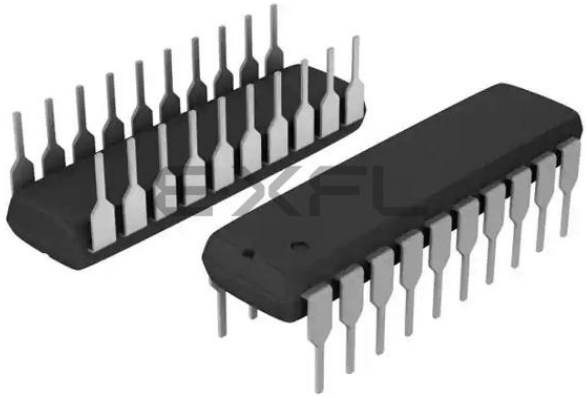

Welcome to E-XFL.COM

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, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	11
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 2x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0821ph020sg

List of Figures

- 0°C to +70°C standard temperature and –40°C to +105°C extended temperature operating ranges

Part Selection Guide

Table 1 identifies the basic features and package styles available for each device within the Z8 Encore! XP® F0822 Series.

Table 1. Z8 Encore! XP® F0822 Series Part Selection Guide

Part Number	Flash (KB)	RAM (KB)	I/O	16-bit Timers with PWM	ADC Inputs	UARTs with IrDA	I^2C	SPI	Package Pin Counts	
									20	28
Z8F0822	8	1	19	2	5	1	1	1		X
Z8F0821	8	1	11	2	2	1	1		X	
Z8F0812	8	1	19	2	0	1	1	1		X
Z8F0811	8	1	11	2	0	1	1		X	
Z8F0422	4	1	19	2	5	1	1	1		X
Z8F0421	4	1	11	2	2	1	1		X	
Z8F0412	4	1	19	2	0	1	1	1		X
Z8F0411	4	1	11	2	0	1	1		X	

Signal and Pin Descriptions

Z8 Encore! XP® F0822 Series products are available in a variety of packages, styles, and pin configurations. This chapter describes the signals and available pin configurations for each of the package styles. For information regarding the physical package specifications, see the Packaging chapter on page 221.

Available Packages

Table 2 identifies the package styles available for each device within the Z8 Encore! XP® F0822 Series.

Table 2. Z8 Encore! XP® F0822 Series Package Options

Part Number	10-Bit ADC	20-Pin SSOP and PDIP	28-Pin SOIC and PDIP
Z8F0822	Yes		X
Z8F0821	Yes	X	
Z8F0812	No		X
Z8F0811	No	X	
Z8F0422	Yes		X
Z8F0421	Yes	X	
Z8F0412	No		X
Z8F0411	No	X	

Pin Configurations

Figures 2 through 5 display the pin configurations for all of the packages available in the Z8 Encore! XP® F0822 Series. See Table 4 on page 13 for a description of the signals.

Note: The analog input alternate functions (ANAx) are not available on Z8 Encore! XP® F0822 Series devices.

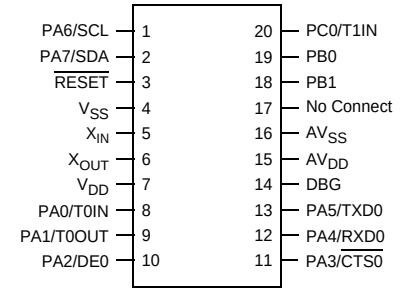

Figure 4. The Z8F0811 and Z8F0411 MCUs in 20-Pin SSOP and PDIP Packages

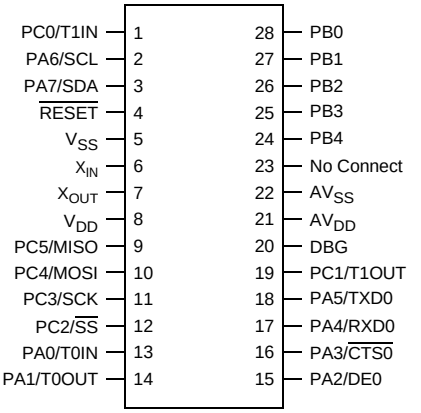

Figure 5. The Z8F0812 and Z8F0412 MCUs in 28-Pin SOIC and PDIP Packages

Bit	Description
[7:0] PWMH, PWML	**Pulse-Width Modulator High and Low Bytes** These two bytes, {PWMH[7:0], PWML[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the PWM output changes state. The PWM output value is set by the TPOL bit in the Timer Control (TxCTL) Register. The TxPWMH and TxPWML registers also store the 16-bit captured timer value when operating in CAPTURE or CAPTURE/COMPARE modes.

