W	W	W	W	W	W	W
Note: W = Write Only.								

Bit Position	**Value**	**Description**
7 ZDI_BUSAK_EN	0	Bus requests by external peripherals using the $\overline{\text{BUSREQ}}$ pin are ignored. The bus acknowledge signal, $\overline{\text{BUSACK}}$, is not asserted in response to any bus requests.
	1	Bus requests by external peripherals using the $\overline{\text{BUSREQ}}$ pin are accepted. A bus acknowledge occurs at the end of the current ZDI operation. The bus acknowledge is indicated by asserting the $\overline{\text{BUSACK}}$ pin in response to a bus request.
6 ZDI_BUSAK	0	Deassert the bus acknowledge pin ($\overline{\text{BUSACK}}$) to return control of the address and data buses back to ZDI.
	1	Assert the bus acknowledge pin ($\overline{\text{BUSACK}}$) to pass control of the address and data buses to an external peripheral.
[5:0]	000000	Reserved.

Instruction Store 4:0 Registers

The ZDI Instruction Store registers are located in the ZDI Register Write Only address space. They can be written with instruction data for direct execution by the CPU. When the ZDI_IS0 register is written, the eZ80F92 device exits the ZDI BREAK state and executes a single instruction. The Op Codes and operands for the instruction come from these Instruction Store registers. The Instruction Store Register 0 is the first byte fetched, followed by Instruction Store registers 1, 2, 3, and 4, as necessary. Only the bytes the processor requires to execute the instruction must be stored in these registers. Some CPU instructions, when combined with the MEMORY mode suffixes (.SIS, .SIL, .LIS, or .LIL), require 6 bytes to operate. These 6-byte instructions cannot be executed directly using the ZDI Instruction Store registers. See Table 101.

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

Table 101. Instruction Store 4:0 Registers(ZDI_IS4 = 21h, ZDI_IS3 = 22h, ZDI_IS2 = 23h, ZDI_IS1 = 24h, and ZDI_IS0 = 25h in the ZDI Register Write Only Address Space)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	X	X	X	X	X	X	X	X
CPU Access	W	W	W	W	W	W	W	W
Note: X = Undefined; W = Write.								

Bit Position	**Value**	**Description**
[7:0] ZDI_IS4, ZDI_IS3, ZDI_IS2, ZDI_IS1, or ZDI_IS0	00h–FFh	These registers contain the Op Codes and operands for immediate execution by the CPU following a Write to ZDI_IS0. The ZDI_IS0 register contains the first Op Code of the instruction. The remaining ZDI_ISx registers contain any additional Op Codes or operand dates required for execution of the required instruction.

eZ80® Product ID Revision Register

The eZ80 Product ID Revision register identifies the current revision of the eZ80F92 product. See Table 105.

Table 105. eZ80 Product ID Revision Register(ZDI_ID_REV = 02h in the ZDI Register Read Only Address Space)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	X	X	X	X	X	X	X	X
CPU Access	R	R	R	R	R	R	R	R
Note: X = Undetermined; R = Read Only.								

Bit Position	**Value**	**Description**
[7:0] ZDI_ID_REV	00h–FFh	Identifies the current revision of the eZ80F92 product.

ZDI Status Register

The ZDI Status register provides current information about the eZ80F92 device. See Table 106.

Table 106. ZDI Status Register(ZDI_STAT = 03h in the ZDI Register Read Only Address Space)

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

Bit Position	**Value**	**Description**
7 ZDI_ACTIVE	0	The CPU is not functioning in ZDI mode.
	1	The CPU is currently functioning in ZDI mode.
6	0	Reserved.
5 HALT_SLP	0	The CPU is not currently in HALT or SLEEP mode.
	1	The CPU is currently in HALT or SLEEP mode.

Writes to the same row without first erasing it. Otherwise, the burden is on software to ensure that the 16 ms maximum cumulative programming time between erasures is not exceeded for a row.

