**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, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 11 |
| Program Memory Size | 8KB (8K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 3.6V |
| Data Converters | A/D 2x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°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/z8f0821hh020eg |

Interrupt Request Control Registers . . . . . 238
GPIO Control Registers . . . . . 241
Watchdog Timer Control Registers . . . . . 244
Flash Control Registers . . . . . 246
Index . . . . . 248
Customer Support . . . . . 258

# Signal and Pin Descriptions

Z8 Encore! XP® F0822 Series products are available in a variety of packages, styles, and pin configurations. This chapter describes the signals and available pin configurations for each of the package styles. For information regarding the physical package specifications, see the Packaging chapter on page 221.

## Available Packages

Table 2 identifies the package styles available for each device within the Z8 Encore! XP® F0822 Series.

**Table 2. Z8 Encore! XP® F0822 Series Package Options**

| Part Number | 10-Bit ADC | 20-Pin SSOP and PDIP | 28-Pin SOIC and PDIP |
|---|---|---|---|
| Z8F0822 | Yes | | X |
| Z8F0821 | Yes | X | |
| Z8F0812 | No | | X |
| Z8F0811 | No | X | |
| Z8F0422 | Yes | | X |
| Z8F0421 | Yes | X | |
| Z8F0412 | No | | X |
| Z8F0411 | No | X | |

## Pin Configurations

Figures 2 through 5 display the pin configurations for all of the packages available in the Z8 Encore! XP® F0822 Series. See Table 4 on page 13 for a description of the signals.

**Note:** The analog input alternate functions (ANAx) are not available on Z8 Encore! XP® F0822 Series devices.

of Reset, all GPIO pins are configured as inputs. All GPIO programmable pull-ups are disabled.

During Reset, the eZ8 CPU and the on-chip peripherals are idle; however, the on-chip crystal oscillator and WDT oscillator continue to run. The system clock begins operating following the WDT oscillator cycle count. The eZ8 CPU and on-chip peripherals remain idle through all of the 16 cycles of the system clock.

Upon Reset, control registers within the Register File which have a defined Reset value are loaded with their reset values. Other control registers (including the Stack Pointer, Register Pointer, and Flags) and general-purpose RAM are undefined following the Reset. The eZ8 CPU fetches the Reset vector at Program memory addresses 0002H and 0003H and loads that value into the Program Counter. Program execution begins at the Reset vector address.

## Reset Sources

Table 9 lists the reset sources as a function of the operating mode. The remainder of this section provides more detail about the individual reset sources.

**Note:** A POR/VBO event always has priority over all other possible reset sources to ensure a full system reset occurs.

**Table 9. Reset Sources and Resulting Reset Type**

| Operating Mode | Reset Source | Reset Type |
|---|---|---|
| NORMAL or HALT modes | POR/VBO | System Reset |
| | WDT time-out when configured for Reset | System Reset |
| | $\overline{\text{RESET}}$ pin assertion | System Reset |
| | OCD-initiated Reset (OCDCTL[0] set to 1) | System Reset except the OCD is unaffected by the reset |
| STOP Mode | POR/ VBO | System Reset |
| | $\overline{\text{RESET}}$ pin assertion | System Reset |
| | DBG pin driven Low | System Reset |

### Power-On Reset

Each device in the Z8 Encore! XP® F0822 Series contains an internal POR circuit. The POR circuit monitors the supply voltage and holds the device in the Reset state until the supply voltage reaches a safe operating level. After the supply voltage exceeds the POR

To avoid missing interrupts, use the coding style in Example 2 to clear bits in the Interrupt Request 0 Register:

**Example 2.** A good coding style that avoids lost interrupt requests:

```
ANDX IRQ0, MASK
```

## Software Interrupt Assertion

Program code generates interrupts directly. Writing 1 to the appropriate 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:** Zilog recommends not using a coding style to generate software interrupts by setting 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 3, which follows.

**Example 3.** A poor coding style that can result in lost interrupt requests:

```
LDX r0, IRQ0
OR r0, MASK
LDX IRQ0, r0
```

To avoid missing interrupts, use the coding style in Example 4 to set bits in the Interrupt Request registers:

**Example 4.** A good coding style that avoids lost interrupt requests:

```
ORX IRQ0, MASK
```

## Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register, shown in Table 27, stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ2 Register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU reads the IRQ2 Register to determine if any interrupt requests are pending.

