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	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	19
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 5x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0422sj020eg

Table 24. Interrupt Vectors in Order of Priority

Priority	Program Memory Vector Address	Interrupt Source
Highest	0002H	Reset (not an interrupt)
	0004H	WDT (see the Watchdog Timer chapter on page 70)
	0006H	Illegal Instruction Trap (not an interrupt)
	0008H	Reserved
	000AH	Timer 1
	000CH	Timer 0
	000EH	UART 0 receiver
	0010H	UART 0 transmitter
	0012H	I^2C
	0014H	SPI
	0016H	ADC
	0018H	Port A7, rising or falling input edge
	001AH	Port A6, rising or falling input edge
	001CH	Port A5, rising or falling input edge
	001EH	Port A4, rising or falling input edge
	0020H	Port A3, rising or falling input edge
	0022H	Port A2, rising or falling input edge
	0024H	Port A1, rising or falling input edge
	0026H	Port A0, rising or falling input edge
	0028H	Reserved
	002AH	Reserved
	002CH	Reserved
	002EH	Reserved
	0030H	Port C3, both input edges
	0032H	Port C2, both input edges
	0034H	Port C1, both input edges
Lowest	0036H	Port C0, both input edges

- Writing a 1 to the IRQE bit in the Interrupt Control Register

Interrupts are globally disabled by any of the following actions:

- Execution of a Disable Interrupt (DI) instruction
- eZ8 CPU acknowledgement of an interrupt service request from the interrupt controller
- Writing a 0 to the IRQE bit in the Interrupt Control Register
- Reset
- Execution of a trap instruction
- Illegal instruction trap

Interrupt Vectors and Priority

The interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority, and Level 1 is the lowest priority. If all of the interrupts were enabled with identical interrupt priority (all as Level 2 interrupts), then interrupt priority would be assigned from highest to lowest as specified in Table 24. Level 3 interrupts always have higher priority than Level 2 interrupts which in turn always have higher priority than Level 1 interrupts. Within each interrupt priority level (Level 1, Level 2, or Level 3), priority is assigned as specified in Table 24. A Reset, WDT interrupt (if enabled), and an Illegal Instruction Trap will always have the highest priority.

Interrupt Assertion

Interrupt sources assert their interrupt requests for only a single system clock period (single pulse). When the interrupt request is acknowledged by the eZ8 CPU, the corresponding bit in the Interrupt Request Register is cleared until the next interrupt occurs. Writing a 0 to the corresponding bit in the Interrupt Request Register likewise clears the interrupt request.

Caution: Zilog recommends not using a coding style that clears bits in the Interrupt Request registers. All incoming interrupts received between execution of the first LDX command and the final LDX command are lost. See Example 1, which follows.

Example 1. A poor coding style that can result in lost interrupt requests:

```
LDX r0, IRQ0
AND r0, MASK
LDX IRQ0, r0
```

value matches the PWM value, the Timer Output toggles. The timer continues counting until it reaches the reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes.

If the TPOL bit in the Timer Control Register is set to 1, the Timer Output signal begins as a High (1) and then transitions to a Low (0) when the timer value matches the PWM value. The Timer Output signal returns to a High (1) after the timer reaches the reload value and is reset to 0001H.

If the TPOL bit in the Timer Control Register is set to 0, the Timer Output signal begins as a Low (0) and then transitions to a High (1) when the timer value matches the PWM value. The Timer Output signal returns to a Low (0) after the timer reaches the reload value and is reset to 0001H.

Observe the following procedure for configuring a timer for PWM Mode and initiating the PWM operation:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for PWM Mode
   - Set the prescale value
   - Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output alternate function
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H). This only affects the first pass in PWM Mode. After the first timer reset in PWM Mode, counting always begins at the reset value of 0001H.
3. Write to the PWM High and Low Byte registers to set the PWM value.
4. Write to the Timer Reload High and Low Byte registers to set the reload value (PWM period). The reload value must be greater than the PWM value.
5. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control Register to enable the timer and initiate counting.

