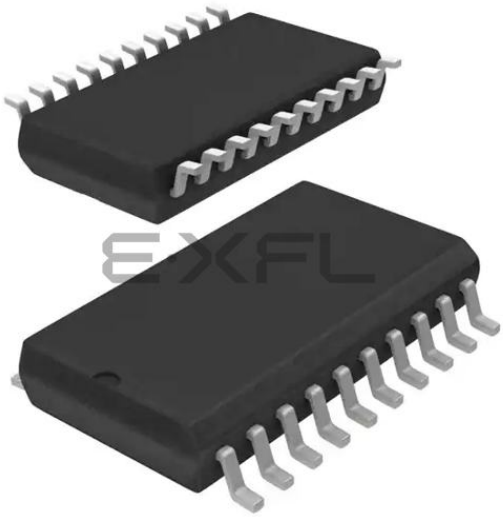

Welcome to E-XFL.COM

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, LINbus, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	17
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0880sh020sg

List of Tables

Chapter 6. Low-Power Modes

The Z8 Encore! XP F1680 Series products have power-saving features. The highest level of power reduction is provided by the STOP Mode. The next lower level of power reduction is provided by the HALT Mode.

Further power savings can be implemented by disabling individual peripheral blocks while in NORMAL Mode.

6.1. STOP Mode

Executing the eZ8 CPU's Stop instruction places the device into STOP Mode. In STOP Mode, the operating characteristics are:

- Primary crystal oscillator and internal precision oscillator are stopped; XIN and XOUT (if previously enabled) are disabled and PA0/PA1 reverts to the states programmed by the GPIO registers
- System clock is stopped
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- Watchdog Timer's internal RC oscillator continues operating if enabled by the Oscillator Control Register
- If enabled, the Watchdog Timer (WDT) logic continues operating
- If enabled, the 32kHz secondary oscillator continues operating
- If enabled for operation in STOP Mode, the Timer logic continues to operate with 32kHz secondary oscillator as the Timer clock source
- If enabled for operation in STOP Mode by the associated Flash option bit, the VBO protection circuit continues operating; the low-voltage detection circuit continues to operate if enabled by the Power Control Register
- Low-Power Operational Amplifier and comparator continue to operate if enabled by the Power Control Register
- All other on-chip peripherals are idle

- Writing 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 0 to the IRQE bit in the interrupt control register
- Reset
- Execution of a Trap instruction
- Illegal Instruction Trap
- Primary Oscillator Fail Trap
- Watchdog Oscillator Fail Trap

8.3.2. 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 the interrupts are enabled with identical interrupt priority (for example, all as Level 2 interrupts), the interrupt priority is assigned from highest to lowest as specified in Table 36 on page 69. Level 3 interrupts are always assigned higher priority than Level 2 interrupts which, in turn, always are assigned 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 36. Reset, Watchdog Timer interrupt (if enabled), Primary Oscillator Fail Trap, Watchdog Timer Oscillator Fail Trap and Illegal Instruction Trap always have highest (Level 3) priority.

8.3.3. 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.

8.4.3. Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register, shown in Table 39, 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 can read the Interrupt Request 2 register to determine if any interrupt requests are pending.

Table 39. Interrupt Request 2 Register (IRQ2)

Bits	7	6	5	4	3	2	1	0
Field	Reserved	MCTI	U1RXI	U1TXI	PC3I	PC2I	PC1I	PC0I
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FC6H							

Bit	Description
[7]	Reserved; must be 0.
[6] MCTI	**Multi-channel timer Interrupt Request** 0 = No interrupt request is pending for multi-channel timer. 1 = An interrupt request from multi-channel timer is awaiting service.
[5] U1RXI	**UART 1 Receiver Interrupt Request** 0 = No interrupt request is pending for the UART 1 receiver. 1 = An interrupt request from the UART 1 receiver is awaiting service.
[4] U1TXI	**UART 1 Transmitter Interrupt Request** 0 = No interrupt request is pending for the UART 1 transmitter. 1 = An interrupt request from the UART 1 transmitter is awaiting service.
[3:0] 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; *x* indicates the specific GPIO Port C pin number (0–3).

In CONTINUOUS Mode, the timer clock always provides the timer input. The timer period is calculated using the following equation:

$$\text{CONTINUOUS Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

If an initial starting value other than 0001H is loaded into the Timer High and Low Byte registers, the ONE-SHOT Mode equation must be used to determine the first time-out period.

