Zilog - Z8F0822SJ020SC00TR 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	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	8KB (8K 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	0°C ~ 70°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/z8f0822sj020sc00tr

Braces

The curly braces { }, indicate a single register or bus created by concatenating some combination of smaller registers, buses, or individual bits.

- **Example:** The 12-bit register address {`0H`, RP[7:4], R1[3:0]} is composed of a 4-bit hexadecimal value (`0H`) and two 4-bit register values taken from the Register Pointer (RP) and Working Register R1. `0H` is the most significant nibble (4-bit value) of the 12-bit register, and R1[3:0] is the least significant nibble of the 12-bit register.

Parentheses

The parentheses ( ), indicate an indirect register address lookup.

- **Example:** (R1) is the memory location referenced by the address contained in the Working Register R1.

Parentheses/Bracket Combinations

The parentheses ( ), indicate an indirect register address lookup and the square brackets, [ ], indicate a register or bus.

- **Example:** Assume PC[15:0] contains the value `1234h`. (PC [15:0]) then refers to the contents of the memory location at address `1234h`.

Use of the Words *Set*, *Reset* and *Clear*

The word *set* implies that a register bit or a condition contains a logical 1. The words re*set* or *clear* imply that a register bit or a condition contains a logical 0. When either of these terms is followed by a number, the word *logical* cannot be included; however, it is implied.

Notation for Bits and Similar Registers

A field of bits within a register is designated as: Register[*n*:*n*].

- **Example:** ADDR[15:0] refers to bits 15 through bit 0 of the Address.

Use of the Terms *LSB*, *MSB*, *lsb*, and *msb*

In this document, the terms *LSB* and *MSB,* when appearing in upper case, mean *least significant byte* and *most significant byte*, respectively. The lowercase forms, *lsb* and *msb*, mean *least significant bit* and *most significant bit*, respectively.

Use of Initial Uppercase Letters

Initial uppercase letters designate settings and conditions in general text.

- **Example 1**: The receiver forces the SCL line to Low.
- **Example 2:** The Master generates a STOP condition to abort the transfer.

Port A Address
PAADDR (FD0H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port A Address[7:0]
Selects Port Sub-Registers:
00H = No function
01H = Data direction
02H = Alternate function
03H = Output control (open-drain)
04H = High drive enable
05H = STOP mode recovery enable
06H = Pull-up enable
07H-FFH = No function

Port A Control
PACTL (FD1H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port A Control[7:0]
Provides Access to Port Sub-Registers

Port A Input Data
PAIN (FD2H - Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

Port A Input Data [7:0]

Port A Output Data
PAOUT (FD3H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port A Output Data [7:0]

- WDT’s internal RC oscillator continues to operate.
- If enabled, the WDT continues to operate.
- All other on-chip peripherals continue to operate.

The eZ8 CPU can be brought out of HALT mode by any of the following operations:

- Interrupt
- WDT time-out (interrupt or reset)
- Power-On Reset
- Voltage Brownout reset
- External $\overline{\text{RESET}}$ pin assertion

To minimize current in HALT mode, all GPIO pins which are configured as inputs must be driven to one of the supply rails (V_{CC} or GND).

Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register (Table 27) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ2 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU reads the IRQ2 Register to determine if any interrupt requests are pending.

Table 27. Interrupt Request 2 Register (IRQ2)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				PC3I	PC2I	PC1I	PC0I
RESET	0							
R/W	R/W							
ADDR	FC6H							

Reserved—Must be 0

PC*x*I—Port C Pin *x* Interrupt Request
0 = No interrupt request is pending for GPIO Port C pin *x*.
1 = An interrupt request from GPIO Port C pin *x* is awaiting service.
Where *x* indicates the specific GPIO Port C pin number (0 through 3).

IRQ0 Enable High and Low Bit Registers

The IRQ0 Enable High and Low Bit Registers (Table 29 and Table 30) form a priority encoded enabling for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register. Table 28 describes the priority control for IRQ0.

Table 28. IRQ0 Enable and Priority Encoding

IRQ0ENH[*x*]	IRQ0ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

where *x* indicates the register bits from 0 through 7.

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 Option Bits on page 163.

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 Reset and Stop Mode Recovery on page 39. Default operation is for the WDT and its RC oscillator to be enabled during STOP mode.

