

Welcome to [E-XFL.COM](#)

### What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

Product Status	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	5MHz
Connectivity	IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, LED, POR, PWM, WDT
Number of I/O	22
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 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.600", 15.24mm)
Supplier Device Package	-
Purchase URL	<a href="https://www.e-xfl.com/product-detail/zilog/z8f0423pj005eg2156">https://www.e-xfl.com/product-detail/zilog/z8f0423pj005eg2156</a>

# Revision History

Each instance in this document's revision history reflects a change from its previous edition. For more details, refer to the corresponding page(s) or appropriate links furnished in the table below.

Date	Revision Level	Chapter/Section	Description	Page No.
Sep	15	LED Drive Enable Register	Clarified statement surrounding the Alternate Function Register as it relates to the LED function; revised Flash Sector Protect Register description; revised Packaging chapter.	<u>51</u> , <u>144</u> , <u>210</u>
Mar 2008	14	n/a	Changed branding to <i>Z8 Encore! XP F0823 Series</i> where appropriate.	All
Dec 2007	13	Pin Description, General-Purpose Input/Output, Interrupt Controller, Watchdog Timer, Electrical Characteristics, and Ordering Information	Updated title from <i>Z8 Encore! 8K and 4K Series</i> to <i>Z8 Encore! XP Z8F0823 Series</i> . Updated Figure 3, Table 15, Table 35, Tables 59 through 61, Table 119 and Part Number Suffix Designations section.	<u>8</u> , <u>36</u> , <u>60</u> , <u>95</u> , <u>199</u> , and <u>220</u>
Aug 2007	12	Part Selection Guide, External Clock Setup, and Program Memory	Updated Table 1, Table 16, and Program Memory section.	<u>2</u> , <u>35</u> , and <u>13</u>
Jun 2007	11	n/a	Updated to combine Z8 Encore! 8K and Z8 Encore! 4K Series.	All
Dec 2006	10	Ordering Information	Updated Ordering Information chapter.	<u>211</u>

## 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 15 lists the port pins available with each device and package type.

**Table 15. 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

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

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port A <sup>1</sup>	PA0	T0IN/T0OUT	Timer 0 Input/Timer 0 Output Complement	N/A
		Reserved		
	PA1	T0OUT	Timer 0 Output	
		Reserved		
	PA2	DE0	UART 0 Driver Enable	
		Reserved		
	PA3	CTS0	UART 0 Clear to Send	
		Reserved		
	PA4	RXD0/IRRX0	UART 0 / IrDA 0 Receive Data	
		Reserved		
	PA5	TXD0/IRTX0	UART 0 / IrDA 0 Transmit Data	
		Reserved		
	PA6	T1IN/T1OUT <sup>2</sup>	Timer 1 Input/Timer 1 Output Complement	
		Reserved		
	PA7	T1OUT	Timer 1 Output	
		Reserved		

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<sub>REF</sub> 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<sub>REF</sub> is available on PC2 in 20-pin parts only.



Table 18. GPIO Port Registers and Subregisters (Continued)

Port Register Mnemonic	Port Register Name
PxHDE	High Drive Enable.
PxSMRE	Stop Mode Recovery Source Enable.
PxPUE	Pull-up Enable.
PxAFS1	Alternate Function Set 1.
PxAFS2	Alternate Function Set 2.

## Port A–C Address Registers

The Port A–C Address registers select the GPIO port functionality accessible through the Port A–C Control registers. The Port A–C Address and Control registers combine to provide access to all GPIO port controls (Table 19).

Table 19. Port A–C GPIO Address Registers (PxADDR)

Bit	7	6	5	4	3	2	1	0
Field	PADDR[7:0]							
RESET	00H							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FD0H, FD4H, FD8H							

Bit	Description
[7:0] PADDR	<b>Port Address</b> The Port Address selects one of the subregisters accessible through the Port Control Register. See Table 20 for each subregister function.

