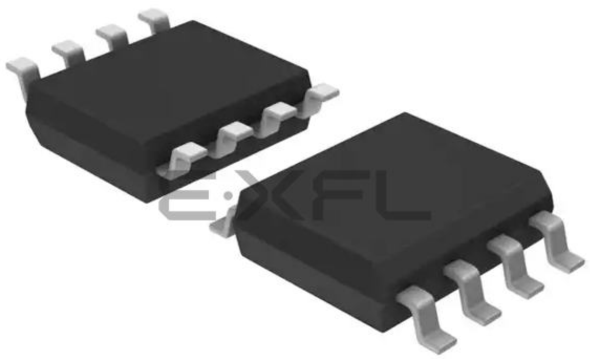

What is "Embedded - Microcontrollers"?

"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	5MHz
Connectivity	IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, LED, POR, PWM, WDT
Number of I/O	6
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 4x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	8-SOIC (0.154", 3.90mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0823sb005sg

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Z8, Z8 Encore! and Z8 Encore! XP are trademarks or registered trademarks of Zilog, Inc. All other product or service names are the property of their respective owners.

CPU and Peripheral Overview

The eZ8 CPU, Zilog's latest 8-bit central processing unit (CPU), meets the continuing demand for faster and code-efficient microcontrollers. The eZ8 CPU executes a superset of the original Z8 instruction set. The eZ8 CPU features include:

- Direct register-to-register architecture allows each register to function as an accumulator, improving execution time and decreasing the required program memory
- Software stack allows much greater depth in subroutine calls and interrupts than hardware stacks
- Compatible with existing Z8 code
- Expanded internal Register File allows access of up to 4 KB
- New instructions improve execution efficiency for code developed using higher-level programming languages, including C
- Pipelined instruction fetch and execution
- New instructions for improved performance including BIT, BSWAP, BTJ, CPC, LDC, LDCI, LEA, MULT, and SRL
- New instructions support 12-bit linear addressing of the Register File
- Up to 10 MIPS operation
- C-Compiler friendly
- 2 to 9 clock cycles per instruction

For more information about the eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128) available for download at www.zilog.com.

General-Purpose I/O

F0823 Series features 6 to 24 port pins (Ports A–C) for general-purpose I/O (GPIO). The number of GPIO pins available is a function of package. Each pin is individually programmable. 5V-tolerant input pins are available on all I/Os on 8-pin devices, most I/Os on other package types.

Flash Controller

The Flash Controller programs and erases Flash memory. The Flash Controller supports protection against accidental program and erasure, as well as factory serialization and read protection.

Table 17. Port Alternate Function Mapping (Non 8-Pin Parts) (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port B[3]	PB03	Reserved		AFS1[0]: 0
		ANA0	ADC Analog Input	AFS1[0]: 1
	PB1	Reserved		AFS1[1]: 0
		ANA1	ADC Analog Input	AFS1[1]: 1
	PB2	Reserved		AFS1[2]: 0
		ANA2	ADC Analog Input	AFS1[2]: 1
	PB3	CLKIN	External Clock Input	AFS1[3]: 0
		ANA3	ADC Analog Input	AFS1[3]: 1
	PB4	Reserved		AFS1[4]: 0
		ANA7	ADC Analog Input	AFS1[4]: 1
	PB5	Reserved		AFS1[5]: 0
		V_{REF}[4]	ADC Voltage Reference	AFS1[5]: 1
	PB6	Reserved		AFS1[6]: 0
		Reserved		AFS1[6]: 1
	PB7	Reserved		AFS1[7]: 0
		Reserved		AFS1[7]: 1

Notes:

1. Because there is only a single alternate function for each Port A pin, the Alternate Function Set registers are not implemented for Port A. Enabling alternate function selections as described in the Port A–C Alternate Function Subregisters section on page 43 automatically enables the associated alternate function.
2. Whether PA0/PA6 take on the timer input or timer output complement function depends on the timer configuration as described in the Timer Pin Signal Operation section on page 83.
3. Because there are at most two choices of alternate function for any pin of Port B, the Alternate Function Set register AFS2 is implemented but not used to select the function. Also, alternate function selection as described in the Port A–C Alternate Function Subregisters section on page 43 must also be enabled.
4. V_{REF} is available on PB5 in 28-pin products only.
5. Because there are at most two choices of alternate function for any pin of Port C, the Alternate Function Set register AFS2 is implemented but not used to select the function. Also, Alternate Function selection as described in the Port A–C Alternate Function Subregisters section on page 43 must also be enabled.
6. V_{REF} is available on PC2 in 20-pin parts only.