The PWM period is calculated using the following equation.

$$\text{PWM Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Bit	Description (Continued)
[6] TPOL (cont'd.)	**GATED Mode** 0 = Timer counts when the Timer Input signal is High (1) and interrupts are generated on the falling edge of the Timer Input. 1 = Timer counts when the Timer Input signal is Low (0) and interrupts are generated on the rising edge of the Timer Input. **CAPTURE/COMPARE Mode** 0 = Counting is started on the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal. 1 = Counting is started on the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal.
[5:3] PRES	**Prescale Value** The timer input clock is divided by 2^{PRES}, where PRES is set from 0 to 7. The prescaler is reset each time the timer is disabled to ensure proper clock division each time the timer is restarted. 000 = Divide by 1. 001 = Divide by 2. 010 = Divide by 4. 011 = Divide by 8. 100 = Divide by 16. 101 = Divide by 32. 110 = Divide by 64. 111 = Divide by 128.
[2:0] TMODE	**Timer Mode** 000 = ONE-SHOT Mode. 001 = CONTINUOUS Mode. 010 = COUNTER Mode. 011 = PWM Mode. 100 = CAPTURE Mode. 101 = COMPARE Mode. 110 = GATED Mode. 111 = CAPTURE/COMPARE Mode.

Table 47. Watchdog Timer Approximate Time-Out Delays

WDT Reload Value (Hex)	WDT Reload Value (Decimal)	Approximate Time-Out Delay (with 10 kHz Typical WDT Oscillator Frequency)	
		Typical	Description
000004	4	400 µs	Minimum time-out delay
FFFFFF	16,777,215	1677.5 s	Maximum time-out delay

Watchdog Timer Refresh

When first enabled, the WDT is loaded with the value in the WDT Reload registers. The WDT then counts down to 000000H unless a WDT instruction is executed by the eZ8 CPU. Execution of the WDT instruction causes the downcounter to be reloaded with the WDT reload value stored in the WDT Reload registers. Counting resumes following the reload operation.

When Z8 Encore! XP® F0822 Series device is operating in DEBUG Mode (using the OCD), the WDT is continuously refreshed to prevent spurious WDT time-outs.

Watchdog Timer Time-Out Response

The WDT times out when the counter reaches 000000H. A WDT time-out generates either an Interrupt or a Reset. The WDT_RES option bit determines the time-out response of the WDT. For information regarding programming of the WDT_RES option bit, see the Option Bits chapter on page 155.

WDT Interrupt in Normal Operation

If configured to generate an interrupt when a time-out occurs, the WDT issues an interrupt request to the interrupt controller and sets the WDT status bit in the WDT Control Register. If interrupts are enabled, the eZ8 CPU responds to the interrupt request by fetching the WDT interrupt vector and executing the code from the vector address. After time-out and interrupt generation, the WDT counter rolls over to its maximum value of FFFFFH and continues counting. The WDT counter is not automatically returned to its reload value.

WDT Reset in STOP Mode

If enabled in STOP Mode and configured to generate a Reset when a time-out occurs and the device is in STOP Mode, the WDT initiates a Stop Mode Recovery. Both the WDT status bit and the stop bit in the WDT Control Register is set to 1 following the WDT time-out in STOP Mode. For more information, see the Reset and Stop Mode Recovery chapter on page 21. Default operation is for the WDT and its RC oscillator to be enabled during STOP Mode.

