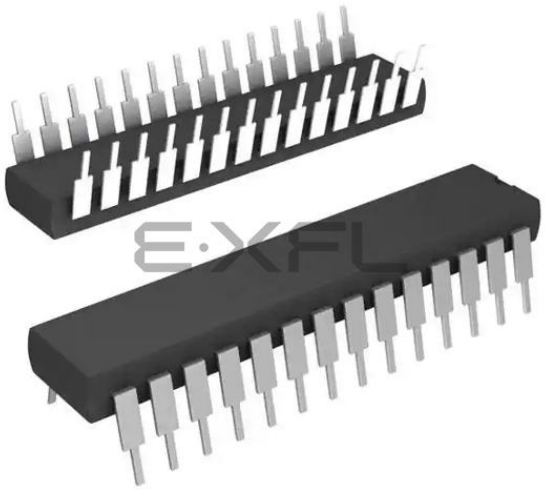

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	19
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	-
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	https://www.e-xfl.com/product-detail/zilog/z8f0412pj020eg

Table 3. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
X_{OUT}	O	**External Crystal Output** This pin is the output of the crystal oscillator. A crystal is connected between external crystal output and the X_{IN} pin to form the oscillator. When the system clock is referred in this manual, it refers to the frequency of the signal at this pin. This pin must remain unconnected when not using a crystal.
On-Chip Debugger		
DBG	I/O	**Debug** This pin is the control and data input and output to and from the OCD. The DBG pin is open-drain and must have an external pull-up resistor to ensure proper operation. **Caution:** For operation of the OCD, all power pins (V_{DD} and AV_{DD}) must be supplied with power and all ground pins (V_{SS} and AV_{SS}) must be properly grounded.
Reset		
$\overline{\text{RESET}}$	I	**RESET** Generates a Reset when asserted (driven Low).
Power Supply		
V_{DD}	I	Digital Power Supply.
AV_{DD}	I	**Analog Power Supply** Must be powered up and grounded to VDD, even if not using analog features.
V_{SS}	I	**Digital Ground.**
AV_{SS}	I	**Analog Ground** Must be grounded and connected to VSS, even if not using analog features.

Port A–C Address Registers

The Port A–C Address registers, shown in Table 14, 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 control.

Table 14. Port A–C GPIO Address Registers (P*x*ADDR)

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

Bit	Description
[7:0] PADDR	**Port Address** The Port Address selects one of the subregisters accessible through the Port Control Register. 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. 05H = Stop Mode Recovery Source Enable. 06H = Pull-up Enable. 07H–FFH = no function.

Port A–C Output Data Register

The Port A–C Output Data Register, shown in Table 23, controls the output data to the pins.

Table 23. Port A–C Output Data Register (P*x*OUT)

Bit	7	6	5	4	3	2	1	0
Field	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0							
R/W	R/W							
Address	FD3H, FD7H, FDBH							

Bit	Description
[7:0] PxOUT	**Port Output Data** These bits contain the data to be driven to the port pins. The values are only driven if the corresponding pin is configured as an output and the pin is not configured for alternate function operation. 0 = Drive a logical 0 (Low). 1 = Drive a logical 1 (High). This High value is not driven if the drain has been disabled by setting the corresponding Port Output Control Register bit to 1.

Note: x indicates register bits in the range [7:0].

- Writing a 1 to the IRQE bit in the Interrupt Control Register

Interrupts are globally disabled by any of the following actions:

- Execution of a Disable Interrupt (DI) instruction
- eZ8 CPU acknowledgement of an interrupt service request from the interrupt controller
- Writing a 0 to the IRQE bit in the Interrupt Control Register
- Reset
- Execution of a trap instruction
- Illegal instruction trap

Interrupt Vectors and Priority

The interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority, and Level 1 is the lowest priority. If all of the interrupts were enabled with identical interrupt priority (all as Level 2 interrupts), then interrupt priority would be assigned from highest to lowest as specified in Table 24. Level 3 interrupts always have higher priority than Level 2 interrupts which in turn always have higher priority than Level 1 interrupts. Within each interrupt priority level (Level 1, Level 2, or Level 3), priority is assigned as specified in Table 24. A Reset, WDT interrupt (if enabled), and an Illegal Instruction Trap will always have the highest priority.

