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	5MHz
Connectivity	IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, LED, POR, PWM, WDT
Number of I/O	16
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SSOP (0.209", 5.30mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0423hh005sc

General-Purpose Input/Output

Z8 Encore! XP® F0823 Series products support a maximum of 24 port pins (Ports A–C) for general-purpose input/output (GPIO) operations. Each port contains control and data registers. The GPIO control registers determine data direction, open-drain, output drive current, programmable pull-ups, Stop Mode Recovery functionality, and alternate pin functions. Each port pin is individually programmable. In addition, the Port C pins are capable of direct LED drive at programmable drive strengths.

GPIO Port Availability By Device

Table 14 lists the port pins available with each device and package type.

Table 14. Port Availability by Device and Package Type

Devices	Package	10-Bit ADC	Port A	Port B	Port C	Total I/O
Z8F0823SB, Z8F0823PB Z8F0423SB, Z8F0423PB Z8F0223SB, Z8F0223PB Z8F0123SB, Z8F0123PB	8-pin	Yes	[5:0]	No	No	6
Z8F0813SB, Z8F0813PB Z8F0413SB, Z8F0413PB Z8F0213SB, Z8F0213PB Z8F0113SB, Z8F011vPB	8-pin	No	[5:0]	No	No	6
Z8F0823PH, Z8F0823HH Z8F0423PH, Z8F0423HH Z8F0223PH, Z8F0223HH Z8F0123PH, Z8F0123HH	20-pin	Yes	[7:0]	[3:0]	[3:0]	16
Z8F0813PH, Z8F0813HH Z8F0413PH, Z8F0413HH Z8F0213PH, Z8F0213HH Z8F0113PH, Z8F0113HH	20-pin	No	[7:0]	[3:0]	[3:0]	16
Z8F0823PJ, Z8F0823SJ Z8F0423PJ, Z8F0423SJ Z8F0223PJ, Z8F0223SJ Z8F0123PJ, Z8F0123SJ	28-pin	Yes	[7:0]	[5:0]	[7:0]	22
Z8F0813PJ, Z8F0813SJ Z8F0413PJ, Z8F0413SJ Z8F0213PJ, Z8F0213SJ Z8F0113PJ, Z8F0113SJ	28-pin	No	[7:0]	[7:0]	[7:0]	24

PSMRE[7:0]—Port Stop Mode Recovery Source Enabled.
0 = The Port pin is not configured as a Stop Mode Recovery source. Transitions on this pin during STOP mode do not initiate Stop Mode Recovery.
1 = The Port pin is configured as a Stop Mode Recovery source. Any logic transition on this pin during STOP mode initiates Stop Mode Recovery.

Port A–C Pull-up Enable Sub-Registers

The Port A–C Pull-up Enable sub-register (Table 25) 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 sub-registers enables a weak internal resistive pull-up on the specified Port pins.

Table 25. Port A–C Pull-Up Enable Sub-Registers (PxPUE)

BITS	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
ADDR	If 06H in Port A–C Address Register, accessible through the Port A–C Control Register							

PPUE[7:0]—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.

Port A–C Alternate Function Set 1 Sub-Registers

The Port A–C Alternate Function Set1 sub-register (Table 26) is accessed through the Port A–C Control register by writing `07H` to the Port A–C Address register. The Alternate Function Set 1 sub-registers selects the alternate function available at a port pin. Alternate Functions selected by setting or clearing bits of this register are defined in GPIO Alternate Functions on page 36.

Note: *Alternate function selection on port pins must also be enabled as described in* Port A–C Alternate Function Sub-Registers *on page 45*.

! Caution: *To avoid missing interrupts, use the following coding style to clear bits in the Interrupt Request 0 register:*

Good coding style that avoids lost interrupt requests:
ANDX IRQ0, MASK

Software Interrupt Assertion

Program code generates interrupts directly. Writing a 1 to the correct bit in the Interrupt Request register triggers an interrupt (assuming that interrupt is enabled). When the interrupt request is acknowledged by the eZ8 CPU, the bit in the Interrupt Request register is automatically cleared to 0.