Port A–C Pull-up Enable Subregisters

The Port A–C Pull-up Enable Subregister (Table 27) is accessed through the Port A–C Control Register by writing `06H` to the Port A–C Address Register. Setting the bits in the Port A–C Pull-up Enable subregisters enables a weak internal resistive pull-up on the specified Port pins.

Table 27. Port A–C Pull-Up Enable Subregisters (PPUEx)

Bit	7	6	5	4	3	2	1	0
Field	PPUE7	PPUE6	PPUE5	PPUE4	PPUE3	PPUE2	PPUE1	PPUE0
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	If 06H in Port A–C Address Register, accessible through the Port A–C Control Register							

Bit	Description
[7:0] PPUEx	**Port Pull-up Enabled** 0 = The weak pull-up on the Port pin is disabled. 1 = The weak pull-up on the Port pin is enabled.

Note: x indicates the specific GPIO port pin number (7–0).

Bit	Description (Continued)
[5] T0ENL	**Timer 0 Interrupt Request Enable Low Bit**
[4] U0RENL	**UART 0 Receive Interrupt Request Enable Low Bit**
[3] U0TENL	**UART 0 Transmit Interrupt Request Enable Low Bit**
[2:1]	**Reserved** These bits are reserved and must be programmed to 00.
[0] ADCENL	**ADC Interrupt Request Enable Low Bit**

IRQ1 Enable High and Low Bit Registers

Table 42 describes the priority control for IRQ1. The IRQ1 Enable High and Low Bit registers (Table 43 and Table 44) form a priority-encoded enabling for interrupts in the Interrupt Request 1 Register. Priority is generated by setting bits in each register.

Table 42. IRQ1 Enable and Priority Encoding

IRQ1ENH[*x*]	IRQ1ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High
Note: *x* indicates register bits 0–7.			

Interrupt Control Register

The Interrupt Control (IRQCTL) Register (Table 50) contains the master enable bit for all interrupts.

Table 50. Interrupt Control Register (IRQCTL)

Bit	7	6	5	4	3	2	1	0
Field	IRQE	Reserved						
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R	R	R	R	R	R	R
Address	FCFH							

Bit	Description
[7] IRQE	**Interrupt Request Enable** This bit is set to 1 by executing an Enable Interrupts (EI) or Interrupt Return (IRET) instruction, or by a direct register write of a 1 to this bit. It is reset to 0 by executing a DI instruction, eZ8 CPU acknowledgement of an interrupt request, reset or by a direct register write of a 0 to this bit. 0 = Interrupts are disabled. 1 = Interrupts are enabled.
[6:0]	**Reserved** These bits are reserved and must be programmed to 0000000 when read.

enabled, the Timer Output pin changes state (from Low to High or from High to Low) at timer reload.

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

1. Write to the Timer Control 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 selection 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 High and Low Byte registers to set the starting count value. This only affects the first pass in COUNTER Mode. After the first timer reload in COUNTER Mode, counting always begins at the reset value of `0001H`. In COUNTER Mode the Timer High and Low Byte registers must be written with the value `0001H`.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control Register to enable the timer.

In COUNTER Mode, the number of timer input transitions since the timer start is computed via the following equation:

$$\text{COUNTER Mode Timer Input Transitions} = \text{Current Count Value} - \text{Start Value}$$

COMPARATOR COUNTER Mode

In COMPARATOR COUNTER Mode, the timer counts input transitions from the analog comparator output. The `TPOL` bit in the Timer Control Register selects whether the count occurs on the rising edge or the falling edge of the comparator output signal. In COMPARATOR COUNTER Mode, the prescaler is disabled.