**Table 27. Interrupt Request 2 Register (IRQ2)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | PC3I | PC2I | PC1I | PC0I |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC6H | | | | | | | |

| Bit | Description |
|---|---|
| [7:4] | **Reserved**<br>These bits are reserved and must be programmed to 0000. |
| [3:0]<br>PCxI | **Port C Pin *x* Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port C pin *x*.<br>1 = An interrupt request from GPIO Port C pin *x* is awaiting service. |

Note: x indicates register bits in the range [3:0].

## Interrupt Control Register

The Interrupt Control (IRQCTL) Register, shown in Table 38, contains the master enable bit for all interrupts.

**Table 38. Interrupt Control Register (IRQCTL)**

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

| Bit | Description |
|---|---|
| [7]<br>IRQE | **Interrupt Request Enable**<br>This bit is set to 1 by execution of an Enable Interrupts (EI) or Interrupt Return (IRET) instruction, or by a direct register write of a 1 to this bit. It is reset to 0 by executing a DI instruction, eZ8 CPU acknowledgement of an interrupt request, Reset or by a direct register write of a 0 to this bit.<br>0 = Interrupts are disabled.<br>1 = Interrupts are enabled. |
| [6:0] | **Reserved**<br>These bits are reserved and must be programmed to 000000. |

12. The I2C Controller loads the contents of the I2C Shift Register with the contents of the I2C Data Register.
13. The I2C Controller shifts the data out of using the SDA signal. After the first bit is sent, the transmit interrupt is asserted.
14. If more bytes remain to be sent, return to Step 9.
15. Software responds by setting the stop bit of the I2C Control Register (or start bit to initiate a new transaction). In the STOP case, software clears the TXI bit of the I2C Control Register at the same time.
16. The I2C Controller completes transmission of the data on the SDA signal.
17. The slave can either Acknowledge or Not Acknowledge the last byte. Because either the stop or start bit is already set, the NCKI interrupt does not occur.
18. The I2C Controller sends the stop (or restart) condition to the I2C bus. The stop or start bit is cleared.

## Address-Only Transaction with a 10-Bit Address

In situations in which software must determine if a slave with a 10-bit address is responding without sending or receiving data, a transaction is performed which only consists of an address phase. Figure 28 displays this *address only* transaction to determine if a slave with a 10-bit address will acknowledge.

As an example, this transaction is used after a write has been executed to an EEPROM to determine when the EEPROM completes its internal write operation and is again responding to I2C transactions. If the slave does not acknowledge, the transaction is repeated until the slave is able to acknowledge.

| S | Slave Address 1st Seven Bits | W = 0 | A/A | Slave Address 2nd Byte | A/A |
|---|---|---|---|---|---|

**Figure 28. 10-Bit Address Only Transaction Format**

Observe the following procedure for an address-only transaction to a 10-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 (TDRE = 1).
4. Software responds to the TDRE interrupt by writing the first slave address byte. The least-significant bit must be 0 for the write operation.
5. Software asserts the start bit of the I2C Control Register.

The first seven bits transmitted in the first byte are 11110XX. The two XX bits are the two most-significant bits of the 10-bit address. The lowest bit of the first byte transferred is the read/write control bit (=0). The transmit operation is carried out in the same manner as 7-bit addressing.

Observe the following procedure for a transmit operation on a 10-bit addressed slave:

1. Software asserts the IEN bit in the $I^2C$ Control Register.
2. Software asserts the TXI bit of the $I^2C$ Control Register to enable transmit interrupts.
3. The $I^2C$ interrupt asserts because the $I^2C$ Data Register is empty.
4. Software responds to the TDRE interrupt by writing the first slave address byte to the $I^2C$ Data Register. The least-significant bit must be 0 for the write operation.
5. Software asserts the start bit of the $I^2C$ Control Register.
6. The $I^2C$ Controller sends the start condition to the $I^2C$ Slave.
7. The $I^2C$ Controller loads the $I^2C$ Shift Register with the contents of the $I^2C$ Data Register.
8. After one bit of address is shifted out by the SDA signal, the transmit interrupt is asserted.
9. Software responds by writing the second byte of address into the contents of the $I^2C$ Data Register.
10. The $I^2C$ Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. If the $I^2C$ Slave acknowledges the first address byte by pulling the SDA signal Low during the next High period of SCL, the $I^2C$ Controller sets the ACK bit in the $I^2C$ Status Register. Continue to Step 12.