Timer 0–3 Control 0 Registers

The Timer 0–3 Control 0 (TxCTL0) registers, shown in Table 45, allow cascading of the Timers.

Table 45. Timer 0–3 Control 0 Registers (TxCTL0)

Bit	7	6	5	4	3	2	1	0
Field	Reserved			CSC	Reserved			
RESET	0							
R/W	R/W							
Address	F06H, F0EH, F16H, F1EH							

Bit	Description
[7:5]	**Reserved** These bits are reserved and must be programmed to 000.
[4] CSC	**Cascade Timers** 0 = Timer Input signal comes from the pin. 1 = For Timer 0, input signal is connected to Timer 1 output. For Timer 1, the input signal is connected to the Timer 0 output.
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

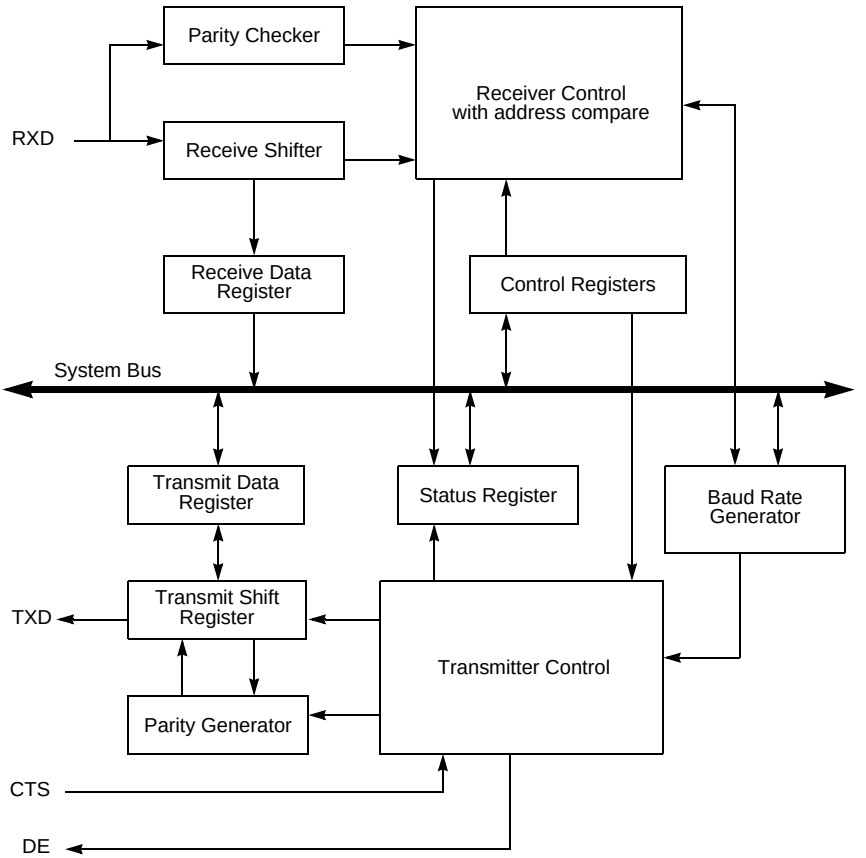

Figure 11. UART Block Diagram

Operation

The UART always transmits and receives data in an 8-bit data format, least-significant bit first. An even or odd parity bit is optionally added to the data stream. Each character begins with an active Low start bit and ends with either 1 or 2 active High stop bits. Figures 12 and 13 display the asynchronous data format used by the UART without parity and with parity, respectively.

- Set or clear the CTSE bit to enable or disable control from the remote receiver using the $\overline{\text{CTS}}$ pin.

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.

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.

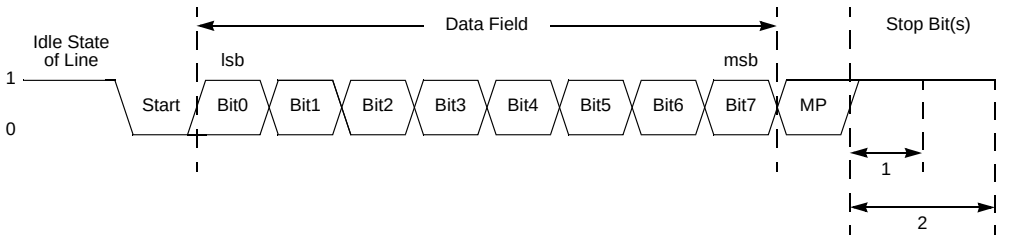

Figure 14. UART Asynchronous Multiprocessor Mode Data Format

SPI Control Register

The SPI Control Register, shown in Table 65, configures the SPI for transmit and receive operations.