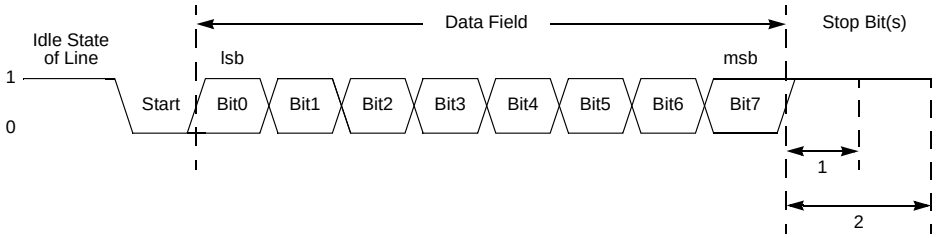

Figure 12. UART Asynchronous Data Format without Parity

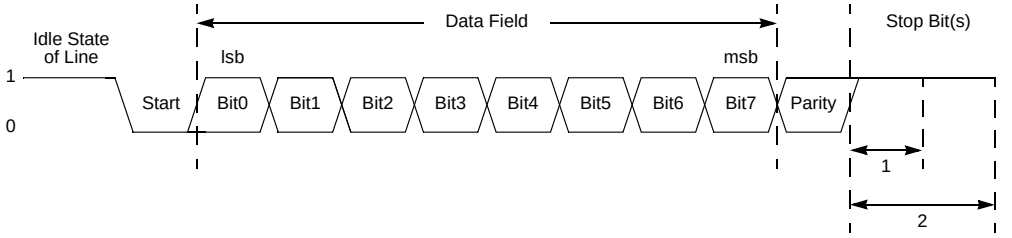

Figure 13. UART Asynchronous Data Format with Parity

Transmitting Data using Polled Method

Observe the following procedure to transmit data using polled method of operation:

1. Write to the UART Baud Rate High Byte and Low Byte registers to set the required baud rate.
2. Enable the UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. If MULTIPROCESSOR Mode is required, write to the UART Control 1 Register to enable multiprocessor (9-bit) mode functions.
   - Set the Multiprocessor Mode Select (MPEN) to enable MULTIPROCESSOR Mode.
4. Write to the UART Control 0 Register to:
   - Set the transmit enable bit (TEN) to enable the UART for data transmission
   - If parity is required, and MULTIPROCESSOR Mode is not enabled, set the parity enable bit (PEN) and select either even or odd parity (PSEL).

The UART is now configured for interrupt-driven data reception. When the UART Receiver Interrupt is detected, the associated ISR performs the following operations:

1. Check the UART Status 0 Register to determine the source of the interrupt, whether error, break or received data.
2. If the interrupt was due to data available, read the data from the UART Receive Data Register. If operating in MULTIPROCESSOR (9-Bit) Mode, further actions may be required depending on the Multiprocessor Mode bits MPMD[1:0].
3. Clear the UART Receiver Interrupt in the applicable Interrupt Request Register.
4. Execute the IRET instruction to return from the ISR and await more data.

Clear To Send Operation

The CTS pin, if enabled by the CTSE bit of the UART Control 0 Register, performs flow control on the outgoing transmit datastream. The Clear To Send ($\overline{\text{CTS}}$) input pin is sampled one system clock before beginning any new character transmission. To delay transmission of the next data character, an external receiver must deassert $\overline{\text{CTS}}$ at least one system clock cycle before a new data transmission begins. For multiple character transmissions, this would be done during stop bit transmission. If $\overline{\text{CTS}}$ deasserts in the middle of a character transmission, the current character is sent completely.

Multiprocessor (9-Bit) Mode

The UART features a MULTIPROCESSOR (9-Bit) Mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In MULTIPROCESSOR Mode (also referred to as 9-bit mode), the multiprocessor bit is transmitted following the 8 bits of data and immediately preceding the stop bit(s); this character format is shown in Figure 14.

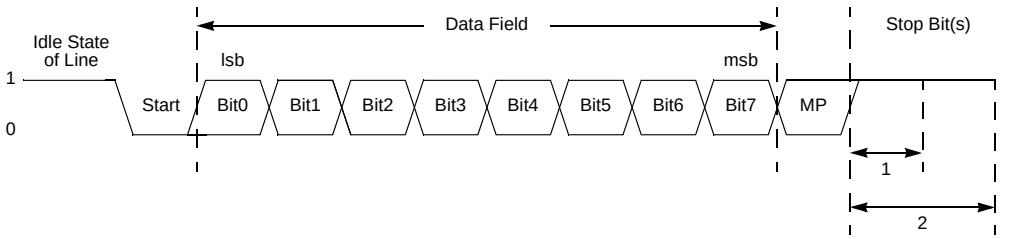

Figure 14. UART Asynchronous Multiprocessor Mode Data Format

12. The I2C Controller loads the contents of the I2C Shift Register with the contents of the I2C Data Register.
13. The I2C Controller shifts the data out of using the SDA signal. After the first bit is sent, the transmit interrupt is asserted.
14. If more bytes remain to be sent, return to Step 9.
15. Software responds by setting the stop bit of the I2C Control Register (or start bit to initiate a new transaction). In the STOP case, software clears the TXI bit of the I2C Control Register at the same time.
16. The I2C Controller completes transmission of the data on the SDA signal.
17. The slave can either Acknowledge or Not Acknowledge the last byte. Because either the stop or start bit is already set, the NCKI interrupt does not occur.
18. The I2C Controller sends the stop (or restart) condition to the I2C bus. The stop or start bit is cleared.