9.2.3.4. COUNTER Mode

In COUNTER Mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO port pin Timer Input alternate function. The TPOL bit in the Timer Control 1 Register selects whether the count occurs on the rising edge or the falling edge of the Timer Input signal. In COUNTER Mode, the prescaler is disabled.

! Caution: The input frequency of the Timer Input signal must not exceed one-fourth the timer clock frequency.

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or High to Low) at timer reload.

Observe the following steps to configure a timer for COUNTER Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer.
   - Configure the timer for COUNTER Mode.
   - Select either the rising edge or falling edge of the Timer Input signal for the count. This also sets the initial logic level (High or Low) for the Timer Output Alternate Function. However, the Timer Output function is not required to be enabled.
2. Write to the Timer Control 2 Register to choose the timer clock source.

rate. To avoid an autobaud overrun error, the system clock must not be greater than 2^{19} times the baud rate (16 bit counter following 3-bit prescaler when counting the 8 bit times of the Autobaud sequence).

Following the Synch character, the LIN-UART hardware transits to the Active state, in which the identifier character is received and the characters of the response section of the message are sent or received. The slave remains in this Active state until a break is received or software forces a state change. After it is in an Active state (i.e., autobaud has completed), a break of 10 or more bit times is recognized and causes a transition to the Autobaud state.

If the identifier character indicates that this slave device is not participating in the message, software sets the LinState[1:0] = 01b (Wait for Break state) to ignore the rest of the message. No further receive interrupts will occur until the next break.

12.1.11. LIN-UART Interrupts

The LIN-UART features separate interrupts for the transmitter and receiver. In addition, when the LIN-UART primary functionality is disabled, the Baud Rate Generator can also function as a basic timer with interrupt capability.

12.1.11.1. Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register Empty bit (TDRE) is set to 1. This indicates that the transmitter is ready to accept new data for transmission. The TDRE interrupt occurs when the transmitter is initially enabled and after the Transmit Shift Register has shifted out the first bit of a character. At this point, the Transmit Data Register can be written with the next character to send. This provides 7 bit periods of latency to load the Transmit Data Register before the Transmit Shift Register completes shifting the current character. Writing to the LIN-UART Transmit Data Register clears the TDRE bit to 0.

12.1.11.2. Receiver Interrupts

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

- A data byte has been received and is available in the LIN-UART Receive Data Register. This interrupt can be disabled independent of the other receiver interrupt sources via the RDAIRQ bit (this feature is useful in devices which support DMA). The received data interrupt occurs after the receive character has been 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.

12.1.12. LIN-UART Baud Rate Generator

The LIN-UART Baud Rate Generator creates a lower frequency baud rate clock for data transmission. The input to the Baud Rate Generator is the system clock. The LIN-UART Baud Rate High and Low Byte registers combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data-transmission rate (baud rate) of the LIN-UART. The LIN-UART data rate for normal UART operation is calculated using the following equation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Baud Rate Divisor Value}}$$

The LIN-UART data rate for LIN mode UART operation is calculated using the following equation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{\text{UART Baud Rate Divisor Value}}$$

When the LIN-UART is disabled, the BRG functions as a basic 16-bit timer with interrupt on time-out. To configure the BRG as a timer with interrupt on time-out, follow the procedure below:

1. Disable the LIN-UART receiver by clearing the REN bit in the LIN-UART Control 0 Register to 0 (i.e., the TEN bit can be asserted; transmit activity can occur).
2. Load the appropriate 16-bit count value into the LIN-UART Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and the associated interrupt by setting the BRGCTL bit in the LIN-UART Control 1 Register to 1.

12.2. Noise Filter

A noise filter circuit is included which filters noise on a digital input signal (such as UART Receive Data) before the data is sampled by the block. This noise filter is likely to be a requirement for protocols with a noisy environment.

The noise filter contains the following features:

- Synchronizes the receive input data to the System Clock
- Noise Filter Enable (NFEN) input selects whether the noise filter is bypassed (NFEN = 0) or included (NFEN=1) in the receive data path

endec and passed to the UART. Communication is half-duplex, that is, simultaneous data transmission and reception is not allowed.