Table 20. PADDR[7:0] Subregister Functions

PADDR[7:0]	Port Control Subregister Accessible Using the Port A–C Control Registers
00H	No function. Provides some protection against accidental Port reconfiguration.
01H	Data Direction.
02H	Alternate Function.
03H	Output Control (Open-Drain).
04H	High Drive Enable.

**Table 23. Port A–C Alternate Function Subregisters (PxAF)**

Bit	7	6	5	4	3	2	1	0
Field	AF7	AF6	AF5	AF4	AF3	AF2	AF1	AF0
RESET	00H (Ports A–C); 04H (Port A of 8-pin device)							
R/W	R/W							
Address	If 02H in Port A–C Address Register, accessible through the Port A–C Control Register							

Bit	Description
[7:0] AFx	<b>Port Alternate Function enabled</b> 0 = The port pin is in NORMAL Mode and the DDx bit in the Port A–C Data Direction Subregister determines the direction of the pin. 1 = The alternate function selected through Alternate Function Set subregisters is enabled. Port pin operation is controlled by the alternate function.

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

## Port A–C Output Control Subregisters

The Port A–C Output Control Subregister (Table 24) is accessed through the Port A–C Control Register by writing 03H to the Port A–C Address Register. Setting the bits in the Port A–C Output Control subregisters to 1 configures the specified port pins for open-drain operation. These subregisters affect the pins directly and, as a result, alternate functions are also affected.

**Table 24. Port A–C Output Control Subregisters (PxOC)**

Bit	7	6	5	4	3	2	1	0
Field	POC7	POC6	POC5	POC4	POC3	POC2	POC1	POC0
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 03H in Port A–C Address Register, accessible through the Port A–C Control Register							

Bit	Description
[7:0] POCx	<b>Port Output Control</b> These bits function independently of the alternate function bit and always disable the drains if set to 1. 0 = The drains are enabled for any output mode (unless overridden by the alternate function). 1 = The drain of the associated pin is disabled (open-drain mode).

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

## Port A–C Stop Mode Recovery Source Enable Subregisters

The Port A–C Stop Mode Recovery Source Enable Subregister (Table 26) is accessed through the Port A–C Control Register by writing 05H to the Port A–C Address Register. Setting the bits in the Port A–C Stop Mode Recovery Source Enable subregisters to 1 configures the specified Port pins as a Stop Mode Recovery source. During STOP Mode, any logic transition on a Port pin enabled as a Stop Mode Recovery source initiates Stop Mode Recovery.

**Table 26. Port A–C Stop Mode Recovery Source Enable Subregisters (PSMREx)**

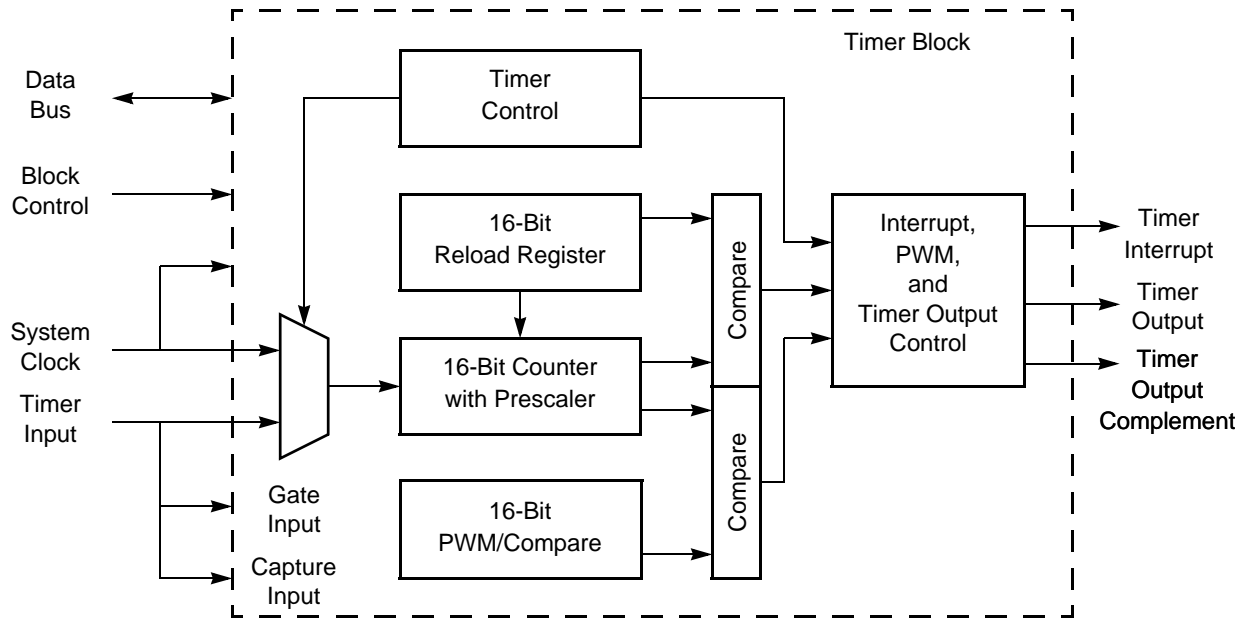
Bit	7	6	5	4	3	2	1	0
Field	PSMRE7	PSMRE6	PSMRE5	PSMRE4	PSMRE3	PSMRE2	PSMRE1	PSMRE0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 05H in Port A–C Address Register, accessible through the Port A–C Control Register							