! Caution: *The following coding style used to generate software interrupts by setting bits in the Interrupt Request registers is not recommended. All incoming interrupts received between execution of the first LDX command and the final LDX command are lost.*

Poor coding style that can result in lost interrupt requests:
LDX r0, IRQ0
OR r0, MASK
LDX IRQ0, r0

! Caution: *To avoid missing interrupts, use the following coding style to set bits in the Interrupt Request registers:*

Good coding style that avoids lost interrupt requests:
ORX IRQ0, MASK

Watchdog Timer Interrupt Assertion

The Watchdog Timer interrupt behavior is different from interrupts generated by other sources. The Watchdog Timer continues to assert an interrupt as long as the timeout condition continues. As it operates on a different (and usually slower) clock domain than the rest of the device, the Watchdog Timer continues to assert this interrupt for many system clocks until the counter rolls over.

! Caution: *To avoid re-triggerings of the Watchdog Timer interrupt after exiting the associated interrupt service routine, it is recommended that the service routine continues to read from the RSTSTAT register until the* `WDT` *bit is cleared as given in the following coding sample:*

```
CLEARWDT:
LDX r0, RSTSTAT ; read reset status register to clear wdt
   bit
BTJNZ 5, r0, CLEARWDT ; loop until bit is cleared
```

generated and the count value in the Timer High and Low Byte registers is reset to `0001H` and counting resumes. The `INPCAP` bit in TxCTL1 register is set to indicate the timer interrupt is because of an input capture event.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001H` and counting resumes. The `INPCAP` bit in TxCTL1 register is cleared to indicate the timer interrupt is not caused by an input capture event.

Follow the steps below for configuring a timer for CAPTURE RESTART mode and initiating the count:

1. Write to the Timer Control register to:
   - Disable the timer.
   - Configure the timer for CAPTURE RESTART mode. Setting the mode also involves writing to `TMODEHI` bit in TxCTL1 register.
   - Set the prescale value.
   - Set the Capture edge (rising or falling) for the Timer Input.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001H`).
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. Clear the Timer PWM High and Low Byte registers to `0000H`. This allows the software to determine if interrupts were generated by either a Capture or a Reload event. If the PWM High and Low Byte registers still contain `0000H` after the interrupt, the interrupt was generated by a Reload.
5. Enable the timer interrupt, if appropriate, and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt is generated for both input Capture and Reload events. If appropriate, configure the timer interrupt to be generated only at the input Capture event or the Reload event by setting TICONFIG field of the TxCTL1 register.
6. Configure the associated GPIO port pin for the Timer Input alternate function.
7. Write to the Timer Control register to enable the timer and initiate counting.

In CAPTURE mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

COMPARE Mode

In COMPARE mode, the timer counts up to the 16-bit maximum Compare value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the

Follow the steps below to configure a timer for GATED mode and to initiate the count:

1. Write to the Timer Control register to:
   - Disable the timer
   - Configure the timer for Gated mode
   - Set the prescale value
2. Write to the Timer High and Low Byte registers to set the starting count value. Writing these registers only affects the first pass in GATED mode. After the first timer reset in GATED mode, counting always begins at the reset value of `0001H`.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. Enable the timer interrupt, if appropriate, and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt is generated for both input deassertion and Reload events. If appropriate, configure the timer interrupt to be generated only at the input deassertion event or the Reload event by setting TICONFIG field of the TxCTL1 register.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control register to enable the timer.
7. Assert the Timer Input signal to initiate the counting.

CAPTURE/COMPARE Mode

In CAPTURE/COMPARE mode, the timer begins counting on the first external Timer Input transition. The acceptable transition (rising edge or falling edge) is set by the `TPOL` bit in the Timer Control Register. The timer input is the system clock.

Every subsequent acceptable transition (after the first) of the Timer Input signal captures the current count value. The Capture value is written to the Timer PWM High and Low Byte Registers. When the Capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte registers is reset to `0001H`, and counting resumes. The `INPCAP` bit in TxCTL1 register is set to indicate the timer interrupt is caused by an input Capture event.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001H` and counting resumes. The `INPCAP` bit in TxCTL1 register is cleared to indicate the timer interrupt is not because of an input Capture event.

Follow the steps below for configuring a timer for CAPTURE/COMPARE mode and initiating the count:

1. Write to the Timer Control register to:
   - Disable the timer

- Configure the timer for CAPTURE/COMPARE mode
- Set the prescale value
- Set the Capture edge (rising or falling) for the Timer Input

2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001H`).
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. Enable the timer interrupt, if appropriate, and set the timer interrupt priority by writing to the relevant interrupt registers.By default, the timer interrupt are generated for both input Capture and Reload events. If appropriate, configure the timer interrupt to be generated only at the input Capture event or the Reload event by setting TICONFIG field of the TxCTL1 register.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control register to enable the timer.
7. Counting begins on the first appropriate transition of the Timer Input signal. No interrupt is generated by this first edge.

