


## 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 | 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 7x10b |
| 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/z8f0823hh005eg |

# List of Figures

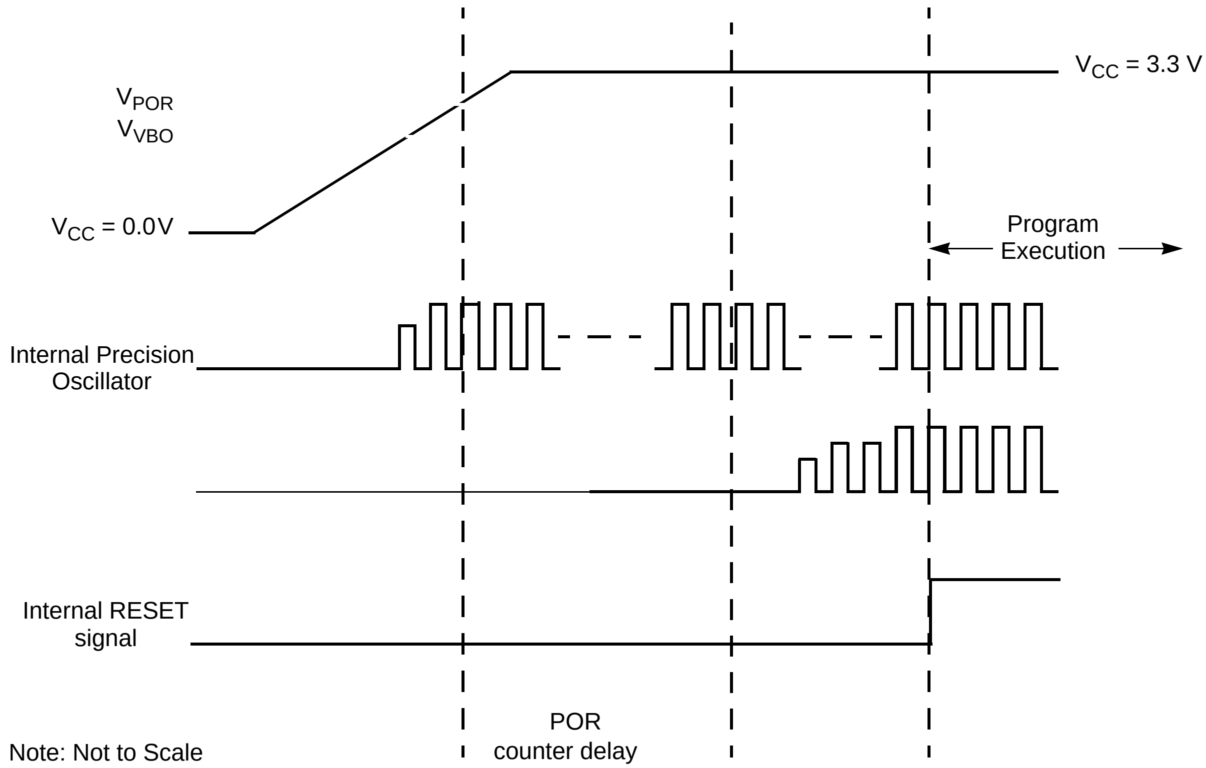



**Figure 5. Power-On Reset Operation**

## Voltage Brown-Out Reset

The devices in the Z8 Encore! XP F0823 Series provide low VBO protection. The VBO circuit senses when the supply voltage drops to an unsafe level (below the VBO threshold voltage) and forces the device into the Reset state. While the supply voltage remains below the POR voltage threshold ($V_{POR}$), the VBO block holds the device in the Reset.

After the supply voltage again exceeds the Power-On Reset voltage threshold, the device progresses through a full System Reset sequence, as described in the Power-On Reset section on page 23. Following POR, the POR status bit in the Reset Status (RSTSTAT) Register is set to 1. Figure 6 displays Voltage Brown-Out operation. For the VBO and POR threshold voltages ($V_{VBO}$ and $V_{POR}$), see the Electrical Characteristics chapter on page 196.

The VBO circuit can be either enabled or disabled during STOP Mode. Operation during STOP Mode is set by the VBO_AO Flash Option bit. For information about configuring VBO_AO, see the Flash Option Bits chapter on page 146.