Bit	Description
[7:0]	<b>Port Stop Mode Recovery Source Enabled.</b>
PSMREx	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.

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

## Architecture

Figure 9 displays the architecture of the timers.



**Figure 9. Timer Block Diagram**

## Operation

The timers are 16-bit up-counters. Minimum time-out delay is set by loading the value 0001H into the Timer Reload High and Low Byte registers and setting the prescale value to 1. Maximum time-out delay is set by loading the value 0000H into the Timer Reload High and Low Byte registers and setting the prescale value to 128. If the Timer reaches FFFFH, the timer rolls over to 0000H and continues counting.

## Timer Operating Modes

The timers can be configured to operate in the following modes:

### ONE-SHOT Mode

In ONE-SHOT Mode, the timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the reload value, the timer generates an interrupt and the count value in the Timer High and Low Byte registers is reset to 0001H. The timer is automatically disabled and stops counting.

1. Write to the Timer Control Register to:
  - Disable the timer
  - Configure the timer for CONTINUOUS Mode
  - Set the prescale value
  - If using the Timer Output alternate function, set the initial output level (High or Low)
2. Write to the Timer High and Low Byte registers to set the starting count value (usually 0001H). This action only affects the first pass in CONTINUOUS Mode. After the first timer reload in CONTINUOUS 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.
5. Configure the associated GPIO port pin (if using the Timer Output function) for the Timer Output alternate function.
6. Write to the Timer Control Register to enable the timer and initiate counting.

In CONTINUOUS Mode, the system clock always provides the timer input. The timer period is computed via the following equation:

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

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

### **COUNTER Mode**

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

---

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

---

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. Also, if the Timer Output alternate function is

## Software Compensation Procedure

The value read from the ADC high and low byte registers are uncompensated. The user mode software must apply gain and offset correction to this uncompensated value for maximum accuracy. The following formula yields the compensated value:

$$ADC_{comp} = (ADC_{uncomp} - OFFCAL) + ((ADC_{uncomp} - OFFCAL) * GAINCAL) / 2$$

where GAINCAL is the gain calibration byte, OFFCAL is the offset calibration byte and  $ADC_{uncomp}$  is the uncompensated value read from the ADC. The OFFCAL value is in two's complement format, as are the compensated and uncompensated ADC values.

---

► **Note:** The offset compensation is performed first, followed by the gain compensation. One bit of resolution is lost because of rounding on both the offset and gain computations. As a result the ADC registers read back 13 bits: 1 sign bit, two calibration bits lost to rounding and 10 data bits. Also note that in the second term, the multiplication must be performed before the division by  $2^{16}$ . Otherwise, the second term evaluates to zero incorrectly.

---



---

! **Caution:** Although the ADC can be used without the gain and offset compensation, it does exhibit non-unity gain. Designing the ADC with sub-unity gain reduces noise across the ADC range but requires the ADC results to be scaled by a factor of 8/7.

