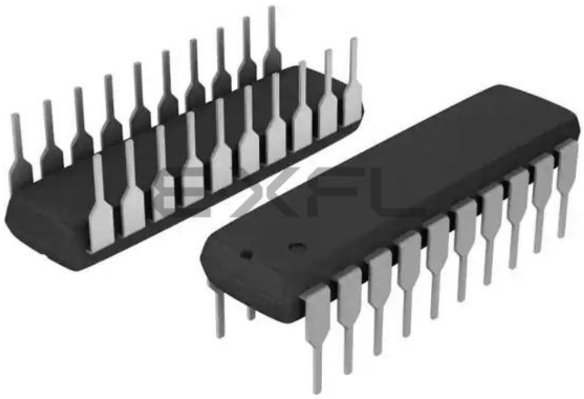

Welcome to E-XFL.COM

## 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 | Through Hole |
| Package / Case | 20-DIP (0.300", 7.62mm) |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/zilog/z8f0423ph005sc |

# Revision History

Each instance in Revision History reflects a change to this document from its previous revision. For more details, refer to the corresponding pages and appropriate links in the table below.

| Date | Revision Level | Description | Page No |
|---|---|---|---|
| March 2008 | 14 | Changed title to Z8 Encore! XP F0823 Series and the contents to match the title. | All |
| December 2007 | 13 | Updated title from Z8 Encore! 8K and 4K Series to Z8 Encore! XP Z8F0823 Series. Updated Figure 3, Table 15, Table 35, Table 59 through Table 61, Table 119, and Part Number Suffix Designations section. | 8, 39, 59, 91, 196, and 226 |
| August 2007 | 12 | Updated Table 1, Table 16, and Program Memory section. | 2, 42, and 13 |
| June 2007 | 11 | Updated to combine Z8 Encore! 8K and Z8 Encore! 4K Series. | All |
| December 2006 | 10 | Updated Ordering Information chapter. | 217 |

# Overview

Zilog's Z8 Encore! XP® microcontroller unit (MCU) family of products are the first Zilog® microcontroller products based on the 8-bit eZ8 CPU core. Z8 Encore! XP F0823 Series products expand upon Zilog's extensive line of 8-bit microcontrollers. The Flash in-circuit programming capability allows for faster development time and program changes in the field. The new eZ8 CPU is upward compatible with existing Z8® instructions. The rich peripheral set of Z8 Encore! XP F0823 Series makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

## Features

The key features of Z8 Encore! XP F0823 Series include:

- 5 MHz eZ8 CPU
- 1 KB, 2 KB, 4 KB, or 8 KB Flash memory with in-circuit programming capability
- 256 B, 512 B, or 1 KB register RAM
- 6 to 24 I/O pins depending upon package
- Internal precision oscillator (IPO)
- Full-duplex UART
- The universal asynchronous receiver/transmitter (UART) baud rate generator (BRG) can be configured and used as a basic 16-bit timer
- Infrared data association (IrDA)-compliant infrared encoder/decoders, integrated with UART
- Two enhanced 16-bit timers with capture, compare, and PWM capability
- Watchdog Timer (WDT) with dedicated internal RC oscillator
- On-Chip Debugger (OCD)
- Optional 8-channel, 10-bit Analog-to-Digital Converter (ADC)
- On-Chip analog comparator
- Up to 20 vectored interrupts
- Direct LED drive with programmable drive strengths
- Voltage Brownout (VBO) protection
- Power-On Reset (POR)

**Table 3. Signal Descriptions (Continued)**

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **Analog** | | |
| ANA[7:0] | I | Analog port. These signals are used as inputs to the ADC. The ANA0, ANA1, and ANA2 pins can also access the inputs and output of the integrated transimpedance amplifier. |
| VREF | I/O | Analog-to-Digital Converter reference voltage input. |
| **Clock Input** | | |
| CLKIN | I | Clock Input Signal. This pin can be used to input a TTL-level signal to be used as the system clock. |
| **LED Drivers** | | |
| LED | O | Direct LED drive capability. All port C pins have the capability to drive an LED without any other external components. These pins have programmable drive strengths set by the GPIO block. |
| **On-Chip Debugger** | | |
| DBG | I/O | Debug. This signal is the control and data input and output to and from the OCD.<br>**!** **Caution:** *The DBG pin is open-drain and requires an external pull-up resistor to ensure proper operation.* |
| **Reset** | | |
| $\overline{\text{RESET}}$ | I/O | RESET. Generates a reset when asserted (driven Low). Also serves as a reset indicator; the Z8 Encore! XP forces this pin Low when in reset. This pin is open-drain and features an enabled internal pull-up resistor. |
| **Power Supply** | | |
| $V_{DD}$ | I | Digital Power Supply. |
| $AV_{DD}$ | I | Analog Power Supply. |
| $V_{SS}$ | I | Digital Ground. |
| $AV_{SS}$ | I | Analog Ground. |