To minimize power consumption in STOP mode, the WDT and its RC oscillator is disabled in STOP mode. The following sequence configures the WDT to be disabled when the Z8F082x family device enters STOP mode following execution of a STOP instruction:

1. Write `55H` to the Watchdog Timer Control Register (WDTCTL).
2. Write `AAH` to the Watchdog Timer Control Register (WDTCTL).
3. Write `81H` to the Watchdog Timer Control Register (WDTCTL) to configure the WDT and its oscillator to be disabled during STOP mode. Alternatively, write `00H` to the WDTCTL as the third step in this sequence to reconfigure the WDT and its oscillator to be enabled during STOP mode. This sequence only affects WDT operation in STOP mode.

STOP—Stop Mode Recovery Indicator
If this bit is set to 1, a Stop Mode Recovery occurred. If the `STOP` and `WDT` bits are both set to 1, the Stop Mode Recovery occurred due to a WDT time-out. If the `STOP` bit is 1 and the `WDT` bit is 0, the Stop Mode Recovery was not caused by a WDT time-out. This bit is reset by a POR or a WDT time-out that occurred while not in STOP mode. Reading this register also resets this bit.

WDT—Watchdog Timer Time-Out Indicator
If this bit is set to 1, a WDT time-out occurred. A POR resets this pin. A Stop Mode Recovery due a change in an input pin also resets this bit. Reading this register resets this bit.

EXT—External Reset Indicator
If this bit is set to 1, a Reset initiated by the external $\overline{\text{RESET}}$ pin occurred. A POR or a Stop Mode Recovery from a change in an input pin resets this bit. Reading this register resets this bit.

Reserved
These bits are reserved and must be 0.

Watchdog Timer Reload Upper, High and Low Byte Registers

The Watchdog Timer Reload Upper, High and Low Byte (WDTU, WDTH, WDTL) Registers (Table 49 through Table 51) form the 24-bit reload value that is loaded into the WDT, when a WDT instruction executes. The 24-bit reload value is {WDTU[7:0], WDTH[7:0], WDTL[7:0]}. Writing to these registers sets the required Reload Value. Reading from these registers returns the current WDT count value.

! **Caution:** *The 24-bit WDT Reload Value must not be set to a value less than* `000004H`.

Table 49. Watchdog Timer Reload Upper Byte Register (WDTU)

BITS	7	6	5	4	3	2	1	0
FIELD	WDTU							
RESET	1							
R/W	R/W*							
ADDR	FF1H							
R/W*—Read returns the current WDT count value. Write sets the desired Reload Value.								

WDTU—WDT Reload Upper Byte
Most significant byte (MSB), Bits[23:16], of the 24-bit WDT reload value.

5. Check the TDRE bit in the UART Status 0 Register to determine if the Transmit Data Register is empty (indicated by a 1). If empty, continue to step 6. If the Transmit Data Register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data Register becomes available to receive new data.
6. Write the UART Control 1 Register to select the outgoing address bit:
   – Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clear it if sending a data byte.
7. Write data byte to the UART Transmit Data Register. The transmitter automatically transfers data to the Transmit Shift Register and then transmits the data.
8. If required, and multiprocessor mode is enabled, make any changes to the Multiprocessor Bit Transmitter (MPBT) value.
9. To transmit additional bytes, return to step 5.

Transmitting Data Using Interrupt-Driven Method

The UART Transmitter interrupt indicates the availability of the Transmit Data Register to accept new data for transmission. Follow the below steps to configure the UART for interrupt-driven data transmission:

1. Write to the UART Baud Rate High 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. Execute a DI instruction to disable interrupts.
4. Write to the Interrupt Control Registers to enable the UART Transmitter interrupt and set the required priority.
5. 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.
6. Write to the UART Control 0 Register to:
   – Set the transmit enable (TEN) bit to enable the UART for data transmission
   – Enable parity, if required, and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
   – Set or clear the CTSE bit to enable or disable control from the remote receiver through the $\overline{\text{CTS}}$ pin.
7. Execute an EI instruction to enable interrupts.