---

## ADC Control Register Definitions

The following sections define the ADC Control registers.

### ADC Control Register 0

The ADC Control Register selects the analog input channel and initiates the analog-to-digital conversion.

**Figure 20. Flash Memory Arrangement**

## Flash Information Area

The Flash information area is separate from program memory and is mapped to the address range `FE00H` to `FFFFH`. Not all these addresses are accessible. Factory trim values for the analog peripherals are stored here. Factory calibration data for the ADC is also stored here.

## Operation

The Flash Controller programs and erases Flash memory. The Flash Controller provides the proper Flash controls and timing for Byte Programming, Page Erase, and Mass Erase of Flash memory.

The Flash Controller contains several protection mechanisms to prevent accidental programming or erasure. These mechanism operate on the page, sector and full-memory levels.

Figure 21 displays a basic Flash Controller flow. The following subsections provide details about the various operations (Lock, Unlock, Byte Programming, Page Protect, Page Unprotect, Page Select Page Erase, and Mass Erase) displayed in Figure 21.

**Table 81. Flash Control Register (FCTL)**

Bit	7	6	5	4	3	2	1	0
Field	FCMD							
RESET	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W
Address	FF8H							

Bit	Description
[7:0]	<b>Flash Command</b>
FCMD	73H = First unlock command. 8CH = Second unlock command. 95H = Page Erase command (must be third command in sequence to initiate Page Erase). 63H = Mass Erase command (must be third command in sequence to initiate Mass Erase). 5EH = Enable Flash Sector Protect Register Access.



**! Caution:** The Flash Frequency High and Low Byte registers must be loaded with the correct value to ensure proper operation of the device. Also, Flash programming and erasure is not supported for system clock frequencies below 20kHz or above 20MHz.

**Table 85. Flash Frequency High Byte Register (FFREQH)**

Bit	7	6	5	4	3	2	1	0
Field	FFREQH							
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	FFAH							

Bit	Description
[7:0]	<b>Flash Frequency High Byte</b>
FFREQH	High byte of the 16-bit Flash Frequency value.

**Table 86. Flash Frequency Low Byte Register (FFREQL)**

Bit	7	6	5	4	3	2	1	0
Field	FFREQL							
RESET	0							
R/W	R/W							
Address	FFBH							

Bit	Description
[7:0]	<b>Flash Frequency Low Byte</b>
FFREQL	Low byte of the 16-bit Flash Frequency value.

Bit	Description
[7:5]	<b>Reserved</b> These bits are reserved and must be programmed to 111 during writes and to 111 when read.
[4] XTLDIS	<b>State of Crystal Oscillator at Reset</b> This bit only enables the crystal oscillator. Its selection as a system clock must be performed manually. 0 = The crystal oscillator is enabled during reset, resulting in longer reset timing. 1 = The crystal oscillator is disabled during reset, resulting in shorter reset timing.  <b>Caution:</b> Programming the XTLDIS bit to zero on 8-pin versions of F0823 Series devices prevents any further communication via the debug pin due to the X <sub>IN</sub> and DBG functions being shared on pin 2 of the 8-pin package. Do not program this bit to zero on 8-pin devices unless no further debugging or Flash programming is required.
[3:0]	<b>Reserved</b> These bits are reserved and must be programmed to 1111 during writes and to 1111 when read.

## Trim Bit Address Space

All available trim bit addresses and their functions are listed in Tables 91 through 93.

**Table 91. Trim Options Bits at Address 0000H**

Bit	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
Address	Information Page Memory 0020H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

Bit	Description
[7:0]	<b>Reserved</b> These bits are reserved. Altering this register may result in incorrect device operation.