Table 65. SPI Control Register (SPICTL)

Bit	7	6	5	4	3	2	1	0
Field	IRQE	STR	BIRQ	PHASE	CLKPOL	WOR	MMEN	SPIEN
RESET	0							
R/W	R/W							
Address	F61H							

Bit	Description
[7] IRQE	**Interrupt Request Enable** 0 = SPI interrupts are disabled. No interrupt requests are sent to the Interrupt Controller. 1 = SPI interrupts are enabled. Interrupt requests are sent to the Interrupt Controller.
[6] STR	**Start an SPI Interrupt Request** 0 = No effect. 1 = Setting this bit to 1 also sets the IRQ bit in the SPI Status Register to 1. Setting this bit forces the SPI to send an interrupt request to the Interrupt Control. This bit can be used by software for a function similar to transmit buffer empty in a UART. Writing a 1 to the IRQ bit in the SPI Status Register clears this bit to 0.
[5] BIRQ	**BRG Timer Interrupt Request** If the SPI is enabled, this bit has no effect. If the SPI is disabled: 0 = BRG timer function is disabled. 1 = BRG timer function and time-out interrupt are enabled.
[4] PHASE	**Phase Select** Sets the phase relationship of the data to the clock. For more information about operation of the PHASE bit, see the SPI Clock Phase and Polarity Control section on page 104.
[3] CLKPOL	**Clock Polarity** 0 = SCK idles Low (0). 1 = SCK idle High (1).
[2] WOR	**Wire-OR (Open-Drain) Mode Enabled** 0 = SPI signal pins not configured for open-drain. 1 = All four SPI signal pins (SCK, $\overline{SS}$, MISO, MOSI) configured for open-drain function. This setting is typically used for multimaster and/or multislave configurations.
[1] MMEN	**SPI MASTER Mode Enable** 0 = SPI configured in SLAVE Mode. 1 = SPI configured in MASTER Mode.
[0] SPIEN	**SPI Enable** 0 = SPI disabled. 1 = SPI enabled.

SPI Status Register

The SPI Status Register, shown in Table 66, indicates the current state of the SPI. All bits revert to their reset state if the SPIEN bit in the SPICTL Register equals 0.

Table 66. SPI Status Register (SPISTAT)

Bit	7	6	5	4	3	2	1	0
Field	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0							1
R/W	R/W*				R			
Address	F62H							
Note: *R/W = read access; write a 1 to clear the bit to 0.								

Bit	Description
[7] IRQ	**Interrupt Request** If SPIEN = 1, this bit is set if the STR bit in the SPICTL Register is set, or upon completion of an SPI Master or Slave transaction. This bit does not set if SPIEN = 0 and the SPI Baud Rate Generator is used as a timer to generate the SPI interrupt. 0 = No SPI interrupt request pending. 1 = SPI interrupt request is pending.
[6] OVR	**Overrun** 0 = An overrun error has not occurred. 1 = An overrun error has been detected.
[5] COL	**Collision** 0 = A multimaster collision (mode fault) has not occurred. 1 = A multimaster collision (mode fault) has been detected.
[4] ABT	**SLAVE Mode Transaction Abort** This bit is set if the SPI is configured in SLAVE Mode, a transaction is occurring and $\overline{SS}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the SPIMODE Register. The IRQ bit also sets, indicating the transaction has completed. 0 = A SLAVE Mode transaction abort has not occurred. 1 = A SLAVE Mode transaction abort has been detected.
[3:2]	**Reserved** These bits are reserved and must be programmed to 00.
[1] TXST	**Transmit Status** 0 = No data transmission currently in progress. 1 = Data transmission currently in progress.
[0] SLAS	**Slave Select** If SPI is enabled as a Slave, then the following bit settings are true: 0 = $\overline{SS}$ input pin is asserted (Low) 1 = $\overline{SS}$ input is not asserted (High). If SPI is enabled as a Master, this bit is not applicable.

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.