Bit	Description (Continued)
[6] TPOL (cont'd.)	**COUNTER Mode** If the timer is enabled the Timer Output signal is complemented after timer reload. 0 = Count occurs on the rising edge of the Timer Input signal. 1 = Count occurs on the falling edge of the Timer Input signal. **PWM SINGLE OUTPUT Mode** 0 = Timer Output is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon reload. 1 = Timer Output is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon reload. **CAPTURE Mode** 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **COMPARE Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. **GATED Mode** 0 = Timer counts when the Timer Input signal is High (1) and interrupts are generated on the falling edge of the Timer Input. 1 = Timer counts when the Timer Input signal is Low (0) and interrupts are generated on the rising edge of the Timer Input. **CAPTURE/COMPARE Mode** 0 = Counting is started on the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal. 1 = Counting is started on the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal. **PWM DUAL OUTPUT Mode** 0 = Timer Output is forced Low (0) and Timer Output Complement is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon reload. When enabled, the Timer Output Complement is forced Low (0) upon PWM count match and forced High (1) upon reload. The PWMD field in TxCTL0 register is a programmable delay to control the number of cycles time delay before the Timer Output and the Timer Output Complement is forced to High (1). 1 = Timer Output is forced High (1) and Timer Output Complement is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon reload.When enabled, the Timer Output Complement is forced High (1) upon PWM count match and forced Low (0) upon reload. The PWMD field in TxCTL0 register is a programmable delay to control the number of cycles time delay before the Timer Output and the Timer Output Complement is forced to Low (0). **CAPTURE RESTART Mode** 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal.

Universal Asynchronous Receiver/ Transmitter

The universal asynchronous receiver/transmitter (UART) is a full-duplex communication channel capable of handling asynchronous data transfers. The UART uses a single 8-bit data mode with selectable parity. The features of UART include:

- 8-bit asynchronous data transfer
- Selectable even- and odd-parity generation and checking
- Option of one or two STOP bits
- Separate transmit and receive interrupts
- Framing, parity, overrun, and break detection
- Separate transmit and receive enables
- 16-bit baud rate generator (BRG)
- Selectable MULTIPROCESSOR (9-bit) Mode with three configurable interrupt schemes
- BRG can be configured and used as a basic 16-bit timer
- Driver Enable output for external bus transceivers

Architecture

The UART consists of three primary functional blocks: transmitter, receiver, and baud rate generator. The UART’s transmitter and receiver function independently, but employ the same baud rate and data format. Figure 10 displays the UART architecture.

   - Set or clear `CTSE` to enable or disable control from the remote receiver using the CTS pin.
8. Execute an `EI` instruction to enable interrupts.

The UART is now configured for interrupt-driven data transmission. Because the UART Transmit Data Register is empty, an interrupt is generated immediately. When the UART Transmit interrupt is detected, the associated interrupt service routine (ISR) performs the following:

1. Write the UART Control 1 Register to select the multiprocessor bit for the byte to be transmitted:

   Set the Multiprocessor Bit Transmitter (`MPBT`) if sending an address byte, clear it if sending a data byte.

2. Write the data byte to the UART Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift register and transmits the data.
3. Clear the UART Transmit interrupt bit in the applicable Interrupt Request register.
4. Execute the `IRET` instruction to return from the interrupt-service routine and wait for the Transmit Data Register to again become empty.

Receiving Data Using the Polled Method

Observe the following steps to configure the UART for polled data reception:

1. Write to the UART Baud Rate High and Low Byte registers to set an acceptable baud rate for the incoming data stream.
2. Enable the UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Write to the UART Control 1 Register to enable MULTIPROCESSOR Mode functions, if appropriate.
4. Write to the UART Control 0 Register to:
   - Set the receive enable bit (`REN`) to enable the UART for data reception
   - Enable parity, if appropriate and if Multiprocessor mode is not enabled, and select either even or odd parity
5. Check the RDA bit in the UART Status 0 Register to determine if the Receive Data Register contains a valid data byte (indicated by a 1). If `RDA` is set to 1 to indicate available data, continue to Step 6. If the Receive Data Register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.
6. Read 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].

$$\left(\frac{1}{\text{Baud Rate (Hz)}}\right) \le \text{DE to Start Bit Setup Time (s)} \le \left(\frac{2}{\text{Baud Rate (Hz)}}\right)$$

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 after the Transmit shift register has shifted the first bit of data out. The Transmit Data Register can now be written with the next character to send. This action 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 UART Transmit Data Register clears the TDRE bit to 0.

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 independently of the other receiver interrupt sources. The received data interrupt occurs after the receive character has been received and placed in the Receive Data Register. To avoid an overrun error, software must respond to this received data available condition before the next character is completely received.