Address-Only Transaction with a 10-Bit Address

In situations in which software must determine if a slave with a 10-bit address is responding without sending or receiving data, a transaction is performed which only consists of an address phase. Figure 28 displays this *address only* transaction to determine if a slave with a 10-bit address will acknowledge.

As an example, this transaction is used after a write has been executed to an EEPROM to determine when the EEPROM completes its internal write operation and is again responding to I2C transactions. If the slave does not acknowledge, the transaction is repeated until the slave is able to acknowledge.

S	Slave Address 1st Seven Bits	W = 0	A/A	Slave Address 2nd Byte	A/A

Figure 28. 10-Bit Address Only Transaction Format

Observe the following procedure for an address-only transaction to a 10-bit addressed slave:

1. Software asserts the IEN bit in the I2C Control Register.
2. Software asserts the TXI bit of the I2C Control Register to enable transmit interrupts.
3. The I2C interrupt asserts, because the I2C Data Register is empty (TDRE = 1).
4. Software responds to the TDRE interrupt by writing the first slave address byte. The least-significant bit must be 0 for the write operation.
5. Software asserts the start bit of the I2C Control Register.

5. The I2C Controller shifts the address and read bit out the SDA signal.
6. If the I2C Slave acknowledges the address by pulling the SDA signal Low during the next High period of SCL, the I2C Controller sets the ACK bit in the I2C Status Register. Continue to Step 7.

   If the slave does not acknowledge, the Not Acknowledge interrupt occurs (NCKI bit is set in the Status Register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the stop bit and clearing the TXI bit. The I2C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remainder of this sequence.
7. The I2C Controller shifts in the byte of data from the I2C Slave on the SDA signal. The I2C Controller sends a Not Acknowledge to the I2C Slave if the NAK bit is set (last byte), else it sends an Acknowledge.
8. The I2C Controller asserts the Receive interrupt (RDRF bit set in the Status Register).
9. Software responds by reading the I2C Data Register which clears the RDRF bit. If there is only one more byte to receive, set the NAK bit of the I2C Control Register.
10. If there are more bytes to transfer, return to Step 7.
11. After the last byte is shifted in, a Not Acknowledge interrupt is generated by the I2C Controller.
12. Software responds by setting the stop bit of the I2C Control Register.
13. A stop condition is sent to the I2C Slave, the stop and NCKI bits are cleared.

Read Transaction with a 10-Bit Address

Figure 31 displays the read transaction format for a 10-bit addressed slave. The shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

S	Slave Address 1st 7 bits	W=0	A	Slave Address 2nd Byte	A	S	Slave Address 1st 7 bits	R=1	A	Data	A	Data	A	P

Figure 31. Receive Data Format for a 10-Bit Addressed Slave

The first seven bits transmitted in the first byte are 11110XX. The two XX bits are the two most significant bits of the 10-bit address. The lowest bit of the first byte transferred is the write control bit.

Observe the following procedure for the data transfer procedure for a read operation to a 10-bit addressed slave:

I²C Data Register

The I²C Data Register, shown in Table 71, holds the data that is to be loaded into the I²C Shift Register during a write to a slave. This register also holds data that is loaded from the I²C Shift Register during a read from a slave. The I²C Shift Register is not accessible in the Register File address space, but is used only to buffer incoming and outgoing data.

Table 71. I²C Data Register (I2CDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	0							
R/W	R/W							
Address	F50H							

I²C Status Register

The read-only I²C Status Register, shown in Table 72, indicates the status of the I²C Controller.