PA0 and PA6 contain two different timer functions, a timer input and a complementary timer output. Both of these functions require the same GPIO configuration, the selection between the two is based on the timer mode. For more details, see the Timers chapter on page 69.

**!** **Caution:** For pins with multiple alternate functions, Zilog recommends writing to the AFS1 and AFS2 subregisters before enabling the alternate function via the AF Subregister. This prevents spurious transitions through unwanted alternate function modes.

**Table 16. Port Alternate Function Mapping (8-Pin Parts)**

| Port | Pin | Mnemonic | Alternate Function Description | Alternate Function Select Register AFS1 | Alternate Function Select Register AFS2 |
|---|---|---|---|---|---|
| **Port A** | PA0 | T0IN | Timer 0 Input | AFS1[0]: 0 | AFS2[0]: 0 |
| | | Reserved | | AFS1[0]: 0 | AFS2[0]: 1 |
| | | Reserved | | AFS1[0]: 1 | AFS2[0]: 0 |
| | | T0OUT | Timer 0 Output Complement | AFS1[0]: 1 | AFS2[0]: 1 |
| | PA1 | T0OUT | Timer 0 Output | AFS1[1]: 0 | AFS2[1]: 0 |
| | | Reserved | | AFS1[1]: 0 | AFS2[1]: 1 |
| | | CLKIN | External Clock Input | AFS1[1]: 1 | AFS2[1]: 0 |
| | | Analog Functions* | ADC Analog Input/$V_{REF}$ | AFS1[1]: 1 | AFS2[1]: 1 |
| | PA2 | DE0 | UART 0 Driver Enable | AFS1[2]: 0 | AFS2[2]: 0 |
| | | RESET | External Reset | AFS1[2]: 0 | AFS2[2]: 1 |
| | | T1OUT | Timer 1 Output | AFS1[2]: 1 | AFS2[2]: 0 |
| | | Reserved | | AFS1[2]: 1 | AFS2[2]: 1 |
| | PA3 | CTS0 | UART 0 Clear to Send | AFS1[3]: 0 | AFS2[3]: 0 |
| | | COUT | Comparator Output | AFS1[3]: 0 | AFS2[3]: 1 |
| | | T1IN | Timer 1 Input | AFS1[3]: 1 | AFS2[3]: 0 |
| | | Analog Functions* | ADC Analog Input | AFS1[3]: 1 | AFS2[3]: 1 |
| | PA4 | RXD0 | UART 0 Receive Data | AFS1[4]: 0 | AFS2[4]: 0 |
| | | Reserved | | AFS1[4]: 0 | AFS2[4]: 1 |
| | | Reserved | | AFS1[4]: 1 | AFS2[4]: 0 |
| | | Analog Functions* | ADC/Comparator Input (N) | AFS1[4]: 1 | AFS2[4]: 1 |
| | PA5 | TXD0 | UART 0 Transmit Data | AFS1[5]: 0 | AFS2[5]: 0 |
| | | T1OUT | Timer 1 Output Complement | AFS1[5]: 0 | AFS2[5]: 1 |
| | | Reserved | | AFS1[5]: 1 | AFS2[5]: 0 |
| | | Analog Functions* | ADC/Comparator Input (P) | AFS1[5]: 1 | AFS2[5]: 1 |

Note: *Analog Functions include ADC inputs, ADC reference and comparator inputs. Also, alternate function selection as described in the Port A–C Alternate Function Subregisters section on page 43 must be enabled.

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

| Port | Pin | Mnemonic | Alternate Function Description | Alternate Function Set Register AFS1 |
|---|---|---|---|---|
| **Port C**[4] | PC0 | Reserved | | AFS1[0]: 0 |
| | | ANA4/CINP | ADC or Comparator Input | AFS1[0]: 1 |
| | PC1 | Reserved | | AFS1[1]: 0 |
| | | ANA5/CINN | ADC or Comparator Input | AFS1[1]: 1 |
| | PC2 | Reserved | | AFS1[2]: 0 |
| | | ANA6/$V_{REF}$[6] | ADC Analog Input or ADC Voltage Reference | AFS1[2]: 1 |
| | PC3 | COUT | Comparator Output | AFS1[3]: 0 |
| | | Reserved | | AFS1[3]: 1 |
| | PC4 | Reserved | | AFS1[4]: 0 |
| | | | | AFS1[4]: 1 |
| | PC5 | Reserved | | AFS1[5]: 0 |
| | | | | AFS1[5]: 1 |
| | PC6 | Reserved | | AFS1[6]: 0 |
| | | | | AFS1[6]: 1 |
| | PC7 | Reserved | | AFS1[7]: 0 |
| | | | | AFS1[7]: 1 |

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. $V_{REF}$ 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. $V_{REF}$ is available on PC2 in 20-pin parts only.