Note: 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 after the valid data has been 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

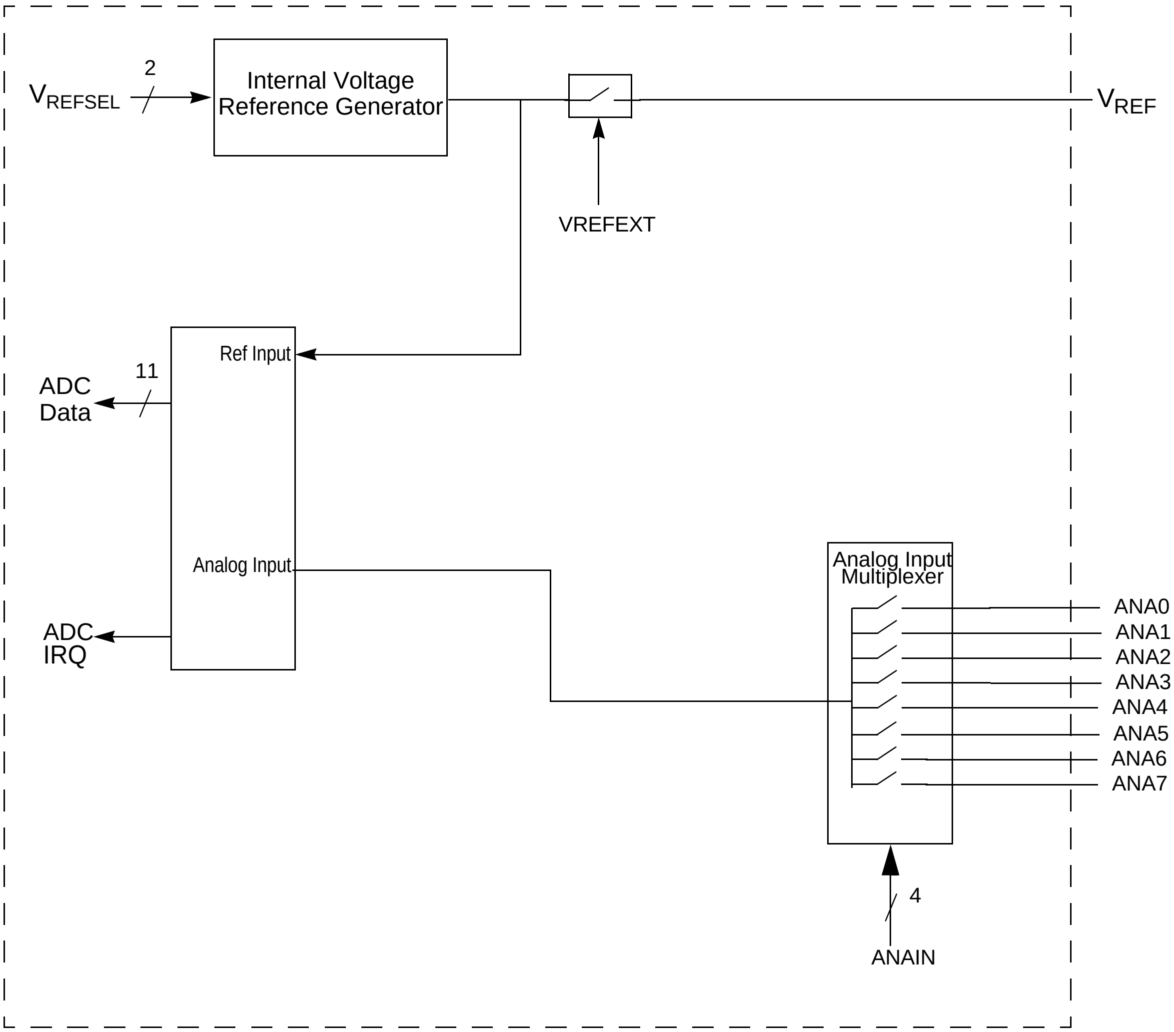

Figure 19. Analog-to-Digital Converter Block Diagram

Operation

The output of the ADC is an 11-bit, signed, two's-complement digital value. The output generally ranges from 0 to +1023, but offset errors can cause small negative values.