    If the slave does not acknowledge the first address byte, the $I^2C$ Controller sets the NCKI bit and clears the ACK bit in the $I^2C$ Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The $I^2C$ Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remainder of this sequence.
12. The $I^2C$ Controller loads the $I^2C$ Shift Register with the contents of the $I^2C$ Data Register.
13. The $I^2C$ Controller shifts the second address byte out the SDA signal. After the first bit has been sent, the transmit interrupt is asserted.
14. Software responds by writing a data byte to the $I^2C$ Data Register.
15. The $I^2C$ Controller completes shifting the contents of the shift register on the SDA signal.

Observe the following procedure to setup the Flash Sector Protect Register from user code:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write 5EH to the Flash Control Register to select the Flash Sector Protect Register.
3. Read and/or write the Flash Sector Protect Register which is now at Register File address FF9H.
4. Write 00H to the Flash Control Register to return the Flash Controller to its reset state.

### Flash Write Protection Option Bit

The Flash Write Protect option bit can block all program and erase operations from user code. For more information, see the Option Bits chapter on page 155.

## Byte Programming

When the Flash Controller is unlocked, writes to Flash memory from user code programs a byte into the Flash if the address is located in the unlocked page. An erased Flash byte contains all 1s (FFH). The programming operation is used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from zero to one requires a Page Erase or Mass Erase operation.

Byte programming is accomplished using the eZ8 CPU's LDC or LDCI instructions. Refer to the eZ8 CPU Core User Manual (UM0128) for a description of the LDC and LDCI instructions.

While the Flash Controller programs the contents of Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. Interrupts that occur when a programming operation is in progress are serviced after the programming operation is complete. To exit programming mode and lock the Flash Controller, write 00H to the Flash Control Register.

User code cannot program Flash memory on a page that is located in a protected sector. When user code writes memory locations, only addresses located in the unlocked page are programmed. Memory writes outside of the unlocked page are ignored.

**Caution:** Each memory location must not be programmed more than twice before an erase occurs.

Observe the following procedure to program the Flash from user code:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write the page of memory to be programmed to the Page Select Register.
3. Write the first unlock command 73H to the Flash Control Register.

**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,

**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 zero). Reading peripheral control registers through the OCD does not effect peripheral operation. For example, register bits that are normally cleared upon a read operation will not be affected (the WDTSTAT Register is affected by the OCD Read Register operation). If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, this command returns FFH for all of 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 zero). The on-chip Flash Controller must be written to and unlocked for the programming operation to occur. If the Flash Controller is not unlocked, the data is discarded. If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the data is discarded.

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

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

```
DBG ← 0BH
DBG ← Program Memory Address[15:8]
DBG ← Program Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG ⟶ 1-65536 data bytes
```

**(Flash version only) Write Data Memory (0CH).** The Write Data Memory command writes data to Data Memory. This command is equivalent to the LDE and LDEI instructions. Data can be written 1-65536 bytes at a time (65536 bytes can be written by setting size to 0). If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the data is discarded.

```
DBG ← 0CH
DBG ← Data Memory Address[15:8]
DBG ← Data Memory Address[7:0]
DBG ← Size[15:8]
```

# 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.

## General Purpose I/O Port Input Data Sample Timing

Figure 48 displays timing of the GPIO Port input sampling. Table 106 lists the GPIO port input timing.