# Direct LED Drive

The Port C pins provide a current sinked output capable of driving an LED without requiring an external resistor. The output sinks current at programmable levels of 3mA, 7mA, 13mA, and 20mA. This mode is enabled through the LED control registers. The LED Drive Enable (LEDEN) register turns on the drivers. The LED Drive Level (LEDLVLH and LEDLVLL) registers select the sink current.

PC[2:0]. All other signal pins are 5V-tolerant, and can safely handle inputs higher than $V_{DD}$ even with the pull-ups enabled.

## External Clock Setup

For systems using an external TTL drive, PB3 is the clock source for 20- and 28-pin devices. In this case, configure PB3 for alternate function CLKIN. Write the Oscillator Control Register (see the Oscillator Control Register Definitions section on page 171) such that the external oscillator is selected as the system clock. For 8-pin devices, use PA1 instead of PB3.

## GPIO Interrupts

Many of the GPIO port pins are used as interrupt sources. Some port pins are configured to generate an interrupt request on either the rising edge or falling edge of the pin input signal. Other port pin interrupt sources generate an interrupt when any edge occurs (both rising and falling). For more information about interrupts using the GPIO pins, see the Interrupt Controller chapter on page 54.

## GPIO Control Register Definitions

Four registers for each port provide access to GPIO control, input data, and output data. Table 18 lists these port registers. Use the Port A–D Address and Control registers together to provide access to subregisters for port configuration and control.

**Table 18. GPIO Port Registers and Subregisters**

| Port Register Mnemonic | Port Register Name |
|---|---|
| P*x*ADDR | Port A–C Address Register (Selects subregisters). |
| P*x*CTL | Port A–C Control Register (Provides access to subregisters). |
| P*x*IN | Port A–C Input Data Register. |
| P*x*OUT | Port A–C Output Data Register. |
| **Port Subregister Mnemonic** | **Port Register Name** |
| P*x*DD | Data Direction. |
| P*x*AF | Alternate Function. |
| P*x*OC | Output Control (Open-Drain). |

## Port A–C Data Direction Subregisters

The Port A–C Data Direction Subregister is accessed through the Port A–C Control Register by writing `01H` to the Port A–C Address Register; see Table 22.

**Table 22. Port A–C Data Direction Subregisters (PxDD)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Field** | DD7 | DD6 | DD5 | DD4 | DD3 | DD2 | DD1 | DD0 |
| **RESET** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **R/W** | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| **Address** | If 01H in Port A–C Address Register, accessible through the Port A–C Control Register. | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>DDx | **Data Direction**<br>These bits control the direction of the associated port pin. Port Alternate Function operation overrides the Data Direction register setting.<br>0 = Output. Data in the Port A–C Output Data Register is driven onto the port pin.<br>1 = Input. The port pin is sampled and the value written into the Port A–C Input Data Register. The output driver is tristated. |

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

## Port A–C Alternate Function Subregisters

The Port A–C Alternate Function Subregister (Table 23) is accessed through the Port A–C Control Register by writing `02H` to the Port A–C Address Register. The Port A–C Alternate Function subregisters enable the alternate function selection on pins. If disabled, pins functions as GPIO. If enabled, select one of four alternate functions using alternate function set subregisters 1 and 2 as described in the the Port A–C Alternate Function Set 1 Subregisters section on page 48 and the Port A–C Alternate Function Set 2 Subregisters section on page 49. See the GPIO Alternate Functions section on page 34 to determine the alternate function associated with each port pin.

**!** **Caution:** Do not enable alternate functions for GPIO port pins for which there is no associated alternate function. Failure to follow this guideline can result in unpredictable operation.