16. If the I²C Slave sends an acknowledge 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 17.

    If the slave does not acknowledge the second address byte or one of the data bytes, the I²C Controller sets the NCKI bit and clears the ACK bit in the I²C 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 remainder of this sequence.
17. The I²C Controller shifts the data out by the SDA signal. After the first bit is sent, the transmit interrupt is asserted.
18. If more bytes remain to be sent, return to Step 14.
19. If the last byte is currently being sent, software sets the stop bit of the I²C Control Register (or start bit to initiate a new transaction). In the stop case, software simultaneously clears the TXI bit of the I²C Control Register.
20. The I²C Controller completes transmission of the last data byte on the SDA signal.
21. 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.
22. The I²C Controller sends the stop (or restart) condition to the I²C bus and clears the stop (or start) bit.

Read Transaction with a 7-Bit Address

Figure 30 displays the data transfer format for a read operation to a 7-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	R = 1	A	Data	A	Data	A	P/S

Figure 30. Receive Data Transfer Format for a 7-Bit Addressed Slave

Observe the following procedure for a read operation to a 7-bit addressed slave:

1. Software writes the I²C Data Register with a 7-bit Slave address plus the read bit (=1).
2. Software asserts the start bit of the I²C Control Register.
3. If this transfer is a single byte transfer, Software asserts the NAK bit of the I²C Control Register so that after the first byte of data has been read by the I²C Controller, a Not Acknowledge is sent to the I²C Slave.
4. The I²C Controller sends the start condition.

Bit	Description (Continued)	
[4:0] TXRXSTATE	**Internal State** Value of the internal I^2C state machine.	
	TXRXSTATE	**State Description**
	0_0000	Idle State.
	0_0001	Start State.
	0_0010	Send/Receive data bit 7.
	0_0011	Send/Receive data bit 6.
	0_0100	Send/Receive data bit 5.
	0_0101	Send/Receive data bit 4.
	0_0110	Send/Receive data bit 3.
	0_0111	Send/Receive data bit 2.
	0_1000	Send/Receive data bit 1.
	0_1001	Send/Receive data bit 0.
	0_1010	Data Acknowledge State.
	0_1011	Second half of data Acknowledge State used only for not acknowledge.
	0_1100	First part of stop state.
	0_1101	Second part of stop state.
	0_1110	10-bit addressing: Acknowledge State for 2nd address byte; 7-bit addressing: Address Acknowledge State.
	0_1111	10-bit address: Bit 0 (Least significant bit) of 2nd address byte; 7-bit address: Bit 0 (Least significant bit) (R/W) of address byte.
	1_0000	10-bit addressing: Bit 7 (Most significant bit) of 1st address byte.
	1_0001	10-bit addressing: Bit 6 of 1st address byte.
	1_0010	10-bit addressing: Bit 5 of 1st address byte.
	1_0011	10-bit addressing: Bit 4 of 1st address byte.
	1_0100	10-bit addressing: Bit 3 of 1st address byte.
	1_0101	10-bit addressing: Bit 2 of 1st address byte.
	1_0110	10-bit addressing: Bit 1 of 1st address byte.
	1_0111	10-bit addressing: Bit 0 (R/W) of 1st address byte.
	1_1000	10-bit addressing: Acknowledge state for 1st address byte.
	1_1001	10-bit addressing: Bit 7 of 2nd address byte; 7-bit addressing: Bit 7 of address byte.
	1_1010	10-bit addressing: Bit 6 of 2nd address byte; 7-bit addressing: Bit 6 of address byte.
	1_1011	10-bit addressing: Bit 5 of 2nd address byte; 7-bit addressing: Bit 5 of address byte.
	1_1100	10-bit addressing: Bit 4 of 2nd address byte; 7-bit addressing: Bit 4 of address byte.
	1_1101	10-bit addressing: Bit 3 of 2nd address byte; 7-bit addressing: Bit 3 of address byte.
	1_1110	10-bit addressing: Bit 2 of 2nd address byte; 7-bit addressing: Bit 2 of address byte.
	1_1111	10-bit addressing: Bit 1 of 2nd address byte; 7-bit addressing: Bit 1 of address byte.

For more information about bypassing the Flash Controller, refer to the Third Party Flash Programming Support for Z8 Encore! MCU Application Note (AN0117), available for download at www.zilog.com.

Flash Controller Behavior in Debug Mode

The following changes in behavior of the Flash Controller occur when the Flash Controller is accessed using the OCD:

- The Flash Write Protect option bit is ignored
- The Flash Sector Protect Register is ignored for programming and erase operations
- Programming operations are not limited to the page selected in the Page Select Register
- Bits in the Flash Sector Protect Register can be written to 1 or 0
- The second write of the Page Select Register to unlock the Flash Controller is not necessary
- The Page Select Register is written when the Flash Controller is unlocked
- The Mass Erase command is enabled

Flash Control Register Definitions

This section defines the features of the following Flash Control registers.