Interrupt Assertion

Interrupt sources assert their interrupt requests for only a single system clock period (single pulse). When the interrupt request is acknowledged by the eZ8 CPU, the corresponding bit in the Interrupt Request Register is cleared until the next interrupt occurs. Writing a 0 to the corresponding bit in the Interrupt Request Register likewise clears the interrupt request.

Caution: Zilog recommends not using a coding style that clears bits in the Interrupt Request registers. All incoming interrupts received between execution of the first LDX command and the final LDX command are lost. See Example 1, which follows.

Example 1. A poor coding style that can result in lost interrupt requests:

```
LDX r0, IRQ0
AND r0, MASK
LDX IRQ0, r0
```

Observe the following procedure for configuring a timer for CONTINUOUS Mode and initiating the count:

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 starting count value 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. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin 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 calculated using 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, the ONE-SHOT Mode equation must be used 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 system clock frequency.

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 Output Signal Operation

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

Timer Control Register Definitions

This section defines the features of the following Timer Control registers.

Timer 0–1 High and Low Byte Registers: see page 64

Timer Reload High and Low Byte Registers: see page 65

Timer 0–1 PWM High and Low Byte Registers: see page 66

Timer 0–3 Control 0 Registers: see page 67

Timer 0–1 Control 1 Registers: see page 68

Timer 0–1 High and Low Byte Registers

The Timer 0–1 High and Low Byte (TxH and TxL) registers, shown in Tables 39 and 40, 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 TMRL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TMRL reads the register directly.

Zilog does not recommend writing to the Timer High and Low Byte registers while the timer is enabled. 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.

- The first bit of the byte of an address is shifting out and the RD bit of the I2C Status Register is deasserted
- The first bit of a 10-bit address shifts out
- The first bit of write data shifts out

Note: Writing to the I2C Data Register always clears the TRDE bit to 0. When TDRE is asserted, the I2C Controller pauses at the beginning of the Acknowledge cycle of the byte currently shifting out until the data register is written with the next value to send or the stop or start bits are set indicating the current byte is the last one to send.

The fourth interrupt source is the BRG. If the I2C Controller is disabled (IEN bit in the I2CCTL Register = 0) and the BIRQ bit in the I2CCTL Register = 1, an interrupt is generated when the BRG counts down to 1. This allows the I2C Baud Rate Generator to be used by software as a general purpose timer when IEN = 0.

Software Control of I2C Transactions

Software controls I2C transactions by using the I2C Controller interrupt, by polling the I2C Status Register or by DMA. Note that not all products include a DMA Controller.

To use interrupts, the I2C interrupt must be enabled in the Interrupt Controller. The TXI bit in the I2C Control Register must be set to enable transmit interrupts.

To control transactions by polling, the interrupt bits (TDRE, RDRF and NCKI) in the I2C Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

Either or both transmit and receive data movement can be controlled by the DMA Controller. The DMA Controller channel(s) must be initialized to select the I2C transmit and receive requests. Transmit DMA requests require that the TXI bit in the I2C Control Register be set.

Caution: A transmit (write) DMA operation hangs if the slave responds with a Not Acknowledge before the last byte has been sent. After receiving the Not Acknowledge, the I2C Controller sets the NCKI bit in the Status Register and pauses until either the stop or start bits in the Control Register are set.

For a receive (read) DMA transaction to send a Not Acknowledge on the last byte, the receive DMA must be set up to receive *n–1* bytes, then software must set the NAK bit and receive the last (nth) byte directly.

The NCKI interrupt does not occur in the not acknowledge case because the stop bit was set.

Write Transaction with a 7-Bit Address

Figure 27 displays the data transfer format for a 7-bit addressed slave. Shaded regions indicate data transferred from the I²C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I²C Controller.

S	Slave Address	W = 0	A	Data	A	Data	A	Data	A/A	P/S

Figure 27. 7-Bit Addressed Slave Data Transfer Format

Observe the following procedure for a transmit operation to a 7-bit addressed slave:

1. Software asserts the IEN bit in the I²C Control Register.
2. Software asserts the TXI bit of the I²C Control Register to enable transmit interrupts.
3. The I²C interrupt asserts, because the I²C Data Register is empty.
4. Software responds to the TDRE bit by writing a 7-bit Slave address plus write bit (=0) to the I²C Data Register.
5. Software asserts the start bit of the I²C Control Register.
6. The I²C Controller sends the start condition to the I²C Slave.
7. The I²C Controller loads the I²C Shift Register with the contents of the I²C Data Register.
8. After one bit of address has been shifted out by the SDA signal, the transmit interrupt is asserted (TDRE = 1).
9. Software responds by writing the transmit data into the I²C Data Register.
10. The I²C Controller shifts the rest of the address and write bit out by the SDA signal.
11. If the I²C Slave sends an acknowledge (by pulling the SDA signal Low) during the next High period of SCL the I²C Controller sets the ACK bit in the I²C Status Register. Continue to Step 12.