Memory Write

A single-byte memory Write operation uses the address bus and data bus of the eZ80F92 device for programming a single data byte to Flash. While the CPU executes a LOAD instruction, the Flash controller asserts the internal WAIT signal to stall the CPU until the Write is complete. A single-byte Write takes between 66 µs and 85 µs to complete. Programming an entire row using memory Writes therefore takes at most 10.8 ms. This time does not include time required by the CPU to transfer data to the registers which is a function of the instructions employed and the system clock frequency.

The memory Write function does not support multibyte row programming. As memory Writes are self-timed, they can be performed back-to-back without any necessity for polling or interrupts.

Erasing Flash Memory

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

Mass Erase

Performing a Mass Erase operation on Flash memory erases all bits in Flash, including the Information Page. This self-timed operation takes approximately 200 ms to complete.

Page Erase

The smallest erasable unit in Flash memory is a page. Which of the main Flash memory pages or the single Information Page is to be erased is determined by the setting of the FLASH_PAGE register. This self-timed operation takes approximately 10 ms to complete.

Flash Control Registers

The Flash register interface contains all the registers used in Flash memory.
The definitions as follows describe each register.

eZ80® CPU Instruction Set

Table 125 on page 208 through Table 134 on page 208 indicate the eZ80 CPU instructions available for use with the eZ80F92 device. The instructions are grouped by class. More detailed information is available in the *eZ80® CPU User Manual (UM0077)*.

Table 125. Arithmetic Instructions

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

Table 126. Bit Manipulation Instructions

Mnemonic	Instruction
BIT	Bit Test
RES	Reset Bit
SET	Set Bit

Table 127. Block Transfer and Compare Instructions

Mnemonic	Instruction
CPD (CPDR)	Compare and Decrement (with Repeat)
CPI (CPIR)	Compare and Increment (with Repeat)