The ADC registers return 13 bits of data, but the two LSBs are intended for compensation use only. When the compensation routine is performed on the 13 bit raw ADC value, two bits of resolution are lost because of a rounding error. As a result, the final value is an 11-bit number.

   - `CEN` resets to 0 to indicate the conversion is complete

6. If the ADC remains idle for 160 consecutive system clock cycles, it is automatically powered-down.

Continuous Conversion

When configured for continuous conversion, the ADC continuously performs an analog-to-digital conversion on the selected analog input. Each new data value over-writes the previous value stored in the ADC Data registers. An interrupt is generated after each conversion.

! Caution: In CONTINUOUS Mode, ADC updates are limited by the input signal bandwidth of the ADC and the latency of the ADC and its digital filter. Step changes at the input are not detected at the next output from the ADC. The response of the ADC (in all modes) is limited by the input signal bandwidth and the latency.

Observe the following steps for setting up the ADC and initiating continuous conversion:

1. Enable the acceptable analog input by configuring the general-purpose I/O pins for alternate function. This action disables the digital input and output driver.
2. Write the ADC Control/Status Register 1 to configure the ADC:
   - Write the `REFSELH` bit of the pair {`REFSELH`, `REFSELL`} to select the internal voltage reference level or to disable the internal reference. The REFSELH bit is contained in the ADC Control/Status Register 1.

3. Write to the ADC Control Register 0 to configure the ADC for continuous conversion. The bit fields in the ADC Control Register can be written simultaneously:
   - Write to the ANAIN[3:0] field to select from the available analog input sources (different input pins available depending on the device).
   - Set `CONT` to 1 to select continuous conversion.
   - If the internal VREF must be output to a pin, set the REFEXT bit to 1. The internal voltage reference must be enabled in this case.
   - Write the REFSELL bit of the pair {REFSELH, REFSELL} to select the internal voltage reference level or to disable the internal reference. The REFSELL bit is contained in ADC Control Register 0.
   - Set `CEN` to 1 to start the conversions.

4. When the first conversion in continuous operation is complete (after 5129 system clock cycles, plus the 40 cycles for power-up, if necessary), the ADC control logic performs the following operations:
   - `CEN` resets to 0 to indicate the first conversion is complete. `CEN` remains 0 for all subsequent conversions in continuous operation
   - An interrupt request is sent to the Interrupt Controller to indicate the conversion is complete

5. The ADC writes a new data result every 256 system clock cycles. For each completed conversion, the ADC control logic performs the following operations:
   - Writes the 11-bit two's complement result to {ADCD_H[7:0], ADCD_L[7:5]}
   - An interrupt request to the Interrupt Controller denoting conversion complete

6. To disable continuous conversion, clear the `CONT` bit in the ADC Control Register to 0.

Interrupts

The ADC is able to interrupt the CPU whenever a conversion has been completed and the ADC is enabled.

When the ADC is disabled, an interrupt is not asserted; however, an interrupt pending when the ADC is disabled is not cleared.

Calibration and Compensation

Z8 Encore! XP F0823 Series ADC can be factory calibrated for offset error and gain error, with the compensation data stored in Flash memory. Alternatively, user code can perform its own calibration, storing the values into Flash themselves.

Factory Calibration

Devices that have been factory calibrated contain nine bytes of calibration data in the Flash option bit space. This data consists of three bytes for each reference type. For a list of input modes for which calibration data exists, see the Zilog Calibration Data section on page 152. There is 1 byte for offset, and there are 2 bytes for gain correction.

User Calibration

If you have precision references available, its own external calibration can be performed, storing the values into Flash themselves.

In the following list of OCD Commands, data and commands sent from the host to the OCD are identified by `'DBG ← Command/Data'`. Data sent from the OCD back to the host is identified by `'DBG → Data'`.

Read OCD Revision (00H). The Read OCD Revision command determines the version of the OCD. If OCD commands are added, removed, or changed, this revision number changes.

```
DBG ← 00H
DBG → OCDRev[15:8] (Major revision number)
DBG → OCDRev[7:0] (Minor revision number)
```

Read OCD Status Register (02H). The Read OCD Status Register command reads the OCDSTAT Register.

```
DBG ← 02H
DBG → OCDSTAT[7:0]
```

Read Runtime Counter (03H). The Runtime Counter counts system clock cycles in between breakpoints. The 16-bit Runtime Counter counts up from `0000H` and stops at the maximum count of `FFFFH`. The Runtime Counter is overwritten during the Write Memory, Read Memory, Write Register, Read Register, Read Memory CRC, Step Instruction, Stuff Instruction, and Execute Instruction commands.