# Architecture

Figure 8 displays the interrupt controller block diagram.

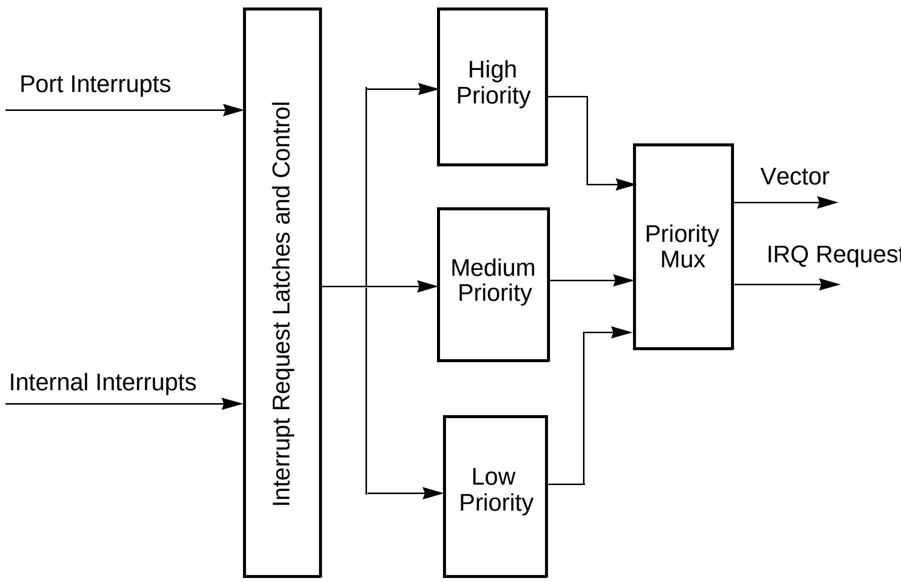



**Figure 8. Interrupt Controller Block Diagram**

# Operation

This section describes the operational aspects of the following functions.

Master Interrupt Enable: see page 56

Interrupt Vectors and Priority: see page 57

Interrupt Assertion: see page 57

Software Interrupt Assertion: see page 58

Watchdog Timer Interrupt Assertion: see page 58

## Master Interrupt Enable

The master interrupt enable bit (`IRQE`) in the Interrupt Control Register globally enables and disables interrupts.

Interrupts are globally enabled by any of the following actions:

- Execution of an Enable Interrupt (`EI`) instruction

Observe the following steps to configure a timer for PWM DUAL OUTPUT Mode and initiating the PWM operation:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for PWM DUAL OUTPUT Mode. Setting the mode also involves writing to the TMODEHI bit in the TxCTL1 Register
   - Set the prescale value
   - Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output alternate function
2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001H`); this write only affects the first pass in PWM Mode. After the first timer reset in PWM Mode, counting always begins at the reset value of `0001H`.
3. Write to the PWM High and Low Byte registers to set the PWM value.
4. Write to the PWM Control Register to set the PWM dead band delay value. The deadband delay must be less than the duration of the positive phase of the PWM signal (as defined by the PWM high and low byte registers). It must also be less than the duration of the negative phase of the PWM signal (as defined by the difference between the PWM registers and the Timer Reload registers).
5. Write to the Timer Reload High and Low Byte registers to set the reload value (PWM period). The reload value must be greater than the PWM value.
6. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. Configure the associated GPIO port pin for the Timer Output and Timer Output Complement alternate functions. The Timer Output Complement function is shared with the Timer Input function for both timers. Setting the timer mode to Dual PWM automatically switches the function from Timer In to Timer Out Complement.
8. Write to the Timer Control Register to enable the timer and initiate counting.

The PWM period is represented by the following equation:

$$\text{PWM Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

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

If `TPOL` is set to 0, the ratio of the PWM output High time to the total period is represented by:

5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control Register to enable the timer.
7. Assert the Timer Input signal to initiate the counting.