In CAPTURE/COMPARE mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Reading the Timer Count Values

The current count value in the timers can be read while counting (enabled). This capability has no effect on timer operation. When the timer is enabled and the Timer High Byte register is read, the contents of the Timer Low Byte register are placed in a holding register. A subsequent read from the Timer Low Byte register returns the value in the holding register. This operation allows accurate reads of the full 16-bit timer count value while enabled. When the timers are not enabled, a read from the Timer Low Byte register returns the actual value in the counter.

Timer Pin Signal Operation

Timer Output is a GPIO port pin alternate function. The Timer Output is toggled every time the counter is reloaded.

The timer input can be used as a selectable counting source. It shares the same pin as the complementary timer output. When selected by the GPIO Alternate Function Registers, this pin functions as a timer input in all modes except for the DUAL PWM OUTPUT mode. For this mode, there is no timer input available.

Timer Control Register Definitions

Timer 0–1 High and Low Byte Registers

The Timer 0–1 High and Low Byte (TxH and TxL) registers (Table 49 and Table 50) contain the current 16-bit timer count value. When the timer is enabled, a read from TxH causes the value in TxL to be stored in a temporary holding register. A read from TxL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TxL reads the register directly.

Writing to the Timer High and Low Byte registers while the timer is enabled is not recommended. There are no temporary holding registers available for write operations, so simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are written during counting, the 8-bit written value is placed in the counter (High or Low Byte) at the next clock edge. The counter continues counting from the new value.

Table 49. Timer 0–1 High Byte Register (TxH)

BITS	7	6	5	4	3	2	1	0
FIELD	TH							
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
ADDR	F00H, F08H							

Table 50. Timer 0–1 Low Byte Register (TxL)

BITS	7	6	5	4	3	2	1	0
FIELD	TL							
RESET	0	0	0	0	0	0	0	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F01H, F09H							

TH and TL—Timer High and Low Bytes
These 2 bytes, {TH[7:0], TL[7:0]}, contain the current 16-bit timer count value

Timer Reload High and Low Byte Registers

The Timer 0–1 Reload High and Low Byte (TxRH and TxRL) registers (Table 51 and Table 52) store a 16-bit Reload value, {TRH[7:0], TRL[7:0]}. Values written to the Timer Reload High Byte register are stored in a temporary holding register. When a write to the Timer Reload Low Byte register occurs, the temporary holding register value is written to the Timer High Byte register. This operation allows simultaneous updates of the 16-bit Timer Reload value.

010 = Divide by 4
011 = Divide by 8
100 = Divide by 16
101 = Divide by 32
110 = Divide by 64
111 = Divide by 128

TMODE—Timer mode
This field along with the TMODEHI bit in TxCTL0 register determines the operating mode of the timer. TMODEHI is the most significant bit of the Timer mode selection value.

0000 = ONE-SHOT mode
0001 = CONTINUOUS mode
0010 = COUNTER mode
0011 = PWM SINGLE OUTPUT mode
0100 = CAPTURE mode
0101 = COMPARE mode
0110 = GATED mode
0111 = CAPTURE/COMPARE mode
1000 = PWM DUAL OUTPUT mode
1001 = CAPTURE RESTART mode
1010 = COMPARATOR COUNTER Mode

6. 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 7. 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.
7. Write the UART Control 1 register to select the outgoing address bit.
8. Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clear it if sending a data byte.
9. 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.
10. Make any changes to the Multiprocessor Bit Transmitter (MPBT) value, if appropriate and MULTIPROCESSOR mode is enabled,.
11. To transmit additional bytes, return to step 5.

Transmitting Data using the Interrupt-Driven Method

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

1. Write to the UART Baud Rate High and Low Byte registers to set the appropriate 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 acceptable priority.
5. Write to the UART Control 1 register to enable MULTIPROCESSOR (9-bit) mode functions, if MULTIPROCESSOR mode is appropriate.
6. Set the MULTIPROCESSOR Mode Select (MPEN) to Enable MULTIPROCESSOR mode.
7. Write to the UART Control 0 register to:
   - Set the transmit enable bit (TEN) to enable the UART for data transmission.
   - Enable parity, if appropriate and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
   - Set or clear CTSE to enable or disable control from the remote receiver using the $\overline{\text{CTS}}$ pin.
8. Execute an EI instruction to enable interrupts.