Receiving Data Using Interrupt-Driven Method

The UART Receiver interrupt indicates the availability of new data (as well as error conditions). Follow the steps below to configure the UART receiver for interrupt-driven operation:

1. Write to the UART Baud Rate High 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. Execute a `DI` instruction to disable interrupts.
4. Write to the Interrupt Control Registers to enable the UART Receiver interrupt and set the required priority.
5. Clear the UART Receiver interrupt in the applicable Interrupt Request Register.
6. Write to the UART Control 1 Register to enable MULTIPROCESSOR (9-bit) mode functions, if desired.
   - Set the Multiprocessor Mode Select (`MPEN`) to enable MULTIPROCESSOR mode.
   - Set the Multiprocessor Mode Bits, `MPMD[1:0],` to select the required address matching scheme.
   - Configure the UART to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for Z8 Encore! XP devices without a DMA block)
7. Write the device address to the Address Compare Register (automatic multiprocessor modes only).
8. Write to the UART Control 0 Register to:
   - Set the receive enable bit (`REN`) to enable the UART for data reception
   - Enable parity, if required, and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
9. Execute an `EI` instruction to enable interrupts.

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

1. Check the UART Status 0 Register to determine the source of the interrupt-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.

Receiver Interrupts

The receiver generates an interrupt when any of the following occurs:

- A data byte is received and is available in the UART Receive Data Register. This interrupt can be disabled independent of the other receiver interrupt sources. The received data interrupt occurs once the receive character is received and placed in the Receive Data Register. Software must respond to this received data available condition before the next character is completely received to avoid an overrun error. In MULTIPROCESSOR mode (`MPEN` = 1), the receive data interrupts are dependent on the multiprocessor configuration and the most recent address byte
- A break is received
- An overrun is detected
- A data framing error is detected

UART Overrun Errors

When an overrun error condition occurs the UART prevents overwriting of the valid data currently in the Receive Data Register. The break detect and overrun status bits are not displayed until the valid data is read.

After the valid data has been read, the UART Status 0 Register is updated to indicate the overrun condition (and Break Detect, if applicable). The `RDA` bit is set to 1 to indicate that the Receive Data Register contains a data byte. However, because the overrun error occurred, this byte cannot contain valid data and should be ignored. The `BRKD` bit indicates if the overrun was caused by a break condition on the line. After reading the status byte indicating an overrun error, the Receive Data Register must be read again to clear the error bits is the UART Status 0 Register. Updates to the Receive Data Register occur only when the next data word is received.

UART Data and Error Handling Procedure

Figure16 on page 99 displays the recommended procedure for UART receiver ISRs.

Baud Rate Generator Interrupts

If the BRG interrupt enable is set, the UART Receiver interrupt asserts when the UART Baud Rate Generator reloads. This action allows the BRG to function as an additional counter if the UART functionality is not employed.

PSEL—Parity Select
0 = Even parity is transmitted and expected on all received data.
1 = Odd parity is transmitted and expected on all received data.

SBRK—Send Break
This bit pauses or breaks data transmission by forcing the Transmit data output to 0. Sending a break interrupts any transmission in progress, so ensure that the transmitter has finished sending data before setting this bit. The UART does not automatically generate a STOP Bit when SBRK is deasserted. Software must time the duration of the Break and the duration of any STOP Bit time desired following the Break.
0 = No break is sent.
1 = The output of the transmitter is zero.

STOP—STOP Bit Select
0 = The transmitter sends one stop bit.
1 = The transmitter sends two stop bits.

LBEN—Loop Back Enable
0 = Normal operation.
1 = All transmitted data is looped back to the receiver.

Table 57. UART Control 1 Register (U0CTL1)

BITS	7	6	5	4	3	2	1	0
FIELD	MPMD[1]	MPEN	MPMD[0]	MPBT	DEPOL	BRGCTL	RDAIRQ	IREN
RESET	0							
R/W	R/W							
ADDR	F43H							

MPMD[1:0]—Multiprocessor Mode
If Multiprocessor (9-bit) mode is enabled,
00 = The UART generates an interrupt request on all received bytes (data and address).
01 = The UART generates an interrupt request only on received address bytes.
10 = The UART generates an interrupt request when a received address byte matches the value stored in the Address Compare Register and on all successive data bytes until an address mismatch occurs.
11 = The UART generates an interrupt request on all received data bytes for which the most recent address byte matched the value in the Address Compare Register.

MPEN—Multiprocessor (9-bit) Enable
This bit is used to enable Multiprocessor (9-bit) mode.
0 = Disable Multiprocessor (9-bit) mode.
1 = Enable Multiprocessor (9-bit) mode.

Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a synchronous interface allowing several SPI-type devices to be interconnected. SPI-compatible devices include EEPROMs, Analog-to-Digital Converters, and ISDN devices. Features of the SPI include:

- Full-duplex, synchronous, and character-oriented communication
- Four-wire interface
- Data transfers rates up to a maximum of one-half the system clock frequency
- Error detection
- Dedicated Baud Rate Generator

The SPI is not available in 20-pin package devices.

Architecture

The SPI is be configured as either a Master (in single or multi-master systems) or a Slave as displayed in Figure 20 through Figure 22.

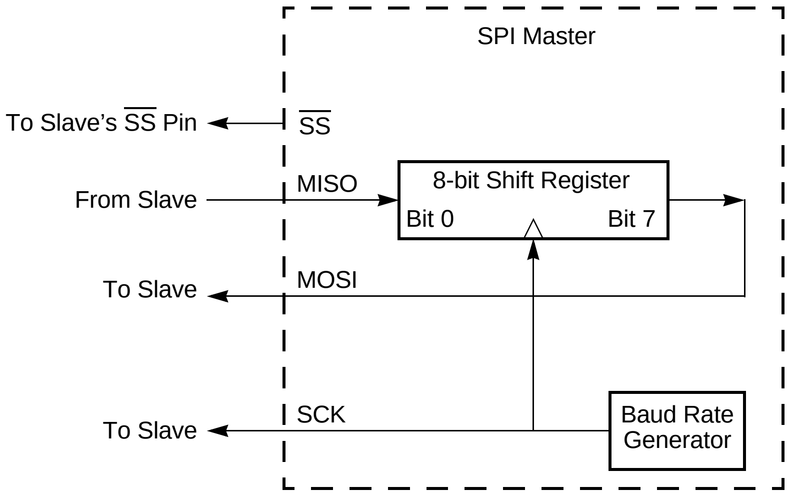

Figure 20. SPI Configured as a Master in a Single Master, Single Slave System

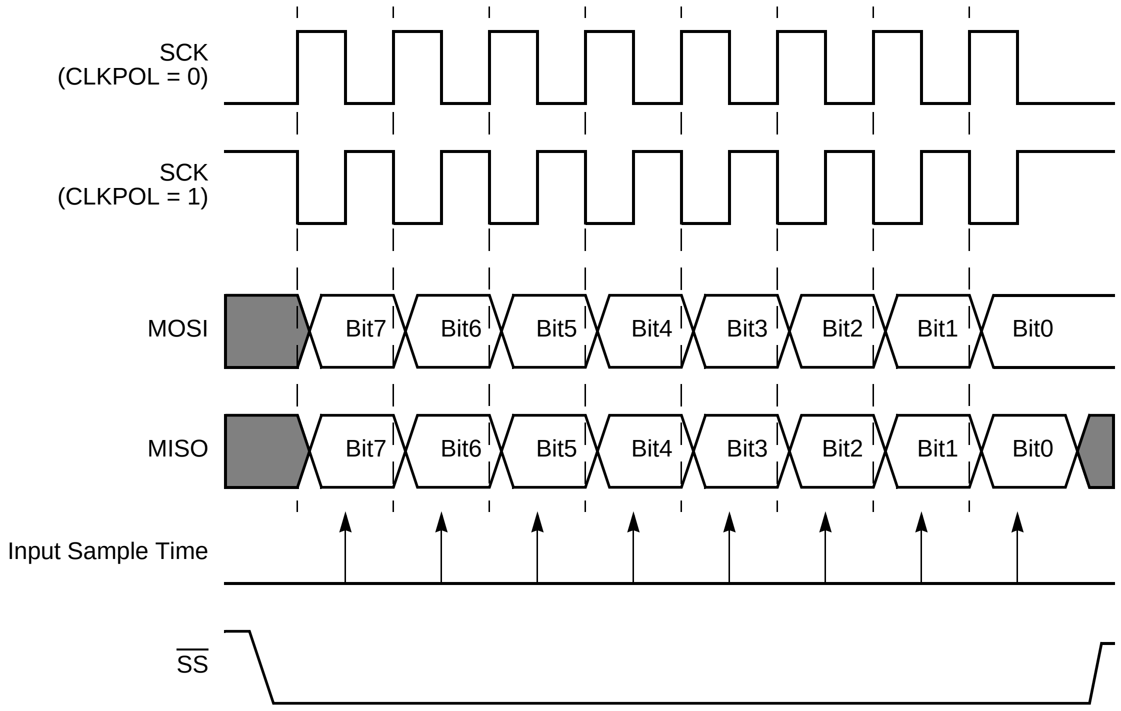

Figure 24. SPI Timing When PHASE is 1

Multi-Master Operation

In a multi-master SPI system, all SCK pins are tied together, all MOSI pins are tied together and all MISO pins are tied together. All SPI pins must then be configured in OPEN-DRAIN mode to prevent bus contention. At any one time, only one SPI device is configured as the Master and all other SPI devices on the bus are configured as Slaves. The Master enables a single Slave by asserting the $\overline{\text{SS}}$ pin on that Slave only. Then, the single Master drives data out its SCK and MOSI pins to the SCK and MOSI pins on the Slaves (including those which are not enabled). The enabled Slave drives data out its MISO pin to the MISO Master pin.

For a Master device operating in a multi-master system, if the $\overline{\text{SS}}$ pin is configured as an input and is driven Low by another Master, the `COL` bit is set to 1 in the SPI Status Register. The `COL` bit indicates the occurrence of a multi-master collision (mode fault error condition).

Slave Operation

The SPI block is configured for SLAVE mode operation by setting the SPIEN bit to 1 and the MMEN bit to 0 in the SPICTL Register and setting the SSIO bit to 0 in the SPIMODE Register. The IRQE, PHASE, CLKPOL, and WOR bits in the SPICTL Register and the

Follow the steps below for a transmit operation on a 10-bit addressed slave:

1. Software asserts the `IEN` bit in the I^{2}C Control Register.
2. Software asserts the `TXI` bit of the I^{2}C Control Register to enable Transmit interrupts.
3. The I^{2}C interrupt asserts because the I^{2}C Data Register is empty.
4. Software responds to the `TDRE` interrupt by writing the first slave address byte to the I^{2}C Data Register. The least-significant bit must be 0 for the write operation.
5. Software asserts the START bit of the I^{2}C Control Register.