**Table 105. Oscillator Control Register (OSCCTL)**

Bit	7	6	5	4	3	2	1	0
Field	INTEN	Reserved	WDTEN	POFEN	WDFEN	SCKSEL		
RESET	1	0	1	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F86H							

Bit	Description
[7] INTEN	<b>Internal Precision Oscillator Enable</b> 1 = Internal precision oscillator is enabled. 0 = Internal precision oscillator is disabled.
[6]	<b>Reserved</b> This bit is reserved and must be programmed to 0 during writes and to 0 when read.
[5] WDTEN	<b>Watchdog Timer Oscillator Enable</b> 1 = Watchdog Timer oscillator is enabled. 0 = Watchdog Timer oscillator is disabled.
[4] POFEN	<b>Primary Oscillator Failure Detection Enable</b> 1 = Failure detection and recovery of primary oscillator is enabled. 0 = Failure detection and recovery of primary oscillator is disabled.
[3] WDFEN	<b>Watchdog Timer Oscillator Failure Detection Enable</b> 1 = Failure detection of Watchdog Timer oscillator is enabled. 0 = Failure detection of Watchdog Timer oscillator is disabled.
[2:0] SCKSEL	<b>System Clock Oscillator Select</b> 000 = Internal precision oscillator functions as system clock at 5.53MHz. 001 = Internal precision oscillator functions as system clock at 32kHz. 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.

# eZ8 CPU Instruction Set

This chapter describes the following features of the eZ8 CPU instruction set:

Assembly Language Programming Introduction: see page 174

Assembly Language Syntax: see page 175

eZ8 CPU Instruction Notation: see page 176

eZ8 CPU Instruction Classes: see page 178

eZ8 CPU Instruction Summary: see page 182

## Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without concern for actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (opcodes and operands) to represent the instructions themselves. The opcodes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called *statements*. Each statement can contain labels, operations, operands, and comments.

Labels are assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or pseudo-ops, are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the following example.

Assembly Language Source Program Example

```
JP START      ; Everything after the semicolon is a comment.
START:        ; A label called 'START'. The first instruction (JP START) in this
              ; example causes program execution to jump to the point within the
              ; program where the START label occurs.

LD R4, R7     ; A Load (LD) instruction with two operands. The first operand,
              ; Working Register R4, is the destination. The second operand,
              ; Working Register R7, is the source. The contents of R7 is
              ; written into R4.

LD 234H, #01  ; Another Load (LD) instruction with two operands.
              ; The first operand, Extended Mode Register Address 234H,
              ; identifies the destination. The second operand, Immediate Data
              ; value 01H, is the source. The value 01H is written into the
              ; Register at address 234H.
```

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 shown in Table 106.


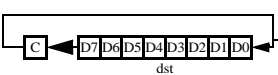
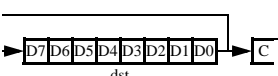
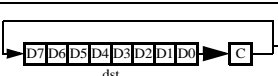
Table 106. Assembly Language Syntax Example 1

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 shown in Table 107.

Table 118. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
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
RR dst		R		E0	*	*	*	*	–	–	2	2
		IR		E1							2	3
RRC dst		R		C0	*	*	*	*	–	–	2	2
		IR		C1							2	3

Note: 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)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Upper Nibble (Hex)	0																
	1																
	2																
	3																
	4																
	5																
	6																
	7	3 ,															
	8																
	9																
	A			3.3 CPC r1,r2	3.4 CPC r1,lr2	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

## UART Timing

Figure 32 and Table 133 provide timing information for UART pins for the case where CTS is used for flow control. The CTS to DE assertion delay (T1) assumes the transmit data register has been loaded with data prior to CTS assertion.