The baud rate error relative to the acceptable baud rate is calculated using the following equation:

$$\text{UART Baud Rate Error (\%)} = 100 \times \left(\frac{\text{Actual Data Rate} - \text{Desired Data Rate}}{\text{Desired Data Rate}}\right)$$

For reliable communication, the UART baud rate error must never exceed five percent. Table 71 provides information about data rate errors for 5.5296 MHz System Clock.

Table 71. UART Baud Rates

5.5296 MHz System Clock			
Acceptable Rate (kHz)	**BRG Divisor (Decimal)**	**Actual Rate (kHz)**	**Error (%)**
1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A
250.0	1	345.6	38.24
115.2	3	115.2	0.00
57.6	6	57.6	0.00
38.4	9	38.4	0.00
19.2	18	19.2	0.00
9.60	36	9.60	0.00
4.80	72	4.80	0.00
2.40	144	2.40	0.00
1.20	288	1.20	0.00
0.60	576	0.60	0.00
0.30	1152	0.30	0.00

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 18 displays data reception. When the Infrared Endec is enabled, the UART's RXD signal is internal to the Z8 Encore! XP® F0823 Series products while the IR_RXD signal is received through the RXD pin.

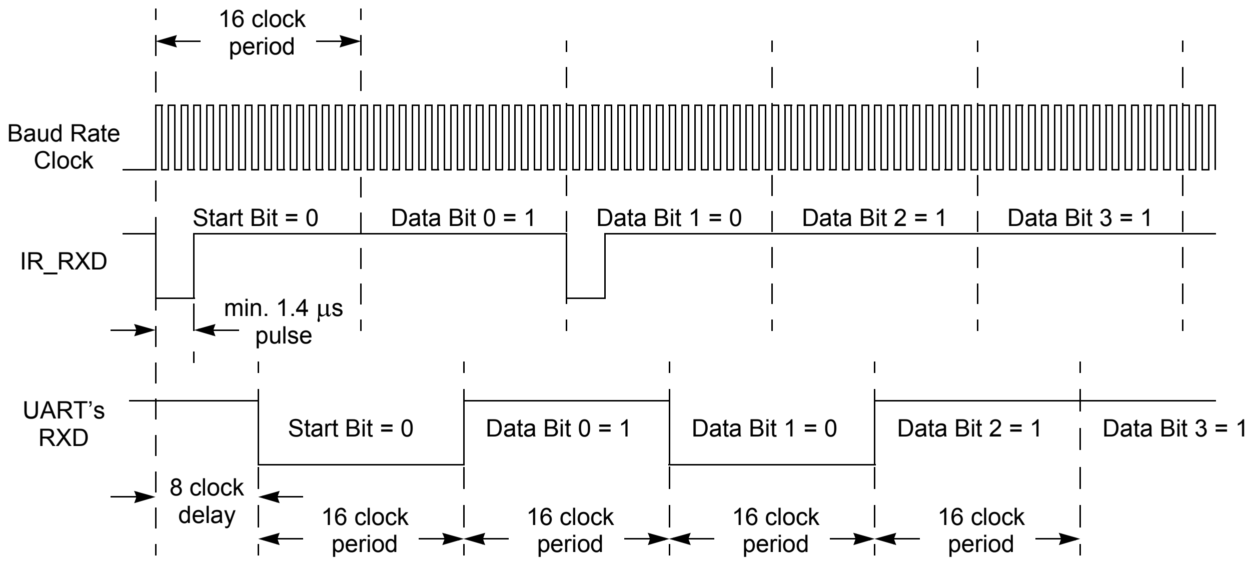


Figure 18. IrDA Data Reception

Infrared 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.*

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. The window remains open until the count again reaches 8 (that is, 24 baud clock periods since the previous pulse was detected), giving the Endec a sampling window of minus four

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 Zilog Calibration Data on page 147. There is 1 byte for offset, 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.

Trim Bit Data Register

The Trim Bid Data (TRMDR) register contains the read or write data for access to the trim option bits.