6. The I^{2}C Controller sends the START condition to the I^{2}C Slave.
7. The I^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C Data Register.
8. After one bit of address is shifted out by the SDA signal, the Transmit Interrupt is asserted.
9. Software responds by writing the second byte of address into the contents of the I^{2}C Data Register.
10. The I^{2}C Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. If the I^{2}C Slave acknowledges the first address byte by pulling the SDA signal low during the next high period of SCL, the I^{2}C Controller sets the `ACK` bit in the I^{2}C Status register. Continue with step 12.

    If the slave does not acknowledge the first address byte, the I^{2}C Controller sets the `NCKI` bit and clears the ACK bit in the I^{2}C Status register. Software responds to the Not Acknowledge interrupt by setting the STOP and FLUSH bits 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 following steps).
12. The I^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C Data Register.
13. The I^{2}C Controller shifts the second address byte out the SDA signal. After the first bit has been sent, the Transmit Interrupt is asserted.
14. Software responds by writing a data byte to the I^{2}C Data Register.
15. The I^{2}C Controller completes shifting the contents of the shift register on the SDA signal.
16. If the I^{2}C Slave sends an acknowledge by pulling the SDA signal low during the next high period of SCL, the I^{2}C Controller sets the `ACK` bit in the I^{2}C Status register. Continue with step 17.

    If the slave does not acknowledge the second address byte or one of the data bytes, the

ADC Control Register Definitions

ADC Control Register

The ADC Control Register selects the analog input channel and initiates the analog-to-digital conversion.

Table 77. ADC Control Register (ADCCTL)

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	CEN	Reserved	VREF	CONT	ANAIN[3:0]			
RESET	0		1	0				
R/W	R/W							
ADDR	F70H							

CEN—Conversion Enable
0 = Conversion is complete. Writing a 0 produces no effect. The ADC automatically clears this bit to 0 when a conversion has been completed.
1 = Begin conversion. Writing a 1 to this bit starts a conversion. If a conversion is already in progress, the conversion restarts. This bit remains 1 until the conversion is complete.

Reserved—Must be 0

$\overline{\text{VREF}}$
0 = Internal reference generator enabled. The `VREF` pin must be left unconnected or capacitively coupled to analog ground (AVSS).
1 = Internal voltage reference generator disabled. An external voltage reference must be provided through the `VREF` pin.