Table 72. I²C Status Register (I2CSTAT)

Bit	7	6	5	4	3	2	1	0
Field	TDRE	RDRF	ACK	10B	RD	TAS	DSS	NCKI
RESET	1	0						
R/W	R							
Address	F51H							

Bit	Description
[7] TDRE	**Transmit Data Register Empty** When the I²C Controller is enabled, this bit is 1 when the I²C Data Register is empty. When this bit is set, an interrupt is generated if the TXI bit is set, except when the I²C Controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit is cleared by writing to the I2CDATA Register.
[6] RDRF	**Receive Data Register Full** This bit is set = 1 when the I²C Controller is enabled and the I²C Controller has received a byte of data. When asserted, this bit causes the I²C Controller to generate an interrupt. This bit is cleared by reading the I²C Data Register (unless the read is performed using execution of the OCD's Read Register command).

Bit	Description (Continued)
[1] FLUSH	**Flush Data** Setting this bit to 1 clears the I2C Data Register and sets the TDRE bit to 1. This bit allows flushing of the I2C Data Register when a Not Acknowledge interrupt is received after the data has been sent to the I2C Data Register. Reading this bit always returns 0.
[0] FILTEN	**I2C Signal Filter Enable** This bit enables low-pass digital filters on the SDA and SCL input signals. These filters reject any input pulse with periods less than a full system clock cycle. The filters introduce a 3-system clock cycle latency on the inputs. 1 = Low-pass filters are enabled. 0 = Low-pass filters are disabled.

I2C Baud Rate High and Low Byte Registers

The I2C Baud Rate High and Low Byte registers, shown in Tables 74 and 75, combine to form a 16-bit reload value, BRG[15:0], for the I2C Baud Rate Generator. When configured as a general purpose timer, the interrupt interval is calculated using the following equation:

Interrupt Interval (s) = System Clock Period (s) ×BRG[15:0]

Table 74. I2C Baud Rate High Byte Register (I2CBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFH							
R/W	R/W							
Address	F53H							

Bit	Description
[7:0] BRH	**I2C Baud Rate High Byte** Most significant byte, BRG[15:8], of the I2C Baud Rate Generator's reload value. **Note:** If the DIAG bit in the I2C Diagnostic Control Register is set to 1, a read of the I2CBRH Register returns the current value of the I2C Baud Rate Counter[15:8].

Flash Memory Address 0000H

Table 90. Option Bits at Flash Memory Address 0000H for 8K Series Flash Devices

Bit	7	6	5	4	3	2	1	0
Field	WDT_RES	WDT_AO	OSC_SEL[1:0]		VBO_AO	RP	Reserved	FWP
RESET	U							
R/W	R/W							
Address	Program Memory 0000H							

Note: U = Unchanged by Reset; R/W = Read/Write.

Bit	Description
[7] WDT_RES	**Watchdog Timer Reset** 0 = Watchdog Timer time-out generates an interrupt request. Interrupts must be globally enabled for the eZ8 CPU to acknowledge the interrupt request. 1 = Watchdog Timer time-out causes a Reset. This setting is the default for unprogrammed (erased) Flash.
[6] WDT_AO	**Watchdog Timer Always On** 0 = Watchdog Timer is automatically enabled upon application of system power. Watchdog Timer can not be disabled. 1 = Watchdog Timer is enabled upon execution of the WDT instruction. After it is enabled, the Watchdog Timer can only be disabled by a Reset or Stop Mode Recovery. This setting is the default for unprogrammed (erased) Flash.
[5:4] OSC_SEL[1:0]	**OSCILLATOR Mode Selection** 00 = On-chip oscillator configured for use with external RC networks (<4MHz). 01 = Minimum power for use with very-low-frequency crystals (32kHz to 1.0MHz). 10 = Medium power for use with medium frequency crystals or ceramic resonators (0.5MHz to 10.0MHz). 11 = Maximum power for use with high-frequency crystals (8.0MHz to 20.0MHz). This setting is the default for unprogrammed (erased) Flash.
[3] VBO_AO	**Voltage Brown-Out Protection Always On** 0 = Voltage Brown-Out Protection is disabled in STOP Mode to reduce total power consumption. 1 = Voltage Brown-Out Protection is always enabled including during STOP Mode. This setting is the default for unprogrammed (erased) Flash.
[2] RP	**Read Protect** 0 = User program code is inaccessible. Limited control features are available through the OCD. 1 = User program code is accessible. All OCD commands are enabled. This setting is the default for unprogrammed (erased) Flash.