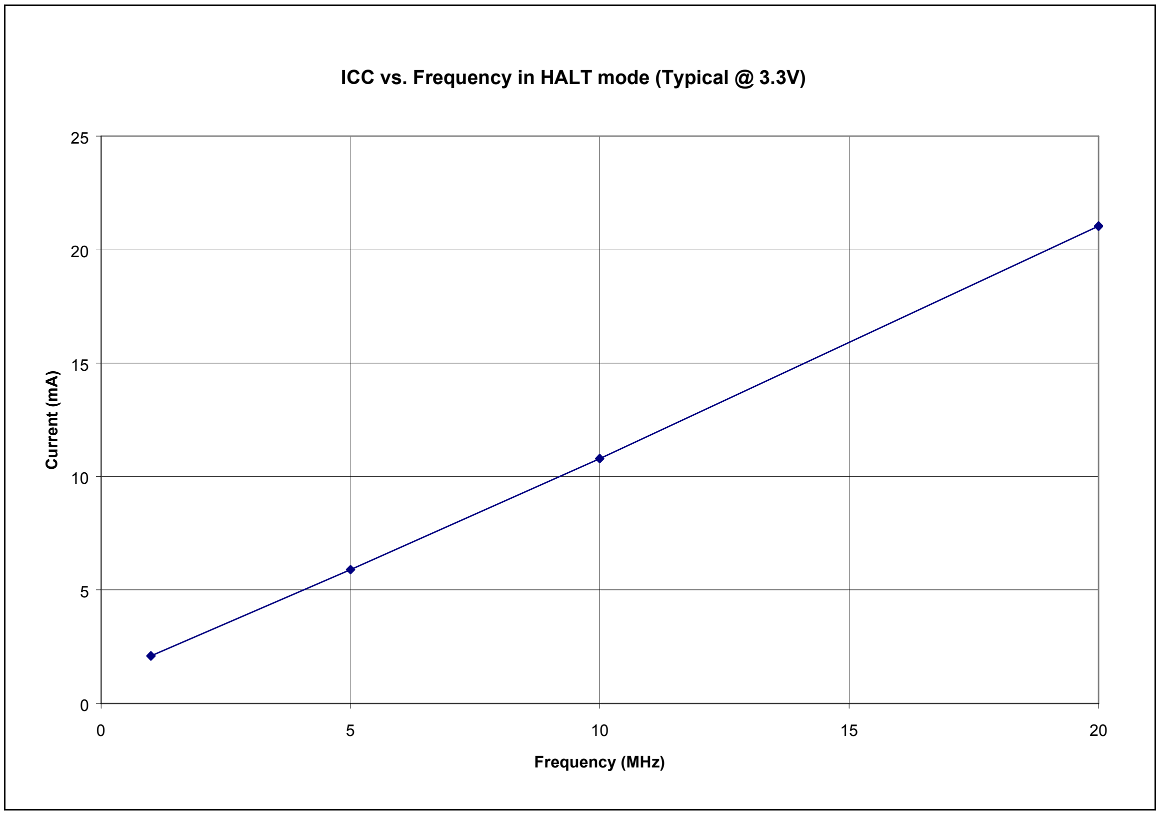

Figure 56. I_{CC} Versus Frequency in HALT Mode

Table 148. External Read Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
T_1	Clock Rise to ADDR Valid Delay	—	13
T_2	Clock Rise to ADDR Hold Time	2.0	—
T_3	Input DATA Valid to Clock Rise Setup Time	1.0	—
T_4	Clock Rise to DATA Hold Time	2.0	—
T_5	Clock Rise to $\overline{\text{CSx}}$ Assertion Delay	2.0	19.0
T_6	Clock Rise to $\overline{\text{CSx}}$ Deassertion Delay	2.0	18.0
T_7	Clock Rise to $\overline{\text{MREQ}}$ Assertion Delay	2.0	16.0
T_8	Clock Rise to $\overline{\text{MREQ}}$ Deassertion Delay	2.0	16.0
T_9	Clock Rise to $\overline{\text{RD}}$ Assertion Delay	2.0	16.0
T_{10}	Clock Rise to $\overline{\text{RD}}$ Deassertion Delay	2.0	16.0

External Memory Write Timing

Figure 59 and Table 149 on page 232 display the timing for external writes.

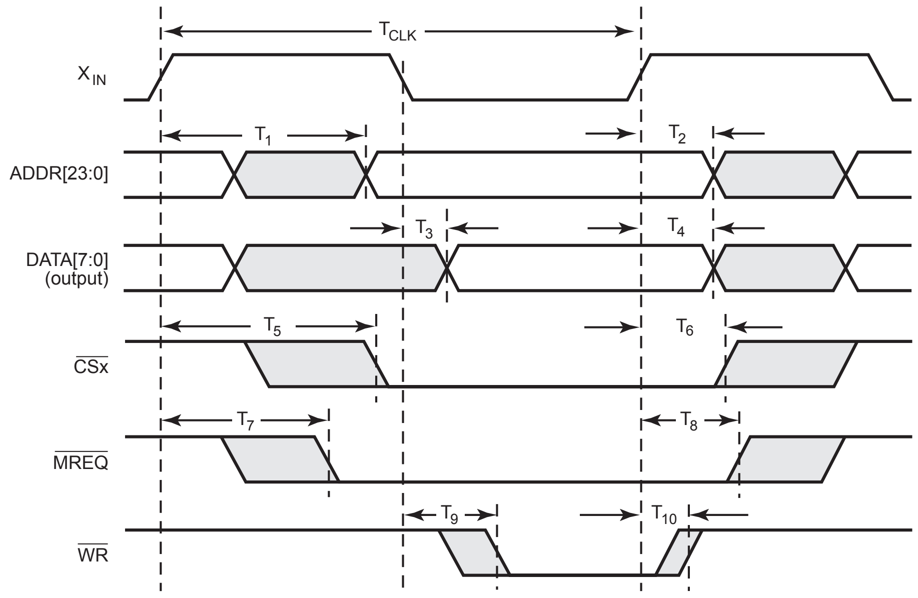

Figure 59.External Memory Write Timing