Table 86. Trim Bit Data Register (TRMDR)

BITS	7	6	5	4	3	2	1	0
FIELD	TRMDR - Trim Bit Data							
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
ADDR	FF7H							

Flash Option Bit Address Space

The first two bytes of Flash program memory at addresses `0000H` and `0001H` are reserved for the user-programmable Flash option bits.

Flash Program Memory Address 0000H

Table 87. Flash Option Bits at Program Memory Address 0000H

BITS	7	6	5	4	3	2	1	0
FIELD	WDT_RES	WDT_AO	Reserved		VBO_AO	FRP	Reserved	FWP
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Program Memory 0000H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

WDT_RES—Watchdog Timer Reset
0 = Watchdog Timer time-out generates an interrupt request. Interrupts must be globally enabled for the eZ8 CPU to acknowledge the interrupt request.
1 = Watchdog Timer time-out causes a system reset. This setting is the default for unprogrammed (erased) Flash.

WDT_AO—Watchdog Timer Always ON
0 = Watchdog Timer is automatically enabled upon application of system power. Watchdog Timer can not be disabled.
1 = Watchdog Timer is enabled upon execution of the WDT instruction. Once enabled, the

```
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

OCD Status Register

The OCD Status register reports status information about the current state of the debugger and the system.

Table 100. OCD Status Register (OCDSTAT)

BITS	7	6	5	4	3	2	1	0
FIELD	DBG	HALT	FRPENB	Reserved				
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R

DBG—Debug Status
0 = NORMAL mode
1 = DEBUG mode

HALT—HALT Mode
0 = Not in HALT mode
1 = In HALT mode

FRPENB—Flash Read Protect Option Bit Enable
0 = FRP bit enabled, that allows disabling of many OCD commands
1 = FRP bit has no effect

Reserved—0 when read

! Caution: *Unintentional accesses to the oscillator control register can actually stop the chip by switching to a non-functioning oscillator. To prevent this condition, the oscillator control block employs a register unlocking/locking scheme.*

OSC Control Register Unlocking/Locking

To write the oscillator control register, unlock it by making two writes to the OSCCTL register with the values `E7H` followed by `18H`. A third write to the OSCCTL register changes the value of the actual register and returns the register to a locked state. Any other sequence of oscillator control register writes has no effect. The values written to unlock the register must be ordered correctly, but are not necessarily consecutive. It is possible to write to or read from other registers within the unlocking/locking operation.

When selecting a new clock source, the primary oscillator failure detection circuitry and the Watchdog Timer oscillator failure circuitry must be disabled. If POFEN and WOFEN are not disabled prior to a clock switch-over, it is possible to generate an interrupt for a failure of either oscillator. The Failure detection circuitry can be enabled anytime after a successful write of OSCSEL in the oscillator control register.

The internal precision oscillator is enabled by default. If the user code changes to a different oscillator, it is appropriate to disable the IPO for power savings. Disabling the IPO does not occur automatically.

Clock Failure Detection and Recovery

Primary Oscillator Failure

Z8 Encore! XP® F0823 Series devices can generate non-maskable interrupt-like events when the primary oscillator fails. To maintain system function in this situation, the clock failure recovery circuitry automatically forces the Watchdog Timer oscillator to drive the system clock. The Watchdog Timer oscillator must be enabled to allow the recovery. Although this oscillator runs at a much slower speed than the original system clock, the CPU continues to operate, allowing execution of a clock failure vector and software routines that either remedy the oscillator failure or issue a failure alert. This automatic switch-over is not available if the Watchdog Timer is the primary oscillator. It is also unavailable if the Watchdog Timer oscillator is disabled, though it is not necessary to enable the Watchdog Timer reset function outlined in the Watchdog Timer on page 87.