    If the slave does not acknowledge, the Not Acknowledge interrupt occurs (NCKI bit is set in the Status Register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I²C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remaining steps in this sequence.

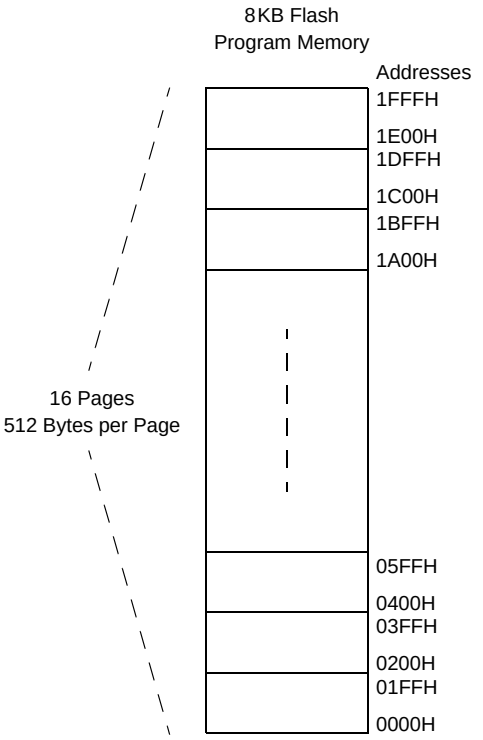

Figure 33. Flash Memory Arrangement

Information Area

Table 83 describes the Z8 Encore! XP® F0822 Series Information Area. This 512-byte Information Area is accessed by setting bit 7 of the Page Select Register to 1. When access is enabled, the Information Area is mapped into Flash memory and overlays the 512 bytes at addresses FE00H to FFFFH. When the Information Area access is enabled, LDC instructions return data from the Information Area. CPU instruction fetches always comes from Flash memory regardless of the Information Area access bit. Access to the Information Area is read-only.

Table 88. Flash Frequency High Byte Register (FFREQH)

Bit	7	6	5	4	3	2	1	0
Field	FFREQH							
RESET	0							
R/W	R/W							
Address	FFAH							

Table 89. 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] FFREQH, FFREQL	**Flash Frequency High and Low Bytes** These 2 bytes, {FFREQH[7:0], FFREQL[7:0]}, contain the 16-bit Flash Frequency value.

Operation

The following section describes the operation of the OCD.

OCD Interface

The OCD uses the DBG pin for communication with an external host. This one-pin interface is a bidirectional open-drain interface that transmits and receives data. Data transmission is half-duplex, in that transmit and receive cannot occur simultaneously. The serial data on the DBG pin is sent using the standard asynchronous data format defined in RS-232. This pin can interface the Z8 Encore! XP® F0822 Series products to the serial port of a host PC using minimal external hardware.Two different methods for connecting the DBG pin to an RS-232 interface are shown in Figures 35 and 36.

! Caution: For operation of the OCD, all power pins (V_{DD} and AV_{DD}) must be supplied with power, and all ground pins (V_{SS} and AV_{SS}) must be properly grounded. The DBG pin is open-drain and must always be connected to V_{DD} through an external pull-up resistor to ensure proper operation.

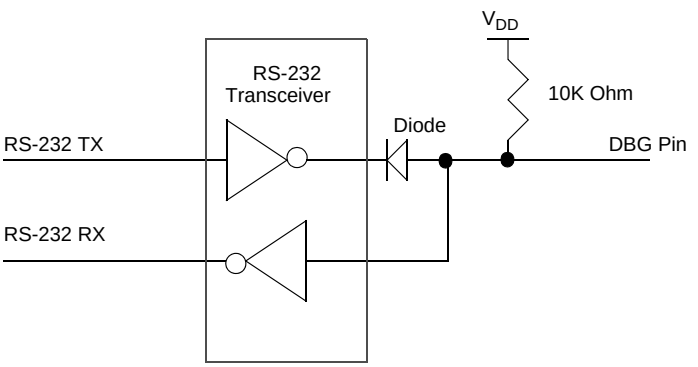

Figure 35. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #1 of 2

Table 93. On-Chip Debugger Commands (Continued)