CONT
0 = SINGLE-SHOT conversion. ADC data is output once at completion of the 5129 system clock cycles.
1 = Continuous conversion. ADC data updated every 256 system clock cycles.

ANAIN—Analog Input Select
These bits select the analog input for conversion. Not all Port pins in this list are available in all packages for Z8 Encore! XP® F0822 Series. See Signal and Pin Descriptions for information regarding the Port pins available with each package style.
Do not enable unavailable analog inputs.
0000 = ANA0
0001 = ANA1
0010 = ANA2
0011 = ANA3
0100 = ANA4

On-Chip Oscillator

Z8 Encore! XP® F0822 Series products feature an on-chip oscillator for use with external crystals with frequencies from 32 kHz to 20 MHz. In addition, the oscillator can support external RC networks with oscillation frequencies up to 4 MHz or ceramic resonators with oscillation frequencies up to 20 MHz. This oscillator generates the primary system clock for the internal eZ8 CPU and the majority of the on-chip peripherals. Alternatively, the X_{IN} input pin can also accept a CMOS-level clock input signal (32 kHz–20 MHz). If an external clock generator is used, the X_{OUT} pin must be left unconnected.

When configured for use with crystal oscillators or external clock drivers, the frequency of the signal on the X_{IN} input pin determines the frequency of the system clock (that is, no internal clock divider). In RC operation, the system clock is driven by a clock divider (divide by 2) to ensure 50% duty cycle.

Operating Modes

Z8 Encore! XP F0822 Series products support 4 different oscillator modes:

- On-chip oscillator configured for use with external RC networks (<4 MHz).
- Minimum power for use with very low frequency crystals (32 kHz to 1.0 MHz).
- Medium power for use with medium frequency crystals or ceramic resonators (0.5 MHz to 10.0 MHz).
- Maximum power for use with high frequency crystals or ceramic resonators (8.0 MHz to 20.0 MHz).

The oscillator mode is selected through user-programmable Option Bits. For more information, see Option Bits on page 163.

Crystal Oscillator Operation

Figure 34 on page 168 displays a recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 20 MHz. Recommended 20 MHz crystal specifications are provided in Table 91 on page 168. Resistor R1 is optional and limits total power dissipation by the crystal. The printed circuit board layout must add no more than 4 pF of stray capacitance to either the X_{IN} or X_{OUT} pins. If oscillation does not occur, reduce the values of capacitors C1 and C2 to decrease loading.

A “reset and stop” function can be achieved by writing `81H` to this register. A “reset and go” function can be achieved by writing `41H` to this register. If the device is in DEBUG mode, a “run” function can be implemented by writing `40H` to this register.

Table 94. OCD Control Register (OCDCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	DBGMODE	BRKEN	DBGACK	BRKLOOP	BRKPC	BRKZRO	Reserved	RST
RESET	0							
R/W	R/W			R				R/W

DBGMODE—Debug Mode
Setting this bit to 1 causes the device to enter DEBUG mode. When in DEBUG mode, the eZ8 CPU stops fetching new instructions. Clearing this bit causes the eZ8 CPU to start running again. This bit is automatically set when a BRK instruction is decoded and Breakpoints are enabled. If the Read Protect Option Bit is enabled, this bit can only be cleared by resetting the device, it cannot be written to 0.
0 = The Z8 Encore! XP F0822 Series device is operating in NORMAL mode.
1 = The Z8 Encore! XP F0822 Series device is in DEBUG mode.

BRKEN—Breakpoint Enable
This bit controls the behavior of the BRK instruction (opcode `00H`). By default, Breakpoints are disabled and the BRK instruction behaves like an NOP instruction. If this bit is set to 1 and a BRK instruction is decoded, the OCD takes action dependent upon the BRKLOOP bit.
0 = BRK instruction is disabled.
1 = BRK instruction is enabled.

DBGACK—Debug Acknowledge
This bit enables the debug acknowledge feature. If this bit is set to 1, then the OCD sends an Debug Acknowledge character (`FFH`) to the host when a Breakpoint occurs.
0 = Debug Acknowledge is disabled.
1 = Debug Acknowledge is enabled.