The baud rate is set by the UART's Baud Rate Generator and supports IrDA standard baud rates from 9600 baud to 115.2KBaud. Higher baud rates are possible, but do not meet IrDA specifications. The UART must be enabled to use the infrared endec. The infrared endec data rate is calculated using the following equation:

$$\text{Infrared Data Rate(bits/s)} = \frac{\text{System Clock Frequency(Hz)}}{16 \times \text{UART Baud Rate Divisor Value}}$$

13.2.1. Transmitting IrDA Data

The data to be transmitted using the infrared transceiver is first sent to the UART. The UART's transmit signal (TXD) and baud rate clock are used by the IrDA to generate the modulation signal (IR_TXD) that drives the infrared transceiver. Each UART/infrared data bit is 16 clocks wide. If the data to be transmitted is 1, the IR_TXD signal remains Low for the full 16-clock period. If the data to be transmitted is 0, the transmitter first outputs a 7-clock Low period, followed by a 3-clock High pulse. Finally, a 6-clock Low pulse is the output to complete the full 16 clock data period. Figure 28 displays IrDA data transmission. When the infrared endec is enabled, the UART's TXD signal is internal to the Z8 Encore! XP F1680 Series products while the IR_TXD signal is the output through the TXD pin.

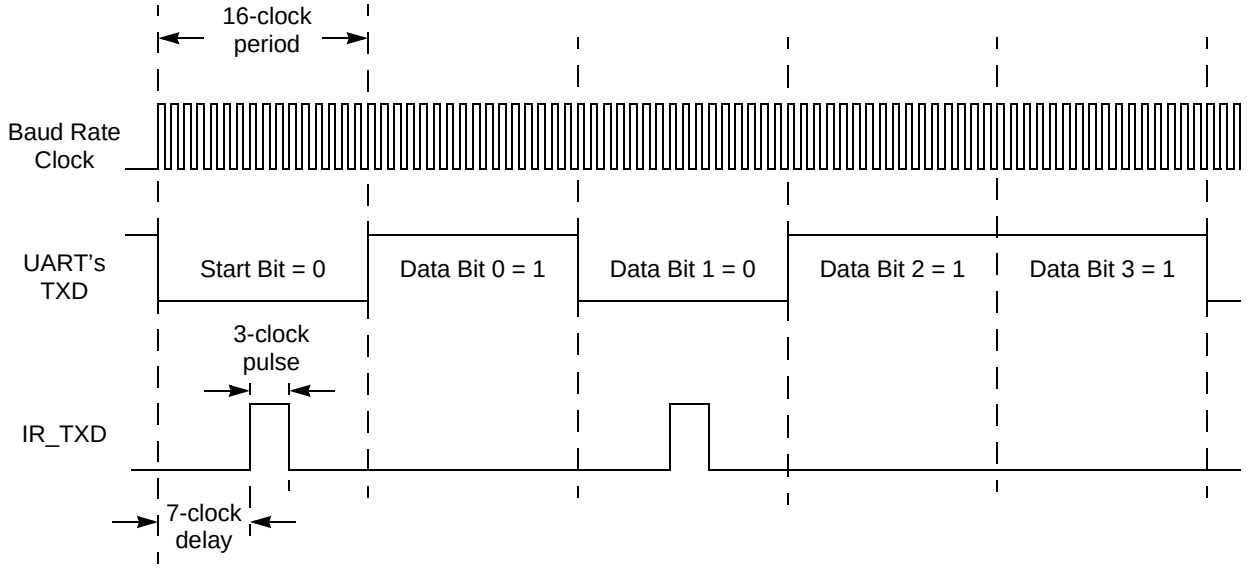

Figure 28. Infrared Data Transmission

13.2.2. Receiving IrDA Data

Data received from the infrared transceiver using the IR_RXD signal through the RXD pin is decoded by the infrared endec and passed to the UART. The UART's baud rate clock is used by the infrared endec to generate the demodulated signal (RXD) that drives the UART. Each UART/infrared data bit is 16 clocks wide. Figure 29 displays data reception. When the infrared endec is enabled, the UART's RXD signal is internal to the Z8 Encore! XP F1680 Series products while the IR_RXD signal is received through the RXD pin.