**Note:** The $AV_{DD}$ and $AV_{SS}$ signals are available only in 28-pin packages with ADC. They are replaced by PB6 and PB7 on 28-pin packages without ADC.

# Pin Characteristics

Table 4 provides detailed information about the characteristics for each pin available on Z8 Encore! XP F0823 Series 20- and 28-pin devices. Data in Table 4 is sorted alphabetically by the pin symbol mnemonic.

**! 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
```

**Table 43. IRQ2 Enable and Priority Encoding (Continued)**

| IRQ2ENH[*x*] | IRQ2ENL[*x*] | Priority | Description |
|---|---|---|---|
| 1 | 1 | Level 3 | High |

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

**Table 44. IRQ2 Enable High Bit Register (IRQ2ENH)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | Reserved | | | | C3ENH | C2ENH | C1ENH | C0ENH |
| 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 | FC7H | | | | | | | |

Reserved—Must be 0

C3ENH—Port C3 Interrupt Request Enable High Bit
C2ENH—Port C2 Interrupt Request Enable High Bit
C1ENH—Port C1 Interrupt Request Enable High Bit
C0ENH—Port C0 Interrupt Request Enable High Bit

**Table 45. IRQ2 Enable Low Bit Register (IRQ2ENL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | Reserved | | | | C3ENL | C2ENL | C1ENL | C0ENL |
| 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 | FC8H | | | | | | | |

Reserved—Must be 0

C3ENL—Port C3 Interrupt Request Enable Low Bit
C2ENL—Port C2 Interrupt Request Enable Low Bit
C1ENL—Port C1 Interrupt Request Enable Low Bit
C0ENL—Port C0 Interrupt Request Enable Low Bit

## Interrupt Edge Select Register

The Interrupt Edge Select (IRQES) register (Table 46) determines whether an interrupt is generated for the rising edge or falling edge on the selected GPIO Port A or Port D input pin.

**Table 48. Interrupt Control Register (IRQCTL)**

| BITS | 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 |
| ADDR | FCFH | | | | | | | |

IRQE—Interrupt Request Enable
This bit is set to 1 by executing an `EI` (Enable Interrupts) or `IRET` (Interrupt Return) 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

Reserved—0 when read

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

## Watchdog Timer Refresh

When first enabled, the WDT is loaded with the value in the Watchdog Timer Reload registers. The Watchdog Timer counts down to `000000H` unless a `WDT` instruction is executed by the eZ8 CPU. Execution of the `WDT` instruction causes the down counter to be reloaded with the WDT Reload value stored in the Watchdog Timer Reload registers. Counting resumes following the reload operation.

When Z8 Encore! XP® F0823 Series devices are operating in DEBUG Mode (using the OCD), the Watchdog Timer is continuously refreshed to prevent any Watchdog Timer time-outs.

## Watchdog Timer Time-Out Response

The Watchdog Timer times out when the counter reaches `000000H`. A time-out of the Watchdog Timer generates either an interrupt or a system reset. The WDT_RES Flash Option Bit determines the time-out response of the Watchdog Timer. For information on programming of the WDT_RES Flash Option Bit, see Flash Option Bits on page 141.

### WDT Interrupt in Normal Operation

If configured to generate an interrupt when a time-out occurs, the Watchdog Timer issues an interrupt request to the interrupt controller and sets the `WDT` status bit in the Watchdog Timer Control register. If interrupts are enabled, the eZ8 CPU responds to the interrupt request by fetching the Watchdog Timer interrupt vector and executing code from the vector address. After time-out and interrupt generation, the Watchdog Timer counter rolls over to its maximum value of `FFFFFH` and continues counting. The Watchdog Timer counter is not automatically returned to its Reload Value.

The Reset Status Register (see Reset Status Register on page 28) must be read before clearing the WDT interrupt. This read clears the WDT time-out Flag and prevents further WDT interrupts for immediately occurring.