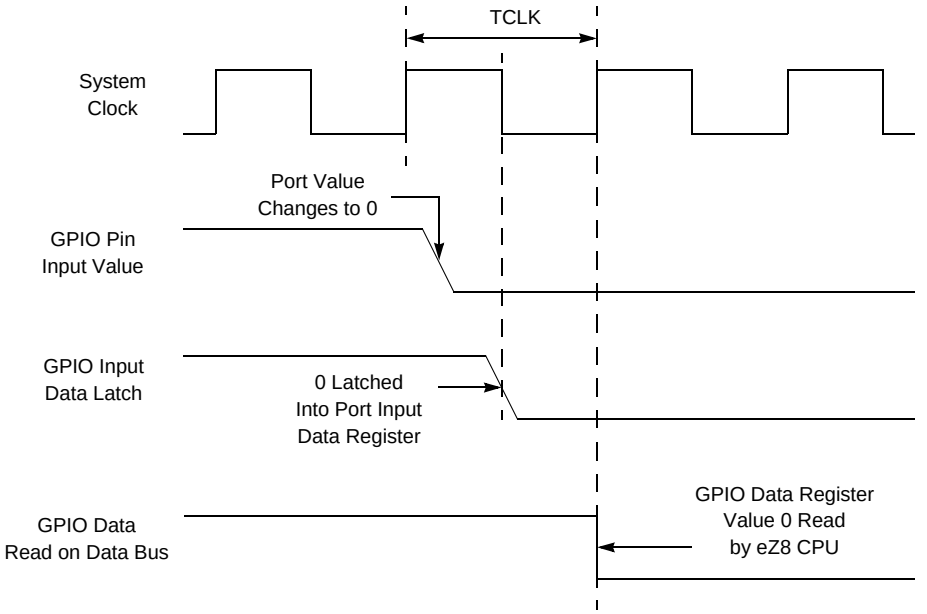


**Figure 48. Port Input Sample Timing**

**Table 106. GPIO Port Input Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_{S\_PORT}$ | Port Input Transition to $X_{IN}$ Fall Setup Time (not pictured) | 5 | – |
| $T_{H\_PORT}$ | $X_{IN}$ Fall to Port Input Transition Hold Time (not pictured) | 5 | – |
| $T_{SMR}$ | GPIO Port Pin Pulse Width to Insure Stop Mode Recovery (for GPIO port pins enabled as SMR sources) | 1µs | |

## 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.

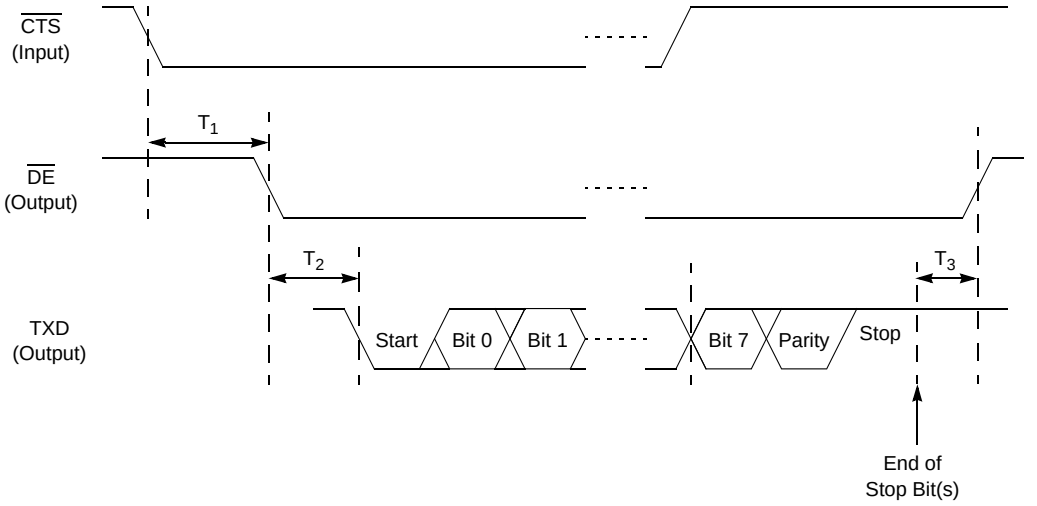


**Figure 54. UART Timing with $\overline{CTS}$**

**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-

# eZ8 CPU Instruction Classes

eZ8 CPU instructions are divided functionally into the following groups:

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Tables 119 through 126 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table as these instruction can be considered as a subset of more than one category. Within these tables, the source operand is identified as 'src', the destination operand is 'dst' and a condition code is 'cc'.