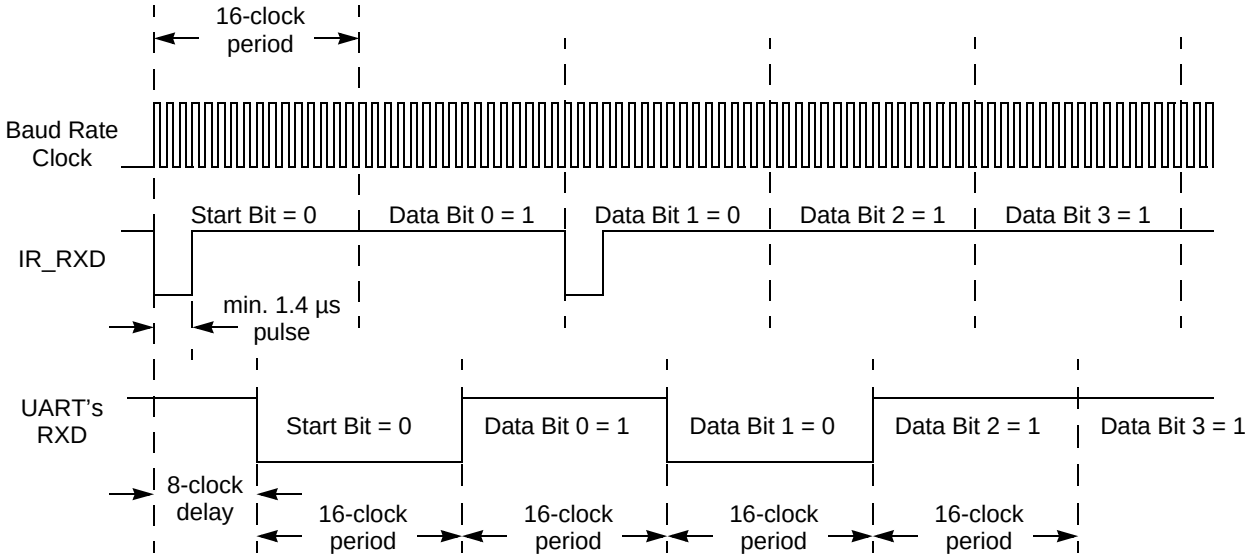

Figure 29. IrDA Data Reception

! Caution: The system clock frequency must be at least 1.0 MHz to ensure proper reception of the 1.4 µs minimum width pulses allowed by the IrDA standard.

13.2.2.1. Endec Receiver Synchronization

The IrDA receiver uses a local baud rate clock counter (0 to 15 clock periods) to generate an input stream for the UART and to create a sampling window for detection of incoming pulses. The generated UART input (UART RXD) is delayed by 8 baud rate clock periods with respect to the incoming IrDA data stream. When a falling edge in the input data stream is detected, the endec counter is reset. When the count reaches a value of 8, the UART RXD value is updated to reflect the value of the decoded data. When the count reaches 12 baud clock periods, the sampling window for the next incoming pulse opens.

Chapter 15. Low-Power Operational Amplifier

The low-power operational amplifier is a standard operational amplifier designed for current measurements. Each of the three ports of the amplifier is accessible from the package pins. The inverting input is commonly used to connect to the current source. The output node connects an external feedback network to the inverting input. The non-inverting output is required to apply a nonzero bias point. In a standard single-supply system, this bias point must be substantially above ground to measure positive input currents. The noninverting input can also be used for offset correction.

Note: This amplifier is a voltage gain operational amplifier. Its transimpedance nature is determined by the feedback network.

The low-power operational amplifier contains only one pin configuration; ANA0 is the output/feedback node, ANA1 is the inverting input and ANA2 is the noninverting input.

To use the low-power operational amplifier, it must be enabled in the Power Control Register Definitions, discussed on page 44. The default state of the low-power operational amplifier is OFF. To use the low-power operational amplifier, the TRAM bit must be cleared, turning it ON (see Power Control Register 0 on page 44). When making normal ADC (i.e., not transimpedance) measurements on ANA0, the TRAM bit must be OFF. Turning the TRAM bit ON interferes with normal ADC measurements. Finally, this bit enables the amplifier even in STOP Mode. If the amplifier is not required in STOP Mode, disable it. Failing to perform this results in STOP Mode currents greater than specified.

As with other ADC measurements, any pins used for analog purposes must be configured as in the GPIO registers (see the Port A–E Alternate Function Subregisters on page 61).

Standard transimpedance measurements are made on ANA0 as selected by the ANAIN[3:0] bits of the ADC Control Register 0, discussed on page 189. It is also possible to make single-ended measurements on ANA1 and ANA2 when the amplifier is enabled which is often useful for determining offset conditions.