### CAPTURE/COMPARE Mode

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

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

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

Observe the following steps to configure a timer for CAPTURE/COMPARE Mode and initiating the count:

1. Write to the Timer Control Register to:
   – Disable the timer
   – Configure the timer for CAPTURE/COMPARE Mode
   – Set the prescale value
   – Set the capture edge (rising or falling) for the Timer Input
2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001H`).
3. Write to the Timer Reload High and Low Byte registers to set the compare value.
4. Enable the timer interrupt, if appropriate, and set the timer interrupt priority by writing to the relevant interrupt registers.By default, the timer interrupt are generated for both input capture and reload events. If appropriate, configure the timer interrupt to be generated only at the input capture event or the reload event by setting TICONFIG field of the TxCTL1 Register.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control Register to enable the timer.

– Set or clear the CTSE bit to enable or disable control from the remote receiver using the $\overline{\text{CTS}}$ pin

6. Check the `TDRE` bit in the UART Status 0 Register to determine if the Transmit Data Register is empty (indicated by a 1). If empty, continue to Step 7. If the Transmit Data Register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data Register becomes available to receive new data.
7. Write the UART Control 1 Register to select the outgoing address bit.
8. Set the Multiprocessor Bit Transmitter (`MPBT`) if sending an address byte, clear it if sending a data byte.
9. Write the data byte to the UART Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift register and transmits the data.
10. Make any changes to the Multiprocessor Bit Transmitter (`MPBT`) value, if appropriate and MULTIPROCESSOR Mode is enabled,.
11. To transmit additional bytes, return to Step 5.

## Transmitting Data Using the Interrupt-Driven Method

The UART Transmitter interrupt indicates the availability of the Transmit Data Register to accept new data for transmission. Observe the following steps to configure the UART for interrupt-driven data transmission:

1. Write to the UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Execute a `DI` instruction to disable interrupts.
4. Write to the Interrupt control registers to enable the UART Transmitter interrupt and set the acceptable priority.
5. Write to the UART Control 1 Register to enable MULTIPROCESSOR (9-bit) Mode functions, if MULTIPROCESSOR Mode is appropriate.
6. Set the MULTIPROCESSOR Mode Select (`MPEN`) to Enable MULTIPROCESSOR Mode.
7. Write to the UART Control 0 Register to:
   – Set the transmit enable bit (`TEN`) to enable the UART for data transmission.
   – Enable parity, if appropriate and if MULTIPROCESSOR Mode is not enabled, and select either even or odd parity.

## UART Status 0 Register

The UART Status 0 and Status 1 registers (Table 66 and Table 67) identify the current UART operating configuration and status.

**Table 66. UART Status 0 Register (U0STAT0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | RDA | PE | OE | FE | BRKD | TDRE | TXE | CTS |
| RESET | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X |
| R/W | R | R | R | R | R | R | R | R |
| Address | F41H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>RDA | **Receive Data Available**<br>This bit indicates that the UART Receive Data Register has received data. Reading the UART Receive Data Register clears this bit.<br>0 = The UART Receive Data Register is empty.<br>1 = There is a byte in the UART Receive Data Register. |
| [6]<br>PE | **Parity Error**<br>This bit indicates that a parity error has occurred. Reading the UART Receive Data register clears this bit.<br>0 = No parity error has occurred.<br>1 = A parity error has occurred. |
| [5]<br>OE | **Overrun Error**<br>This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the UART Receive Data Register has not been read. If the RDA bit is reset to 0, reading the UART Receive Data Register clears this bit.<br>0 = No overrun error occurred.<br>1 = An overrun error occurred. |
| [4]<br>FE | **Framing Error**<br>This bit indicates that a framing error (no Stop bit following data reception) was detected. Reading the UART Receive Data Register clears this bit.<br>0 = No framing error occurred.<br>1 = A framing error occurred. |
| [3]<br>BRKD | **Break Detect**<br>This bit indicates that a break occurred. If the data bits, parity/multiprocessor bit, and Stop bit(s) are all 0s this bit is set to 1. Reading the UART Receive Data Register clears this bit.<br>0 = No break occurred.<br>1 = A break occurred. |

## UART Address Compare Register

The UART Address Compare Register stores the multinode network address of the UART. When the MPMD[1] bit of UART Control Register 0 is set, all incoming address bytes are compared to the value stored in the Address Compare Register. Receive interrupts and RDA assertions only occur in the event of a match.