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

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

```
DBG ← 04H
DBG ← OCDCTL[7:0]
```

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

```
DBG ← 05H
DBG → OCDCTL[7:0]
```

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

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

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

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

Write Register (08H). The Write Register command writes data to the Register File. Data can be written 1–256 bytes at a time (256 bytes can be written by setting size to 0). If the device is not in DEBUG Mode, the address and data values are discarded. If the Flash Read Protect Option bit is enabled, only writes to the Flash Control Registers are allowed and all other register write data values are discarded.

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

Read Register (09H). The Read Register command reads data from the Register File. Data can be read 1–256 bytes at a time (256 bytes can be read by setting size to 0). If the device is not in DEBUG Mode or if the Flash Read Protect Option bit is enabled, this command returns `FFH` for all the data values.

```
DBG ← 09H
DBG ← {4'h0,Register Address[11:8]
DBG ← Register Address[7:0]
DBG ← Size[7:0]
DBG → 1–256 data bytes
```

Write Program Memory (0AH). The Write Program Memory command writes data to Program Memory. This command is equivalent to the `LDC` and `LDCI` instructions. Data can be written 1–65536 bytes at a time (65536 bytes can be written by setting size to 0). The on-chip Flash Controller must be written to and unlocked for the programming operation to occur. If the Flash Controller is not unlocked, the data is discarded. If the device is not in DEBUG Mode or if the Flash Read Protect Option bit is enabled, the data is discarded.

```
DBG ← 0AH
DBG ← Program Memory Address[15:8]
DBG ← Program Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG ← 1–65536 data bytes
```

Read Program Memory (0BH). The Read Program Memory command reads data from Program Memory. This command is equivalent to the `LDC` and `LDCI` instructions. Data can be read 1–65536 bytes at a time (65536 bytes can be read by setting size to 0). If the device is not in DEBUG Mode or if the Flash Read Protect Option Bit is enabled, this command returns `FFH` for the data.

Oscillator Control

Z8 Encore! XP F0823 Series devices uses three possible clocking schemes, each user-selectable. These three schemes are:

- On-chip precision trimmed RC oscillator
- External clock drive
- On-chip low power Watchdog Timer oscillator

In addition, F0823 Series devices contain clock failure detection and recovery circuitry, which allow continued operation despite a failure of the primary oscillator.

Operation

This chapter discusses the logic used to select the system clock and handle primary oscillator failures. A description of the specific operation of each oscillator is outlined elsewhere in this document.

System Clock Selection

The oscillator control block selects from the available clocks. Table 104 details each clock source and its usage.

Table 104. Oscillator Configuration and Selection

Clock Source	Characteristics	Required Setup
Internal Precision RC Oscillator	• 32.8kHz or 5.53MHz • ± 4% accuracy when trimmed • No external components required	• Unlock and write Oscillator Control Register (OSCCTL) to enable and select oscillator at either 5.53MHz or 32.8kHz
External Clock Drive	• 0 to 20MHz • Accuracy dependent on external clock source	• Write GPIO registers to configure PB3 pin for external clock function • Unlock and write OSCCTL to select external system clock • Apply external clock signal to GPIO
Internal Watchdog Timer Oscillator	• 10kHz nominal • ± 40% accuracy; no external components required • Very Low power consumption	• Enable WDT if not enabled and wait until WDT Oscillator is operating. • Unlock and write Oscillator Control Register (OSCCTL) to enable and select oscillator

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7	3 ,															
8																
9																
A			3.3 **CPC** r1,r2	3.4 **CPC** r1,Ir2	4.3 **CPC** R2,R1	4.4 **CPC** IR2,R1	4.3 **CPC** R1,IM	4.4 **CPC** IR1,IM	5.3 **CPCX** ER2,ER1	5.3 **CPCX** IM,ER1						
B																
C	3.2 **SRL** R1	3.3 **SRL** IR1														
D																
E									5, 4 **LDWX** ER2,ER1							
F																

Figure 28. Second Opcode Map after 1FH

P

R