Bit	Description (Continued)
[6,0] ESPIEN1, ESPIEN0	**ESPI Enable and Direction Control** 00 = The ESPI block is disabled. BRG can be used as a general-purpose timer by setting BRGCTL = 1. 01 = Receive Only Mode. Use this setting in SLAVE Mode if software application is receiving data but not sending. TDRE will not assert. Transmitted data will be all 1s. Not valid in MASTER Mode since Master must source data to drive the transfer. 10 = Transmit Only Mode Use this setting in MASTER or SLAVE Mode when the software application is sending data but not receiving. RDRNE will not assert. 11 = Transmit/Receive Mode Use this setting if the software application is both sending and receiving information. Both TDRE and RDRNE will be active.
[5] BRGCTL	**Baud Rate Generator Control** The function of this bit depends upon ESPIEN1,0. When ESPIEN1,0 = 00, this bit allows enabling the BRG to provide periodic interrupts. **If the ESPI is disabled** 0 = The Baud Rate Generator timer function is disabled. Reading the Baud Rate High and Low registers returns the BRG reload value. 1 = The Baud Rate Generator timer function and time-out interrupt is enabled. Reading the Baud Rate High and Low registers returns the BRG Counter value. **If the ESPI is enabled** 0 = Reading the Baud Rate High and Low registers returns the BRG reload value. If MMEN = 1, the BRG is enabled to generate SCK. If MMEN = 0, the BRG is disabled. 1 = Reading the Baud Rate High and Low registers returns the BRG Counter value. If MMEN = 1, the BRG is enabled to generate SCK. If MMEN = 0 the BRG is enabled to provide a Slave SCK time-out. See the SLAVE Mode Abort error description on page 211. **Caution:** If reading the counter one byte at a time while the BRG is counting keep in mind that the values will not be in sync. Zilog recommends reading the counter using (2-byte) word reads.
[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 ESPI Clock Phase and Polarity Control section on page 201.
[3] CLKPOL	**Clock Polarity** 0 = SCK idles Low (0). 1 = SCK idles High (1).
[2] WOR	**Wire-OR (Open-Drain) Mode Enabled** 0 = ESPI signal pins not configured for open-drain. 1 = All four ESPI signal pins (SCK, $\overline{SS}$, MISO and MOSI) configured for open-drain function. This setting is typically used for multi-Master and/or Multi-Slave configurations.
[1] MMEN	**ESPI MASTER Mode Enable** This bit controls the data I/O pin selection and SCK direction. 0 = Data out on MISO, data in on MOSI (used in SPI SLAVE Mode), SCK is an input. 1 = Data out on MOSI, data in on MISO (used in SPI MASTER Mode), SCK is an output.

16.4.5. ESPI Status Register

The ESPI Status Register, shown in Table 113, indicates the current state of the ESPI. All bits revert to their Reset state if the ESPI is disabled.

Table 113. ESPI Status Register (ESPISTAT)

Bits	7	6	5	4	3	2	1	0
Field	TDRE	TUND	COL	ABT	ROVR	RDRNE	TFST	SLAS
Reset	1	0	0	0	0	0	0	1
R/W	R	R/W*	R/W*	R/W*	R/W*	R	R	R
Address	F64H							

Note: R/W* = Read access. Write a 1 to clear the bit to 0.

Bit	Description
[7] TDRE	**Transmit Data Register Empty** 0 = Transmit Data Register is full or ESPI is disabled. 1 = Transmit Data Register is empty. A write to the ESPI (Transmit) Data Register clears this bit.
[6] TUND	**Transmit Underrun** 0 = A Transmit Underrun error has not occurred. 1 = A Transmit Underrun error has occurred.
[5] COL	**Collision** 0 = A multi-Master collision (mode fault) has not occurred. 1 = A multi-Master collision (mode fault) has occurred.
[4] ABT	**SLAVE Mode Transaction Abort** This bit is set if the ESPI 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 ESPIMODE register. This bit can also be set in SLAVE Mode by an SCK monitor time-out (MMEN = 0, BRGCTL = 1). 0 = A SLAVE Mode transaction abort has not occurred. 1 = A SLAVE Mode transaction abort has occurred.
[3] ROVR	**Receive Overrun** 0 = A Receive Overrun error has not occurred. 1 = A Receive Overrun error has occurred.
[2] RDRNE	**Receive Data Register Not Empty** 0 = Receive Data Register is empty. 1 = Receive Data Register is not empty.