Flash Control Register: see page 150

Flash Status Register: see page 151

Page Select Register: see page 152

Flash Sector Protect Register: see page 152

Flash Frequency High and Low Byte Registers: see page 153

Table 92. OCD Baud-Rate Limits

System Clock Frequency (MHz)	Recommended Maximum Baud Rate (Kbps)	Minimum Baud Rate (Kbps)
20.0	2500	39.1
1.0	125.0	1.96
0.032768 (32kHz)	4.096	0.064

If the OCD receives a serial break (nine or more continuous bits Low) the Autobaud Detector/Generator resets. The Autobaud Detector/Generator can then be reconfigured by sending 80H.

OCD Serial Errors

The OCD can detect any of the following error conditions on the DBG pin:

- Serial break (a minimum of nine continuous bits Low)
- Framing error (received stop bit is Low)
- Transmit collision (OCD and host simultaneous transmission detected by the OCD)

When the OCD detects one of these errors, it aborts any command currently in progress, transmits a serial break that is 4096 system clock cycles in duration to the host, and resets the Autobaud Detector/Generator. A framing error or transmit collision can be caused by the host sending a serial break to the OCD. Because of the open-drain nature of the interface, returning a serial break back to the host only extends the length of the serial break if the host releases the serial break early.

The host transmits a serial break on the DBG pin when first connecting to the Z8 Encore! XP® F0822 Series device or when recovering from an error. A serial break from the host resets the Autobaud Generator/Detector but does not reset the OCD Control Register. A serial break leaves the device in DEBUG Mode if that is the current mode. The OCD is held in Reset until the end of the serial break when the DBG pin returns High. Because of the open-drain nature of the DBG pin, the host can send a serial break to the OCD even if the OCD is transmitting a character.

Breakpoints

Execution breakpoints are generated using the BRK instruction (Op Code 00H). When the eZ8 CPU decodes a BRK instruction, it signals the OCD. If breakpoints are enabled, the OCD idles the eZ8 CPU and enters DEBUG Mode. If breakpoints are not enabled, the OCD ignores the BRK signal and the BRK instruction operates as a NOP instruction.

If breakpoints are enabled, the OCD can be configured to automatically enter DEBUG Mode, or to loop on the break instruction. If the OCD is configured to loop on the BRK instruction, then the CPU is still enabled to service DMA and interrupt requests.

The loop on a BRK instruction can be used to service interrupts in the background. For interrupts to be serviced in the background, there cannot be any breakpoints in the ISR. Otherwise, the CPU stops on the breakpoint in the interrupt routine. For interrupts to be serviced in the background, interrupts must also be enabled. Debugging software should not automatically enable interrupts when using this feature, because interrupts are typically disabled during critical sections of code where interrupts should not occur (such as adjusting the stack pointer or modifying shared data).

Software can poll the IDLE bit of the OCDSTAT Register to determine if the OCD is looping on a BRK instruction. When software wants to stop the CPU on the BRK instruction it is looping on, software should not set the DBGMODE bit of the OCDCTL Register. The CPU can have vectored to and be in the middle of an ISR when this bit gets set. Instead, software must clear the BRKLP bit. This allows the CPU to finish the ISR it is in and return the BRK instruction. When the CPU returns to the BRK instruction it was previously looping on, it automatically sets the DBGMODE bit and enter DEBUG Mode.

Software should also note that the majority of the OCD commands are still disabled when the eZ8 CPU is looping on a BRK instruction. The eZ8 CPU must be stopped and the part must be in DEBUG Mode before these commands can be issued.

Breakpoints in Flash Memory

The BRK instruction is Op Code 00H, which corresponds to the fully programmed state of a byte in Flash memory. To implement a Breakpoint, write 00H to the appropriate address, overwriting the current instruction. To remove a Breakpoint, the corresponding page of Flash memory must be erased and reprogrammed with the original data.

OCDCNTR Register

The OCD contains a multipurpose 16-bit counter register. It can be used for the following:

- Count system clock cycles between breakpoints
- Generate a BRK when it counts down to zero
- Generate a BRK when its value matches the Program Counter

When configured as a counter, the OCDCNTR Register starts counting when the OCD leaves DEBUG Mode and stops counting when it enters DEBUG Mode again or when it reaches the maximum count of FFFFH. The OCDCNTR Register automatically resets itself to 0000H when the OCD exits DEBUG Mode if it is configured to count clock cycles between breakpoints.