The primary oscillator failure detection circuitry asserts if the system clock frequency drops below 1 kHz ±50%. If an external signal is selected as the system oscillator, it is possible that a very slow but non-failing clock can generate a failure condition. Under these conditions, do not enable the clock failure circuitry (POFEN must be deasserted in the OSCCTL register).

INTEN—Internal Precision Oscillator Enable
1 = Internal precision oscillator is enabled
0 = Internal precision oscillator is disabled

Reserved—R/W bits must be 0 during writes; 0 when read

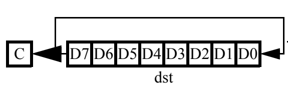
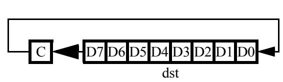
WDTEN—Watchdog Timer Oscillator Enable
1 = Watchdog Timer oscillator is enabled
0 = Watchdog Timer oscillator is disabled

POFEN—Primary Oscillator Failure Detection Enable
1 = Failure detection and recovery of primary oscillator is enabled
0 = Failure detection and recovery of primary oscillator is disabled

WDFEN—Watchdog Timer Oscillator Failure Detection Enable
1 = Failure detection of Watchdog Timer oscillator is enabled
0 = Failure detection of Watchdog Timer oscillator is disabled

SCKSEL—System Clock Oscillator Select
000 = Internal precision oscillator functions as system clock at 5.53 MHz
001 = Internal precision oscillator functions as system clock at 32 kHz
010 = Reserved
011 = Watchdog Timer oscillator functions as system clock
100 = External clock signal on PB3 functions as system clock
101 = Reserved
110 = Reserved
111 = Reserved

Table 115. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Opcode(s) (Hex)	Flags C	Flags Z	Flags S	Flags V	Flags D	Flags H	Fetch Cycles	Instr. Cycles
OR dst, src	dst ← dst OR src	r	r	42	–	*	*	0	–	–	2	3
		r	Ir	43							2	4
		R	R	44							3	3
		R	IR	45							3	4
		R	IM	46							3	3
		IR	IM	47							3	4
ORX dst, src	dst ← dst OR src	ER	ER	48	–	*	*	0	–	–	4	3
		ER	IM	49							4	3
POP dst	dst ← @SP SP ← SP + 1	R		50	–	–	–	–	–	–	2	2
		IR		51							2	3
POPX dst	dst ← @SP SP ← SP + 1	ER		D8	–	–	–	–	–	–	3	2
PUSH src	SP ← SP – 1 @SP ← src	R		70	–	–	–	–	–	–	2	2
		IR		71							2	3
		IM		IF70							3	2
PUSHX src	SP ← SP – 1 @SP ← src	ER		C8	–	–	–	–	–	–	3	2
RCF	C ← 0			CF	0	–	–	–	–	–	1	2
RET	PC ← @SP SP ← SP + 2			AF	–	–	–	–	–	–	1	4
RL dst		R		90	*	*	*	*	–	–	2	2
		IR		91							2	3
RLC dst		R		10	*	*	*	*	–	–	2	2
		IR		11							2	3

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

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, 2 **PUSH** IM															
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

Table 118. DC Characteristics (Continued)

Symbol	Parameter	T_A = -40 °C to +105 °C (unless otherwise specified)			Units	Conditions
		Minimum	Typical	Maximum		
V_{OH2}	High Level Output Voltage	2.4	–	–	V	I_{OH} = -20 mA; V_{DD} = 3.3 V High Output Drive enabled.
I_{IH}	Input Leakage Current	–	$\pm$0.002	$\pm$5	µA	$V_{IN} = V_{DD}$ V_{DD} = 3.3 V;
I_{IL}	Input Leakage Current	–	$\pm$0.007	$\pm$5	µA	$V_{IN} = V_{SS}$ V_{DD} = 3.3 V;
I_{TL}	Tristate Leakage Current	–	–	$\pm$5	µA	
I_{LED}	Controlled Current Drive	1.8	3	4.5	mA	{AFS2,AFS1} = {0,0}
		2.8	7	10.5	mA	{AFS2,AFS1} = {0,1}
		7.8	13	19.5	mA	{AFS2,AFS1} = {1,0}
		12	20	30	mA	{AFS2,AFS1} = {1,1}
C_{PAD}	GPIO Port Pad Capacitance	–	8.0[2]	–	pF	
C_{XIN}	XIN Pad Capacitance	–	8.0[2]	–	pF	
C_{XOUT}	XOUT Pad Capacitance	–	9.5[2]	–	pF	
I_{PU}	Weak Pull-up Current	30	100	350	µA	V_{DD} = 3.0 V–3.6 V
V_{RAM}	RAM Data Retention Voltage	TBD			V	Voltage at which RAM retains static values; no reading or writing is allowed.

Notes
1. This condition excludes all pins that have on-chip pull-ups, when driven Low.
2. These values are provided for design guidance only and are not tested in production.