17.3.2. I²C Interrupt Status Register

The read-only I²C Interrupt Status Register, shown in Table 120, indicates the cause of any current I²C interrupt and provides status of the I²C controller. When an interrupt occurs, one or more of the TDRE, RDRF, SAM, ARBLST, SPRS or NCKI bits is set. The GCA and RD bits do not generate an interrupt but rather provide status associated with the SAM bit interrupt.

Table 120. I²C Interrupt Status Register (I2CISTAT = F51H)

Bits	7	6	5	4	3	2	1	0
Field	TDRE	RDRF	SAM	GCA	RD	ARBLST	SPRS	NCKI
Reset	1	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	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 set, this bit causes the I²C controller to generate an interrupt, 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 clears 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 clears by reading the I2CDATA Register.
[5] SAM	**Slave Address Match** This bit is set = 1 if the I²C controller is enabled in SLAVE Mode and an address is received that matches the unique slave address or General Call Address (if enabled by the GCE bit in the I²C Mode Register). In 10-bit addressing mode, this bit is not set until a match is achieved on both address bytes. When this bit is set, the RD and GCA bits are also valid. This bit clears by reading the I2CISTAT Register.
[4] GCA	**General Call Address** This bit is set in SLAVE Mode when the General Call Address or Start byte is recognized (in either 7 or 10 bit SLAVE Mode). The GCE bit in the I²C Mode Register must be set to enable recognition of the General Call Address and Start byte. This bit clears when IEN = 0 and is updated following the first address byte of each SLAVE Mode transaction. A General Call Address is distinguished from a Start byte by the value of the RD bit (RD = 0 for General Call Address, 1 for Start byte).
[3] RD	**Read** This bit indicates the direction of transfer of the data. It is set when the Master is reading data from the Slave. This bit matches the least-significant bit of the address byte after the start condition occurs (for both MASTER and SLAVE modes). This bit clears when IEN = 0 and is updated following the first address byte of each transaction.

Chapter 18. Comparator

The Z8 Encore! XP F1680 Series devices feature two same general purpose comparators that compares two analog input signals. For each comparator, a GPIO (C0INP/C1INP) pin provides the positive comparator input, the negative input (C0INN/C1INN) can be taken from either an external GPIO pin or an internal reference. The output of each comparator is available as an interrupt source or can be routed to an external pin using the GPIO multiplex. Features for each comparator include:

- Two inputs which are connected using the GPIO multiplex (MUX)
- One input can be connected to a programmable internal reference
- One input can be connected to the on-chip temperature sensor
- Output can trigger timer counting
- Output can be either an interrupt source or an output to an external pin
- Operation in STOP Mode

18.1. Operation

One of the comparator inputs can be connected to an internal reference which is a user-selectable reference that is user-programmable with 200mV resolution.

The comparator can be powered down to save supply current or to continue to operate in STOP Mode. For details, see the Power Control Register 0 section on page 44. In STOP Mode, the comparator interrupt (if enabled) automatically initiates a Stop Mode Recovery and generates an interrupt request. In the Reset Status Register (see page 40), the stop bit is set to 1. Also, the Comparator request bit in the Interrupt Request 1 Register (see page 74) is set. Following completion of the Stop Mode Recovery, and if interrupts are enabled, the CPU responds to the interrupt request by fetching the comparator interrupt vector.

! Caution: Because of the propagation delay of the comparator, spurious interrupts can result after enabling the comparator. Zilog recommends not enabling the comparator without first disabling interrupts, then waiting for the comparator output to settle.

The following code example shows how to safely enable the comparator:

```
di
ldx CMP0,r0
nop
```

```
DBG ← Size[7:0]
DBG → 1-65536 data bytes
```

Read Program Memory CRC (0EH). The Read Program Memory CRC command computes and returns the CRC (cyclic redundancy check) of Program memory using the 16-bit CRC-CCITT polynomial ($x^{16} + x^{12} + x^5 + 1$). The CRC is preset to all 1s. The least-significant bit of the data is shifted through the polynomial first. The CRC is inverted when it is transmitted. If the device is not in DEBUG mode, this command returns FFFFH for the CRC value. Unlike most other OCD Read commands, there is a delay from issuing of the command until the OCD returns the data. The OCD reads Program memory, calculates the CRC value and returns the result. The delay is a function of the Program memory size and is approximately equal to the system clock period multiplied by the number of bytes in Program memory.

```
DBG ← 0EH
DBG → CRC[15:8]
DBG → CRC[7:0]
```