Note: *Applies only to the Flash versions of the F0822 Series of devices.

or generate a BRK when its value matches the Program Counter. Because this register is really a down counter, the returned value is inverted when this register is read so the returned result appears to be an up counter. If the device is not in DEBUG Mode, this command returns FFFFH.

```
DBG ← 03H
DBG → ~OCDCNTR[15:8]
DBG → ~OCDCNTR[7:0]
```

Write OCD Control Register (04H). The Write OCD Control Register command writes the data that follows to the OCDCTL Register. When the Read Protect option bit is enabled, the DBGMODE bit (OCDCTL[7]) can only be set to 1, it cannot be cleared to 0 and the only method of putting the device back into normal operating mode is to reset the device.

```
DBG ← 04H
DBG ← OCDCTL[7:0]
```

Read OCD Control Register (05H). The Read OCD Control Register command reads the value of the OCDCTL Register.

```
DBG ← 05H
DBG → OCDCTL[7:0]
```

Write Program Counter (06H). The Write Program Counter command writes the data that follows to the eZ8 CPU's Program Counter. If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the Program Counter values are discarded.

```
DBG ← 06H
DBG ← ProgramCounter[15:8]
DBG ← ProgramCounter[7:0]
```

Read Program Counter (07H). The Read Program Counter command reads the value in the eZ8 CPU's Program Counter. If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, this command returns FFFFH.

```
DBG ← 07H
DBG → ProgramCounter[15:8]
DBG → ProgramCounter[7:0]
```

Write Register (08H). The Write Register command writes data to the Register File. Data can be written 1-256 bytes at a time (256 bytes can be written by setting size to zero). If the device is not in DEBUG Mode, the address and data values are discarded. If the Read Protect option bit is enabled, then only writes to the Flash Control registers are allowed and all other register write data values are discarded.

```
DBG ← 08H
DBG ← {4'h0,Register Address[11:8]}
DBG ← Register Address[7:0]
DBG ← Size[7:0]
DBG ← 1-256 data bytes
```

SPI SLAVE Mode Timing

Figure 52 and Table 110 provide timing information for the SPI SLAVE Mode pins. Timing is shown with SCK rising edge used to source MISO output data, SCK falling edge used to sample MOSI input data.

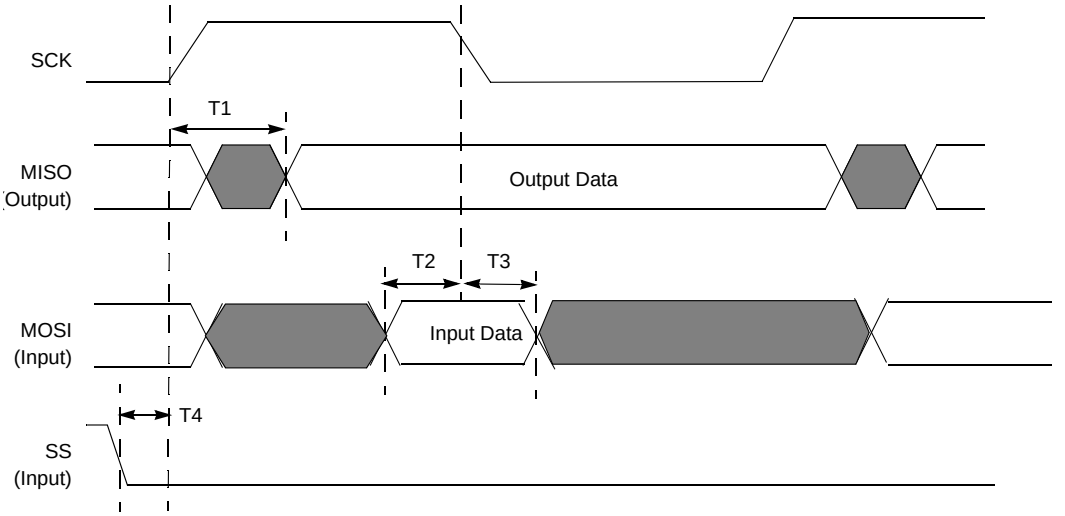

Figure 52. SPI SLAVE Mode Timing

Table 110. SPI SLAVE Mode Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
SPI SLAVE			
T_1	SCK (transmit edge) to MISO output Valid Delay	2 * X_{IN} period	3 * X_{IN} period + 20ns
T_2	MOSI input to SCK (receive edge) Setup Time	0	
T_3	MOSI input to SCK (receive edge) Hold Time	3 * X_{IN} period	
T_4	SS input assertion to SCK setup	1 * X_{IN} period	

resented here by $\overline{DE}$. $\overline{DE}$ asserts after the UART Transmit Data Register has been written. $\overline{DE}$ remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