**Table 70. UART Address Compare Register (U0ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | COMP_ADDR | | | | | | | |
| 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 | F45H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>COMP_ADDR | **Compare Address**<br>This 8-bit value is compared to incoming address bytes. |

## UART Baud Rate High and Low Byte Registers

The UART Baud Rate High and Low Byte registers (Table 71 and Table 72) combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the UART.

**Table 71. UART Baud Rate High Byte Register (U0BRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRH | | | | | | | |
| RESET | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | F46H | | | | | | | |

**Table 72. UART Baud Rate Low Byte Register (U0BRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRL | | | | | | | |
| RESET | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | F47H | | | | | | | |

# Infrared Encoder/Decoder

Z8 Encore! XP F0823 Series products contain a fully-functional, high-performance UART with an infrared encoder/decoder (endec). The infrared endec is integrated with an on-chip UART to allow easy communication between the Z8 Encore! XP and IrDA Physical Layer Specification, Version 1.3-compliant infrared transceivers. Infrared communication provides secure, reliable, low-cost, point-to-point communication between PCs, PDAs, cell phones, printers and other infrared enabled devices.

## Architecture

Figure 16 displays the architecture of the infrared endec.

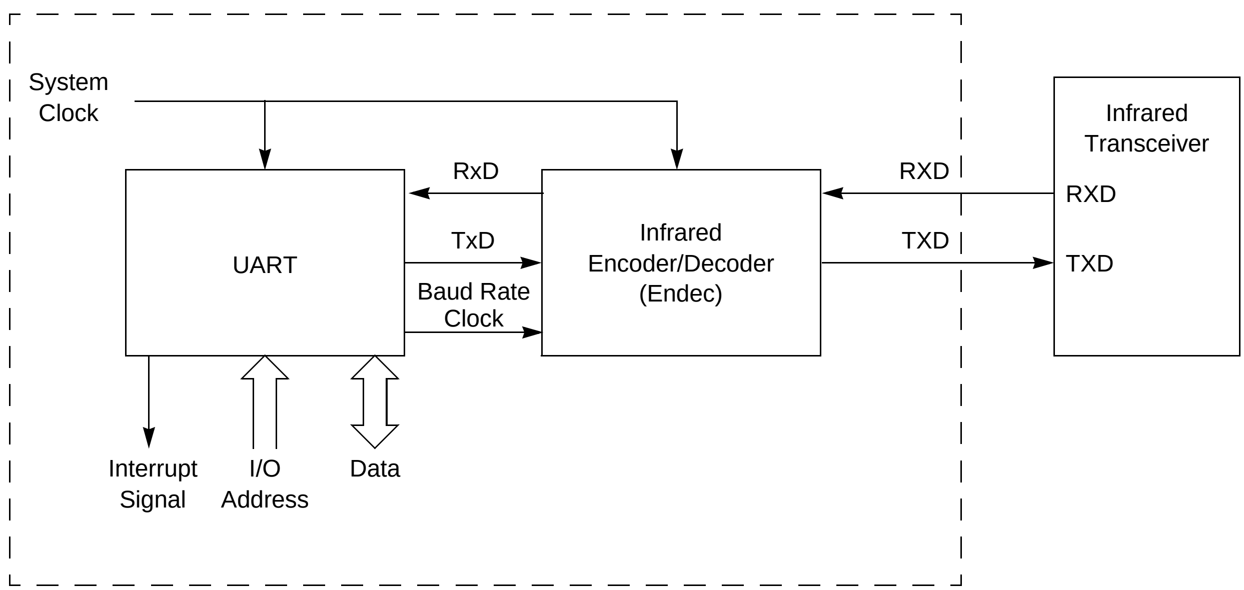



**Figure 16. Infrared Data Communication System Block Diagram**

## Operation

When the infrared endec is enabled, the transmit data from the associated on-chip UART is encoded as digital signals in accordance with the IrDA standard and output to the infrared transceiver through the TXD pin. Similarly, data received from the infrared transceiver is passed to the infrared endec through the RXD pin, decoded by the infrared endec, and

**Table 80. Flash Code Protection Using the Flash Option Bits**

| FWP | Flash Code Protection Description |
|---|---|
| 0 | Programming and erasing disabled for all of Flash Program Memory. In user code programming, Page Erase, and Mass Erase are all disabled. Mass Erase is available through the On-Chip Debugger. |
| 1 | Programming, Page Erase, and Mass Erase are enabled for all of Flash Program Memory. |