**Table 119. Arithmetic Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| ADC | dst, src | Add with Carry |
| ADCX | dst, src | Add with Carry using Extended Addressing |
| ADD | dst, src | Add |
| ADDX | dst, src | Add using Extended Addressing |
| CP | dst, src | Compare |
| CPC | dst, src | Compare with Carry |
| CPCX | dst, src | Compare with Carry using Extended Addressing |
| CPX | dst, src | Compare using Extended Addressing |
| DA | dst | Decimal Adjust |
| DEC | dst | Decrement |
| DECW | dst | Decrement Word |
| INC | dst | Increment |
| INCW | dst | Increment Word |

**Table 126. Rotate and Shift Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| BSWAP | dst | Bit Swap |
| RL | dst | Rotate Left |
| RLC | dst | Rotate Left through Carry |
| RR | dst | Rotate Right |
| RRC | dst | Rotate Right through Carry |
| SRA | dst | Shift Right Arithmetic |
| SRL | dst | Shift Right Logical |
| SWAP | dst | Swap Nibbles |

# eZ8 CPU Instruction Summary

Table 127 summarizes the eZ8 CPU instructions. The table identifies the addressing modes employed by the instruction, the effect upon the Flags Register, the number of CPU clock cycles required for the instruction fetch, and the number of CPU clock cycles required for the instruction execution.

**Table 127. eZ8 CPU Instruction Summary**

| Assembly Mnemonic | Symbolic Operation | Address Mode | | Op Code(s) (Hex) | Flags | | | | | | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | dst | src | | C | Z | S | V | D | H | | |
| ADC dst, src | dst ← dst + src + C | r | r | 12 | * | * | * | * | 0 | * | 2 | 3 |
| | | r | Ir | 13 | | | | | | | 2 | 4 |
| | | R | R | 14 | | | | | | | 3 | 3 |
| | | R | IR | 15 | | | | | | | 3 | 4 |
| | | R | IM | 16 | | | | | | | 3 | 3 |
| | | IR | IM | 17 | | | | | | | 3 | 4 |
| ADCX dst, src | dst ← dst + src + C | ER | ER | 18 | * | * | * | * | 0 | * | 4 | 3 |
| | | ER | IM | 19 | | | | | | | 4 | 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.

**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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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] ← dst[15:8] * dst[7:0] | RR | | F4 | - | - | - | - | - | - | 2 | 8 |
| NOP | No operation | | | 0F | - | - | - | - | - | - | 1 | 2 |
| 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<br>SP ← SP + 1 | R | | 50 | - | - | - | - | - | - | 2 | 2 |
| | | IR | | 51 | | | | | | | 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.

### Hex Address: F01

**Table 131. Timer 0–1 Low Byte Register (TxL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TL | | | | | | | |
| RESET | 0 | | | | | | | 1 |
| R/W | R/W | | | | | | | |
| Address | F01H, F09H | | | | | | | |

### Hex Address: F02

**Table 132. Timer 0–1 Reload High Byte Register (TxRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F02H, F0AH | | | | | | | |

### Hex Address: F03

**Table 133. Timer 0–1 Reload Low Byte Register (TxRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F03H, F0BH | | | | | | | |

### Hex Address: F0F

**Table 145. Timer 0–1 Control Registers (TxCTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | TPOL | PRES | | | TMODE | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F07H, F0FH | | | | | | | |

### Hex Addresses: F10–F3F

This address range is reserved.

## UART Control Registers

For more information about the UART registers, see the UART Control Register Definitions section on page 88.

### Hex Address: F40

**Table 146. UART Transmit Data Register (U0TXD)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TXD | | | | | | | |
| RESET | X | X | X | X | X | X | X | X |
| R/W | W | W | W | W | W | W | W | W |
| Address | F40H | | | | | | | |

**Table 147. UART Receive Data Register (U0RXD)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | RXD | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | F40H | | | | | | | |

# Customer Support

To share comments, get your technical questions answered or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.