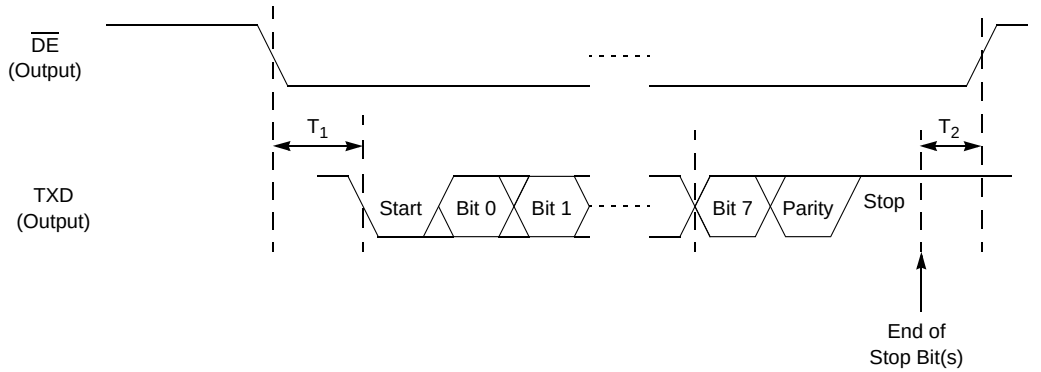

Figure 55. UART Timing without $\overline{CTS}$

Table 113. UART Timing without $\overline{CTS}$

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	$\overline{DE}$ Assertion to TXD Falling Edge (Start) Delay	1 Bit period	1 Bit period + 1 * X_{IN} period
T_2	End of Stop Bit(s) to $\overline{DE}$ Deassertion Delay	1 * X_{IN} period	2 * X_{IN} period

eZ8 CPU Instruction Classes

eZ8 CPU instructions are divided functionally into the following groups:

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Tables 119 through 126 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table as these instruction can be considered as a subset of more than one category. Within these tables, the source operand is identified as 'src', the destination operand is 'dst' and a condition code is 'cc'.

Table 119. Arithmetic Instructions

Mnemonic	Operands	Instruction
ADC	dst, src	Add with Carry
ADCX	dst, src	Add with Carry using Extended Addressing
ADD	dst, src	Add
ADDX	dst, src	Add using Extended Addressing
CP	dst, src	Compare
CPC	dst, src	Compare with Carry
CPCX	dst, src	Compare with Carry using Extended Addressing
CPX	dst, src	Compare using Extended Addressing
DA	dst	Decimal Adjust
DEC	dst	Decrement
DECW	dst	Decrement Word
INC	dst	Increment
INCW	dst	Increment Word

Table 124. Logical Instructions

Mnemonic	Operands	Instruction
AND	dst, src	Logical AND
ANDX	dst, src	Logical AND using Extended Addressing
COM	dst	Complement
OR	dst, src	Logical OR
ORX	dst, src	Logical OR using Extended Addressing
XOR	dst, src	Logical Exclusive OR
XORX	dst, src	Logical Exclusive OR using Extended Addressing

Table 125. Program Control Instructions

Mnemonic	Operands	Instruction
BRK	—	On-Chip Debugger Break
BTJ	p, bit, src, DA	Bit Test and Jump
BTJNZ	bit, src, DA	Bit Test and Jump if Non-Zero
BTJZ	bit, src, DA	Bit Test and Jump if Zero
CALL	dst	Call Procedure
DJNZ	dst, src, RA	Decrement and Jump Non-Zero
IRET	—	Interrupt Return
JP	dst	Jump
JP cc	dst	Jump Conditional
JR	DA	Jump Relative
JR cc	DA	Jump Relative Conditional
RET	—	Return
TRAP	vector	Software Trap

D

E

F

S

T