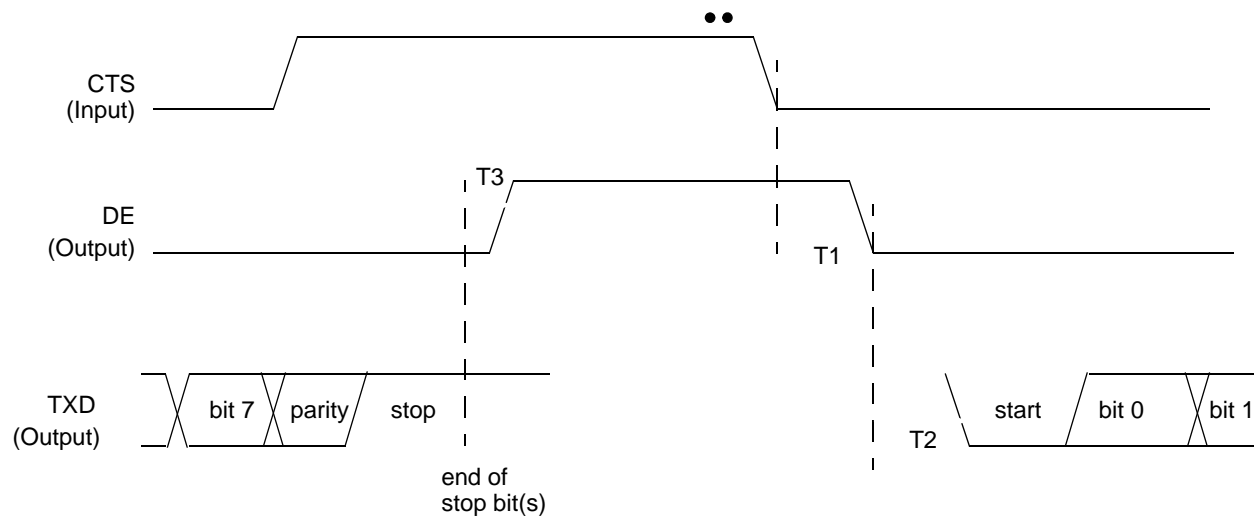


Figure 32. UART Timing With CTS

Table 133. UART Timing With CTS

Parameter	Abbreviation	Delay (ns)	
		Minimum	Maximum
UART			
T <sub>1</sub>	CTS Fall to DE output delay	2 * X <sub>IN</sub> period	2 * X <sub>IN</sub> period + 1 bit time
T <sub>2</sub>	DE assertion to TXD falling edge (start bit) delay	± 5	
T <sub>3</sub>	End of Stop Bit(s) to DE deassertion delay	± 5	



- ONE-SHOT mode 70, 88
- operating mode 70
- PWM mode 75, 76, 89
- reading the timer count values 83
- reload high and low byte registers 84
- timer control register definitions 83
- timer output signal operation 83
- timers 0-3
  - control registers 86, 87
  - high and low byte registers 83, 86
- TM 179
- TMX 179
- tools, hardware and software 220
- transmit
  - IrDA data 118
- transmitting UART data-pollled method 99
- transmitting UART dat-interrupt-driven method 100
- TRAP 181

## **U**

- UART 4
  - architecture 97
  - baud rate generator 108
  - control register definitions 108
  - controller signals 9
  - interrupts 105
  - MULTIPROCESSOR mode 103
  - receiving data using interrupt-driven method 102
  - receiving data using the polled method 101
  - transmitting data using the interrupt-driven method 100
  - transmitting data using the polled method 99
  - x baud rate high and low registers 115
  - x control 0 and control 1 registers 112
  - x status 0 and status 1 registers 110, 111
- UxBRH register 115
- UxBRL register 115
- UxCTL0 register 112, 115
- UxCTL1 register 113
- UxRXD register 109
- UxSTAT0 register 110

- UxSTAT1 register 111
- UxTXD register 109

## **V**

- vector 177
- Voltage Brownout reset (VBR) 24

## **W**

- Watchdog Timer
  - approximate time-out delay 91
  - CNTL 24
  - control register 94, 171
  - electrical characteristics and timing 202, 204
  - interrupt in normal operation 92
  - interrupt in STOP mode 92
  - refresh 92, 180
  - reload unlock sequence 93
  - reload upper, high and low registers 94
  - reset 25
  - reset in normal operation 93
  - reset in STOP mode 93
  - time-out response 92
- Watchdog Timer Control Register (WDTCTL) 94
- WDTCTL register 94, 133, 172
- WDTH register 95
- WDTL register 95
- WDTU register 95
- working register 176
- working register pair 177

## **X**

- X 177
- XOR 181
- XORX 181

## **Z**

- Z8 Encore!
  - block diagram 3
  - features 1