BRKLOOP—Breakpoint Loop
This bit determines what action the OCD takes when a BRK instruction is decoded if breakpoints are enabled (BRKEN is 1). If this bit is 0, then the DBGMODE bit is automatically set to 1 and the OCD enter DEBUG mode. If BRKLOOP is set to 1, then the eZ8 CPU loops on the BRK instruction.
0 = BRK instruction sets DBGMODE to 1.
1 = eZ8 CPU loops on BRK instruction.

BRKPC—Break when PC == OCDCNTR
If this bit is set to 1, then the OCDCNTR register is used as a hardware breakpoint. When the program counter matches the value in the OCDCNTR register, DBGMODE is

On-Chip Debugger Timing

Figure 50 and Table 107 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4 μs maximum rise and fall time.

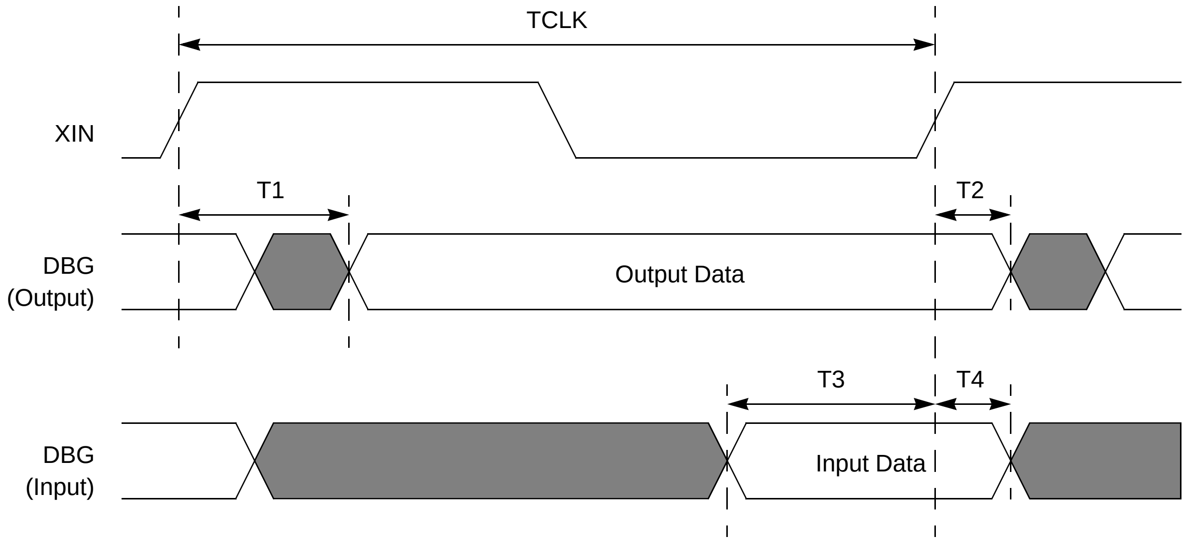

Figure 50. On-Chip Debugger Timing

Table 107. On-Chip Debugger Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
DBG			
T_1	XIN Rise to DBG Valid Delay	–	15
T_2	XIN Rise to DBG Output Hold Time	2	–
T_3	DBG to XIN Rise Input Setup Time	10	–
T_4	DBG to XIN Rise Input Hold Time	5	–
	DBG frequency		System Clock/4

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 118 through Table 125 on page 218 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 118. 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
MULT	dst	Multiply

Table 125. Rotate and Shift Instructions

Mnemonic	Operands	Instruction
BSWAP	dst	Bit Swap
RL	dst	Rotate Left
RLC	dst	Rotate Left through Carry
RR	dst	Rotate Right
RRC	dst	Rotate Right through Carry
SRA	dst	Shift Right Arithmetic
SRL	dst	Shift Right Logical
SWAP	dst	Swap Nibbles

eZ8 CPU Instruction Summary

Table 126 summarizes the eZ8 CPU instructions. The table identifies the addressing modes employed by the instruction, the effect upon the Flags register, the number of CPU clock cycles required for the instruction fetch, and the number of CPU clock cycles required for the instruction execution.

.

Table 126. eZ8 CPU Instruction Summary

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
ADC dst, src	dst ← dst + src + C	r	r	12	*	*	*	*	0	*	2	3
		r	Ir	13							2	4
		R	R	14							3	3
		R	IR	15							3	4
		R	IM	16							3	3
		IR	IM	17							3	4
ADCX dst, src	dst ← dst + src + C	ER	ER	18	*	*	*	*	0	*	4	3
		ER	IM	19							4	3