I²C Timing

Figure 53 and Table 111 provide timing information for I²C pins.

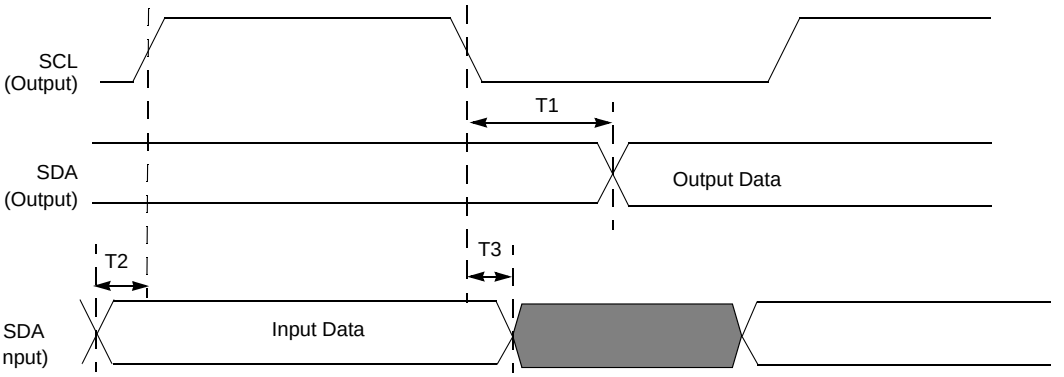

Figure 53. I²C Timing

Table 111. I²C Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
I²C			
T_1	SCL Fall to SDA output delay	SCL period/4	
T_2	SDA Input to SCL rising edge Setup Time	0	
T_3	SDA Input to SCL falling edge Hold Time	0	

Flags Register

The Flags Register contains the status information regarding the most recent arithmetic, logical, bit manipulation or rotate and shift operation. The Flags Register contains six bits of status information that are set or cleared by CPU operations. Four of the bits (C, V, Z and S) can be tested with conditional jump instructions. Two flags (H and D) cannot be tested and are used for Binary-Coded Decimal (BCD) arithmetic.

The two remaining bits, User Flags (F1 and F2), are available as general-purpose status bits. User Flags are unaffected by arithmetic operations and must be set or cleared by instructions. The User Flags cannot be used with conditional Jumps. They are undefined at initial power-up and are unaffected by Reset. Figure 56 illustrates the flags and their bit positions in the Flags Register.

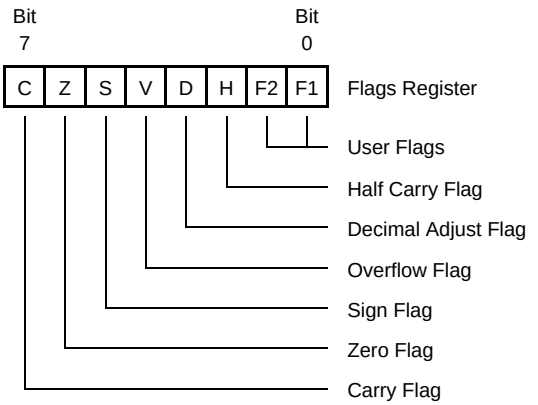

Figure 56. Flags Register

Interrupts, the Software Trap (TRAP) instruction, and Illegal Instruction Traps all write the value of the Flags Register to the stack. Executing an Interrupt Return (IRET) instruction restores the value saved on the stack into the Flags Register.

Hex Address: F0F

Table 145. Timer 0–1 Control Registers (TxCTL)

Bit	7	6	5	4	3	2	1	0
Field	TEN	TPOL	PRES			TMODE		
RESET	0							
R/W	R/W							
Address	F07H, F0FH							

Hex Addresses: F10–F3F

This address range is reserved.

UART Control Registers

For more information about the UART registers, see the UART Control Register Definitions section on page 88.

Hex Address: F40

Table 146. UART Transmit Data Register (U0TXD)

Bit	7	6	5	4	3	2	1	0
Field	TXD							
RESET	X	X	X	X	X	X	X	X
R/W	W	W	W	W	W	W	W	W
Address	F40H							

Table 147. UART Receive Data Register (U0RXD)

Bit	7	6	5	4	3	2	1	0
Field	RXD							
RESET	X							
R/W	R							
Address	F40H							