Debug Command	Command Byte	Enabled When Not in DEBUG Mode?	Disabled by Read Protect Option Bit
Read Register	09H	–	Only reads of the peripheral control registers at address F00H–FFH are allowed.
Write Program Memory	0AH	–	Disabled
Read Program Memory	0BH	–	Disabled
Write Data Memory	0CH	–	Disabled
Read Data Memory	0DH	–	Disabled
Read Program Memory CRC	0EH	–	–
Reserved	0FH	–	–
Step Instruction	10H	–	Disabled
Stuff Instruction	11H	–	Disabled
Execute Instruction	12H	–	Disabled
Reserved	13H–FFH	–	–

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

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

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

Write OCD Counter Register (01H). The Write OCD Counter Register command writes the data that follows to the OCDCNTR Register. If the device is not in DEBUG Mode, the data is discarded.

```
DBG ← 01H
DBG ← OCDCNTR[15:8]
DBG ← OCDCNTR[7:0]
```

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

```
DBG ← 02H
DBG → OCDSTAT[7:0]
```

Read OCD Counter Register (03H). The OCD Counter Register can be used to count system clock cycles in between breakpoints, generate a BRK when it counts down to zero,

On-Chip Oscillator

Z8 Encore! XP® F0822 Series products feature an on-chip oscillator for use with external crystals with frequencies from 32kHz to 20MHz. In addition, the oscillator can support external RC networks with oscillation frequencies up to 4MHz or ceramic resonators with oscillation frequencies up to 20MHz. This oscillator generates the primary system clock for the internal eZ8 CPU and the majority of the on-chip peripherals. Alternatively, the X_{IN} input pin can also accept a CMOS-level clock input signal (32kHz–20MHz). If an external clock generator is used, the X_{OUT} pin must remain unconnected.

When configured for use with crystal oscillators or external clock drivers, the frequency of the signal on the X_{IN} input pin determines the frequency of the system clock (that is, no internal clock divider). In RC operation, the system clock is driven by a clock divider (divide by 2) to ensure 50% duty cycle.

Operating Modes

Z8 Encore! XP® F0822 Series products support 4 different oscillator modes:

- On-chip oscillator configured for use with external RC networks (<4MHz)
- Minimum power for use with very-low-frequency crystals (32kHz to 1.0MHz)
- Medium power for use with medium frequency crystals or ceramic resonators (0.5MHz to 10.0MHz)
- Maximum power for use with high-frequency crystals or ceramic resonators (8.0MHz to 20.0MHz)

The oscillator mode is selected through user-programmable option bits. For more information, see the Option Bits chapter on page 155.

Crystal Oscillator Operation

Figure 38 displays a recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 20MHz. Recommended 20MHz crystal specifications are provided in Table 96. Resistor R1 is optional and limits total power dissipation by the crystal. The printed circuit board layout must add no more than 4pF of stray capacitance to either the X_{IN} or X_{OUT} pins. If oscillation does not occur, reduce the values of capacitors C1 and C2 to decrease loading.

Table 105 lists ADC electrical characteristics and timing data.

Table 105. Analog-to-Digital Converter Electrical Characteristics and Timing

		V_{DD} = 3.0–3.6 V T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
	Resolution	10	–	–	bits	External V_{REF} = 3.0V
	Differential Nonlinearity (DNL)	–0.25	–	0.25	lsb	Guaranteed by design
	Integral Nonlinearity (INL)	–2.0	–	2.0	lsb	External V_{REF} = 3.0V
	DC Offset Error	–35	–	25	mV	80-pin QFP and 64-pin LQFP packages.
V_{REF}	Internal Reference Voltage	1.9	2.0	2.4	V	V_{DD} = 3.0–3.6V T_A = –40°C to 105°C
VC_{REF}	Voltage Coefficient of Internal Reference Voltage	–	78	–	mV/V	V_{REF} variation as a function of AV_{DD}.
TC_{REF}	Temperature Coefficient of Internal Reference Voltage	–	1	–	mV/°C	
	Single-Shot Conversion Period		5129		cycles	System clock cycles
	Continuous Conversion Period		256		cycles	System clock cycles
R_S	Analog Source Impedance	–	–	150	Ω	Recommended
Zin	Input Impedance		150		KΩ	20MHz system clock. Input impedance increases with lower system clock frequency.
V_{REF}	External Reference Voltage			AV_{DD}	V	AV_{DD} <= V_{DD}. When using an external reference voltage, decoupling capacitance should be placed from V_{REF} to AV_{SS}.
I_{REF}	Current draw into V_{REF} pin when driving with external source.		25.0	40.0	µA	