### WDT Interrupt in STOP Mode

If configured to generate an interrupt when a time-out occurs and Z8 Encore! XP F0823 Series are in STOP mode, the Watchdog Timer automatically initiates a Stop Mode Recovery and generates an interrupt request. Both the WDT status bit and the `STOP` bit in the Watchdog Timer Control register are set to 1 following a WDT time-out in STOP mode. For more information on Stop Mode Recovery, see Reset and Stop Mode Recovery on page 21.

If interrupts are enabled, following completion of the Stop Mode Recovery the eZ8 CPU responds to the interrupt request by fetching the Watchdog Timer interrupt vector and executing code from the vector address.

## External Driver Enable

The UART provides a Driver Enable (DE) signal for off-chip bus transceivers. This feature reduces the software overhead associated with using a GPIO pin to control the transceiver when communicating on a multi-transceiver bus, such as RS-485.

Driver Enable is an active High signal that envelopes the entire transmitted data frame including parity and Stop bits as displayed in Figure 14. The Driver Enable signal asserts when a byte is written to the UART Transmit Data register. The Driver Enable signal asserts at least one UART bit period and no greater than two UART bit periods before the Start bit is transmitted. This allows a setup time to enable the transceiver. The Driver Enable signal deasserts one system clock period after the final Stop bit is transmitted. This one system clock delay allows both time for data to clear the transceiver before disabling it, as well as the ability to determine if another character follows the current character. In the event of back to back characters (new data must be written to the Transmit Data Register before the previous character is completely transmitted) the DE signal is not deasserted between characters. The `DEPOL` bit in the UART Control Register 1 sets the polarity of the Driver Enable signal.

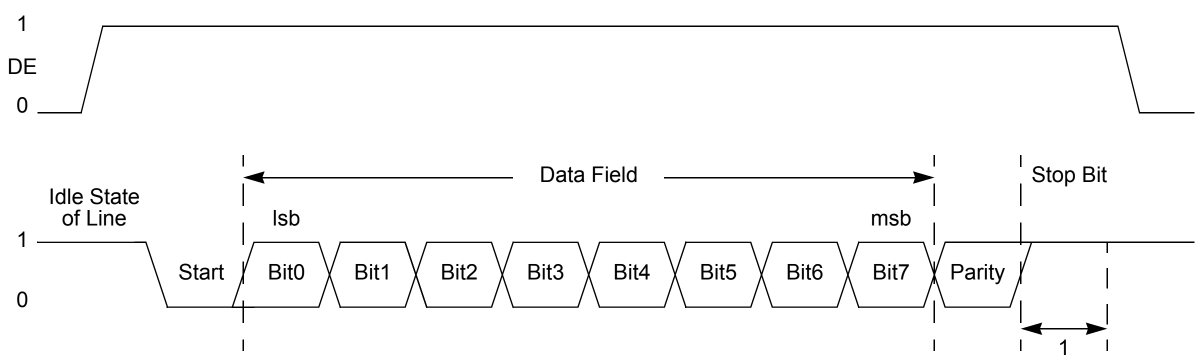


**Figure 14. UART Driver Enable Signal Timing (shown with 1 Stop Bit and Parity)**

The Driver Enable to Start bit setup time is calculated as follows:

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

## UART Interrupts

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

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

SBRK—Send Break
This bit pauses or breaks data transmission. Sending a break interrupts any transmission in progress, so ensure that the transmitter has finished sending data before setting this bit.
0 = No break is sent
1 = Forces a break condition by setting the output of the transmitter to zero

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

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

**Table 67. UART Control 1 Register (U0CTL1)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | MPMD[1] | MPEN | MPMD[0] | MPBT | DEPOL | BRGCTL | RDAIRQ | IREN |
| RESET | 0 | 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 | F43H | | | | | | | |

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

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

MPBT—Multiprocessor Bit Transmit
This bit is applicable only when MULTIPROCESSOR (9-bit) mode is enabled. The 9th bit is used by the receiving device to determine if the data byte contains address or data information.

5. When the conversion is complete, the ADC control logic performs the following operations:
   - 11-bit two's-complement result written to {ADCD_H[7:0], ADCD_L[7:5]}.
   - `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.*

Follow the steps below 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.

# Flash Option Bits

Programmable Flash option bits allow user configuration of certain aspects of Z8 Encore! XP® F0823 Series operation. The feature configuration data is stored in the Flash program memory and loaded into holding registers during Reset. The features available for control through the Flash Option Bits include:

- Watchdog Timer time-out response selection–interrupt or system reset