Step Instruction (10H). The Step Instruction command steps one assembly instruction at the current Program Counter (PC) location. If the device is not in DEBUG mode or the Read Protect Option bit is enabled, the OCD ignores this command.

```
DBG ← 10H
```

Stuff Instruction (11H). The Stuff Instruction command steps one assembly instruction and allows specification of the first byte of the instruction. The remaining 0-4 bytes of the instruction are read from Program memory. This command is useful for stepping over instructions where the first byte of the instruction has been overwritten by a breakpoint. If the device is not in DEBUG mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 11H
DBG ← opcode[7:0]
```

Execute Instruction (12H). The Execute Instruction command allows sending an entire instruction to be executed to the eZ8 CPU. This command can also step over breakpoints. The number of bytes to send for the instruction depends on the op code. If the device is not in DEBUG mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 12H
DBG ← 1-5 byte opcode
```

Write Line Control Register (18H). The Write Line Control Register command writes the data that follows to the Line Control Register.

```
DBG ← 18H
DBG ← LCR[7:0]
```

Read Line Control Register (19H). The Read Line Control Register command returns the current value in the Line Control Register.

```
DBG ← 19H
DBG → LCR[7:0]
```

Tables 178 through 185 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 instructions are to 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 178. 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
SBC	dst, src	Subtract with Carry
SBCX	dst, src	Subtract with Carry using Extended Addressing
SUB	dst, src	Subtract
SUBX	dst, src	Subtract using Extended Addressing

Table 186. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
CLR dst	dst ← 00H	R		B0	–	–	–	–	–	–	2	2
		IR		B1							2	3
COM dst	dst ← ~dst	R		60	–	*	*	0	–	–	2	2
		IR		61							2	3
CP dst, src	dst – src	r	r	A2	*	*	*	*	–	–	2	3
		r	Ir	A3							2	4
		R	R	A4							3	3
		R	IR	A5							3	4
		R	IM	A6							3	3
		IR	IM	A7							3	4
CPC dst, src	dst – src – C	r	r	1F A2	*	*	*	*	–	–	3	3
		r	Ir	1F A3							3	4
		R	R	1F A4							4	3
		R	IR	1F A5							4	4
		R	IM	1F A6							4	3
		IR	IM	1F A7							4	4
CPCX dst, src	dst – src – C	ER	ER	1F A8	*	*	*	*	–	–	5	3
		ER	IM	1F A9							5	3
CPX dst, src	dst – src	ER	ER	A8	*	*	*	*	–	–	4	3
		ER	IM	A9							4	3
DA dst	dst ← DA(dst)	R		40	*	*	*	X	–	–	2	2
		IR		41							2	3
DEC dst	dst ← dst – 1	R		30	–	*	*	*	–	–	2	2
		IR		31							2	3
DECW dst	dst ← dst – 1	RR		80	–	*	*	*	–	–	2	5
		IRR		81							2	6
DI	IRQCTL[7] ← 0			8F	–	–	–	–	–	–	1	2

Flags notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 186. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
LD dst, rc	dst ← src	r	IM	0C-FC	–	–	–	–	–	–	2	2
		r	X(r)	C7							3	3
		X(r)	r	D7							3	4
		r	Ir	E3							2	3
		R	R	E4							3	2
		R	IR	E5							3	4
		R	IM	E6							3	2
		IR	IM	E7							3	3
		Ir	r	F3							2	3
		IR	R	F5							3	3
LDC dst, src	dst ← src	r	Irr	C2	–	–	–	–	–	–	2	5
		Ir	Irr	C5							2	9
		Irr	r	D2							2	5
LDCI dst, src	dst ← src r ← r + 1 rr ← rr + 1	Ir	Irr	C3	–	–	–	–	–	–	2	9
		Irr	Ir	D3							2	9
LDE dst, src	dst ← src	r	Irr	82	–	–	–	–	–	–	2	5
		Irr	r	92							2	5
LDEI dst, src	dst ← src r ← r + 1 rr ← rr + 1	Ir	Irr	83	–	–	–	–	–	–	2	9
		Irr	Ir	93							2	9
LDWX dst, src	dst ← src	ER	ER	1FE8	–	–	–	–	–	–	5	4

Flags notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.