UART Timing

Figure 54 and Table 112 provide timing information for UART pins for the case where the Clear To Send input pin ($\overline{CTS}$) is used for flow control. In this example, it is assumed that the Driver Enable polarity has been configured to be Active Low and is represented here by $\overline{DE}$. The $\overline{CTS}$ to $\overline{DE}$ assertion delay (T1) assumes the UART Transmit Data Register has been loaded with data prior to $\overline{CTS}$ assertion.

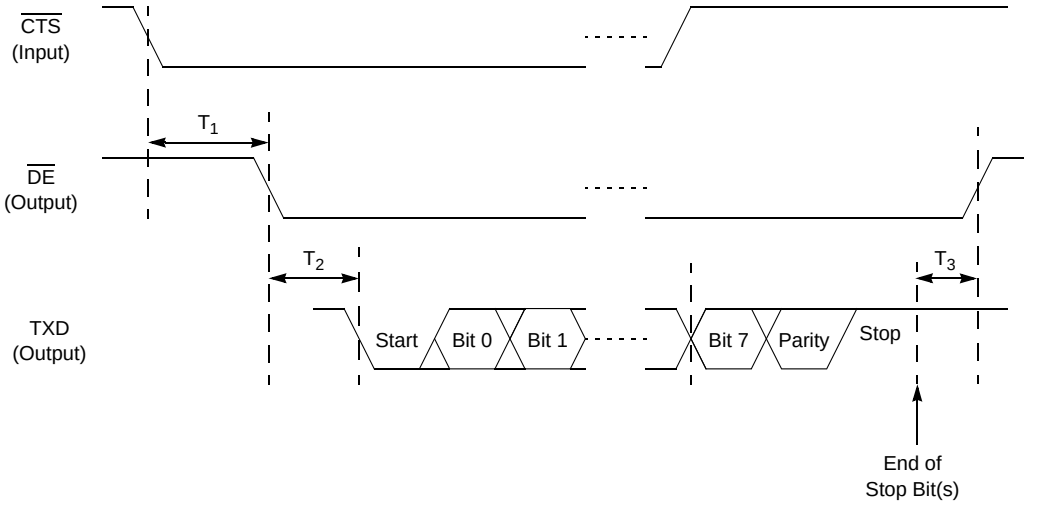

Figure 54. UART Timing with $\overline{CTS}$

Table 112. UART Timing with $\overline{CTS}$

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	$\overline{CTS}$ Fall to $\overline{DE}$ Assertion Delay	2 * X_{IN} period	2 * X_{IN} period + 1 Bit period
T_2	$\overline{DE}$ Assertion to TXD Falling Edge (Start) Delay	1 Bit period	1 Bit period + 1 * X_{IN} period
T_3	End of Stop Bit(s) to $\overline{DE}$ Deassertion Delay	1 * X_{IN} period	2 * X_{IN} period

Figure 55 and Table 113 provide timing information for UART pins for the case where the Clear To Send input signal ($\overline{CTS}$) is not used for flow control. In this example, it is assumed that the Driver Enable polarity has been configured to be Active Low and is rep-

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, 01H  ; Another Load (LD) instruction with two operands.
              ; The first operand, Extended Mode Register Address
              ; 234H, is 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 op code-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. This binary format must be followed by users that prefer manual program coding or intend to implement their 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 result is shown in Table 114.

Table 114. 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 result is shown in Table 115.

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
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
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
SBC dst, src	dst ← dst – src – C	r	r	32	*	*	*	*	1	*	2	3
		r	Ir	33							2	4
		R	R	34							3	3
		R	IR	35							3	4
		R	IM	36							3	3
		IR	IM	37							3	4
SBCX dst, src	dst ← dst – src – C	ER	ER	38	*	*	*	*	1	*	4	3
		ER	IM	39							4	3
SCF	C ← 1			DF	1	-	-	-	-	-	1	2
SRA dst		R		D0	*	*	*	0	-	-	2	2
		IR		D1							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.

R