- Watchdog Timer always on (enabled at Reset)
- The ability to prevent unwanted read access to user code in Program Memory
- The ability to prevent accidental programming and erasure of all or a portion of the user code in Program Memory
- Voltage Brownout configuration-always enabled or disabled during STOP mode to reduce STOP mode power consumption
- Factory trimming information for the internal precision oscillator
- Factory calibration values for ADC
- Factory serialization and randomized lot identifier (optional)

## Operation

### Option Bit Configuration By Reset

Each time the Flash Option Bits are programmed or erased, the device must be Reset for the change to take effect. During any reset operation (System Reset, Power-On Reset, or Stop Mode Recovery), the Flash Option Bits are automatically read from the Flash Program Memory and written to Option Configuration registers. The Option Configuration registers control operation of the devices within the Z8 Encore! XP F0823 Series. Option Bit control is established before the device exits Reset and the eZ8 CPU begins code execution. The Option Configuration registers are not part of the Register File and are not accessible for read or write access.

### Option Bit Types

#### User Option Bits

The user option bits are contained in the first two bytes of program memory. Access to these bits has been provided because these locations contain application-specific device

**Note:** *This bit only enables the crystal oscillator. Its selection as system clock must be done manually.*
*0 = Crystal oscillator is enabled during reset, resulting in longer reset timing*
*1 = Crystal oscillator is disabled during reset, resulting in shorter reset timing*

**Warning:** *Programming the XTLDIS bit to zero on 8-pin versions of this device prevents any further communication via the debug pin. This is due to the fact that the XIN and DBG functions are shared on pin 2 of this package. Do not program this bit to zero on 8-pin devices unless no further debugging or Flash programming is required.*

## Trim Bit Address Space

All available Trim bit addresses and their functions are listed in Table 89 through Table 91.

### Trim Bit Address 0000H—Reserved

**Table 89. Trim Options Bits at Address 0000H**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | Reserved | | | | | | | |
| 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 | Information Page Memory 0020H | | | | | | | |
| **Note:** U = Unchanged by Reset. R/W = Read/Write. | | | | | | | | |

Reserved—Altering this register may result in incorrect device operation.

### Trim Bit Address 0001H—Reserved

**Table 90. Trim Option Bits at 0001H**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | Reserved | | | | | | | |
| 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 | Information Page Memory 0021H | | | | | | | |
| **Note:** U = Unchanged by Reset. R/W = Read/Write. | | | | | | | | |

# Assembly Language Syntax

For proper instruction execution, eZ8 CPU assembly language syntax requires that the operands be written as ‘destination, source’. After assembly, the object code usually has the operands in the order ‘source, destination’, but ordering is opcode-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. You must follow this binary format if you prefer manual program coding or intend to implement your own assembler.

### Example 1

If the contents of Registers `43H` and `08H` are added and the result is stored in `43H`, the assembly syntax and resulting object code is:

**Table 103. Assembly Language Syntax Example 1**

| | | | | |
|---|---|---|---|---|
| **Assembly Language Code** | `ADD` | `43H,` | `08H` | `(ADD dst, src)` |
| **Object Code** | `04` | `08` | `43` | `(OPC src, dst)` |

### Example 2

In general, when an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or, using Escaped Mode Addressing, a Working Register R0–R15. If the contents of Register `43H` and Working Register R8 are added and the result is stored in `43H`, the assembly syntax and resulting object code is:

**Table 104. Assembly Language Syntax Example 2**

| | | | | |
|---|---|---|---|---|
| **Assembly Language Code** | `ADD` | `43H,` | `R8` | `(ADD dst, src)` |
| **Object Code** | `04` | `E8` | `43` | `(OPC src, dst)` |

See the device-specific Product Specification to determine the exact register file range available. The register file size varies, depending on the device type.

# eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary and Description sections, the operands, condition codes, status flags, and address modes are represented by a notational shorthand that is described in Table 105.

### Table 115. eZ8 CPU Instruction Summary (Continued)

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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<br>r ← r + 1<br>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<br>r ← r + 1<br>rr ← rr + 1 | Ir | Irr | 83 | – | – | – | – | – | – | 2 | 9 |
| | | Irr | Ir | 93 | | | | | | | 2 | 9 |
| LDWX dst, src | dst ← src | ER | ER | 1FE8 | – | – | – | – | – | – | 5 | 4 |
| LDX dst, src | dst ← src | r | ER | 84 | – | – | – | – | – | – | 3 | 2 |
| | | Ir | ER | 85 | | | | | | | 3 | 3 |
| | | R | IRR | 86 | | | | | | | 3 | 4 |
| | | IR | IRR | 87 | | | | | | | 3 | 5 |
| | | r | X(rr) | 88 | | | | | | | 3 | 4 |
| | | X(rr) | r | 89 | | | | | | | 3 | 4 |
| | | ER | r | 94 | | | | | | | 3 | 2 |
| | | ER | Ir | 95 | | | | | | | 3 | 3 |
| | | IRR | R | 96 | | | | | | | 3 | 4 |
| | | IRR | IR | 97 | | | | | | | 3 | 5 |
| | | ER | ER | E8 | | | | | | | 4 | 2 |
| | | ER | IM | E9 | | | | | | | 4 | 2 |
| LEA dst, X(src) | dst ← src + X | r | X(r) | 98 | – | – | – | – | – | – | 3 | 3 |
| | | rr | X(rr) | 99 | | | | | | | 3 | 5 |
| MULT dst | dst[15:0] ←<br>dst[15:8] * dst[7:0] | RR | | F4 | – | – | – | – | – | – | 2 | 8 |
| NOP | No operation | | | 0F | – | – | – | – | – | – | 1 | 2 |
| Flags Notation: | * = Value is a function of the result of the operation.<br>– = Unaffected<br>X = Undefined | | | | 0 = Reset to 0<br>1 = Set to 1 | | | | | | | |

## Table 115. eZ8 CPU Instruction Summary (Continued)

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SUBX dst, src | dst ← dst – src | ER | ER | 28 | * | * | * | * | 1 | * | 4 | 3 |
| | | ER | IM | 29 | | | | | | | 4 | 3 |
| SWAP dst | dst[7:4] ↔ dst[3:0] | R | | F0 | X | * | * | X | – | – | 2 | 2 |
| | | IR | | F1 | | | | | | | 2 | 3 |
| TCM dst, src | (NOT dst) AND src | r | r | 62 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 63 | | | | | | | 2 | 4 |
| | | R | R | 64 | | | | | | | 3 | 3 |
| | | R | IR | 65 | | | | | | | 3 | 4 |
| | | R | IM | 66 | | | | | | | 3 | 3 |
| | | IR | IM | 67 | | | | | | | 3 | 4 |
| TCMX dst, src | (NOT dst) AND src | ER | ER | 68 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 69 | | | | | | | 4 | 3 |
| TM dst, src | dst AND src | r | r | 72 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 73 | | | | | | | 2 | 4 |
| | | R | R | 74 | | | | | | | 3 | 3 |
| | | R | IR | 75 | | | | | | | 3 | 4 |
| | | R | IM | 76 | | | | | | | 3 | 3 |
| | | IR | IM | 77 | | | | | | | 3 | 4 |
| TMX dst, src | dst AND src | ER | ER | 78 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 79 | | | | | | | 4 | 3 |
| TRAP Vector | SP ← SP – 2<br>@SP ← PC<br>SP ← SP – 1<br>@SP ← FLAGS<br>PC ← @Vector | | Vector | F2 | – | – | – | – | – | – | 2 | 6 |
| WDT | | | | 5F | – | – | – | – | – | – | 1 | 2 |
| Flags Notation: | * = Value is a function of the result of the operation.<br>– = Unaffected<br>X = Undefined | | | | 0 = Reset to 0<br>1 = Set to 1 | | | | | | | |

## J

## T



## U



## V



## W

# Customer Support

For answers to technical questions about the product, documentation, or any other issues with Zilog's offerings, please visit Zilog's Knowledge Base at http://www.zilog.com/kb.

For any comments, detail technical questions, or reporting problems, please visit Zilog's Technical Support at http://support.zilog.com.