## Flash Code Protection Using the Flash Controller

At Reset, the Flash Controller locks to prevent accidental program or erasure of the Flash memory. To program or erase the Flash memory, first write the Page Select Register with the target page. Unlock the Flash Controller by making two consecutive writes to the Flash Control Register with the values `73H` and `8CH`, sequentially. The Page Select Register must be rewritten with the same page previously stored there. If the two Page Select writes do not match, the controller reverts to a locked state. If the two writes match, the selected page becomes active. For more details, see Figure 21.

After unlocking a specific page, you can enable either Page Program or Erase. Writing the value `95H` causes a Page Erase only if the active page resides in a sector that is not protected. Any other value written to the Flash Control Register locks the Flash Controller. Mass Erase is not allowed in the user code but only in through the Debug Port.

After unlocking a specific page, you can also write to any byte on that page. After a byte is written, the page remains unlocked, allowing for subsequent writes to other bytes on the same page. Further writes to the Flash Control Register cause the active page to revert to a locked state.

## Sector-Based Flash Protection

The final protection mechanism is implemented on a per-sector basis. The Flash memories of Z8 Encore! XP devices are divided into maximum number of 8 sectors. A sector is 1/8 of the total Flash memory size unless this value is smaller than the page size – in which case, the sector and page sizes are equal. On Z8 Encore! F0823 Series devices, the sector size is varied according to the Flash memory configuration shown in Table 79 on page 134.

The Flash Sector Protect Register can be configured to prevent sectors from being programmed or erased. After a sector is protected, it cannot be unprotected by user code. The Flash Sector Protect Register is cleared after reset, and any previously-written protection values are lost. User code must write this register in their initialization routine if they prefer to enable sector protection.

The Flash Sector Protect Register shares its Register File address with the Page Select Register. The Flash Sector Protect Register is accessed by writing the Flash Control Register with `5EH`. After the Flash Sector Protect Register is selected, it can be accessed at the Page Select Register address. When user code writes the Flash Sector Protect Register,

**Table 83. Flash Page Select Register (FPS)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Field** | INFO_EN | PAGE | | | | | | |
| **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** | FF9H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>INFO_EN | **Information Area Enable**<br>0 = Information Area us not selected.<br>1 = Information Area is selected. The Information Area is mapped into the Program Memory address space at addresses FE00H through FFFFH. |
| [6:0]<br>PAGE | **Page Select**<br>This 7-bit field identifies the Flash memory page for Page Erase and page unlocking.<br>• Program Memory Address[15:9] = PAGE[6:0].<br>• For Z8F04x3 devices, the upper 4 bits must always be 0.<br>• For Z8F02x3 devices, the upper 5 bits must always be 0.<br>• For Z8F01x3 devices, the upper 6 bits must always be 0. |

### 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<br>; example causes program execution to jump to the point within the<br>; program where the `START` label occurs. |
| `LD R4, R7` | ; A Load (LD) instruction with two operands. The first operand,<br>; Working Register R4, is the destination. The second operand,<br>; Working Register R7, is the source. The contents of R7 is<br>; written into R4. |
| `LD 234H, #%01` | ; Another Load (LD) instruction with two operands.<br>; The first operand, Extended Mode Register Address `234H`,<br>; identifies the destination. The second operand, Immediate Data<br>; value `01H`, is the source. The value `01H` is written into the<br>; 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 113. CPU Control Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| ATM | — | Atomic Execution |
| CCF | — | Complement Carry Flag |
| DI | — | Disable Interrupts |
| EI | — | Enable Interrupts |
| HALT | — | HALT Mode |
| NOP | — | No Operation |
| RCF | — | Reset Carry Flag |
| SCF | — | Set Carry Flag |
| SRP | src | Set Register Pointer |
| STOP | — | STOP Mode |
| WDT | — | Watchdog Timer Refresh |

**Table 114. Load Instructions**

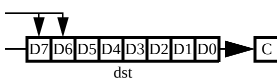
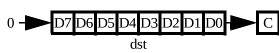
| Mnemonic | Operands | Instruction |
|---|---|---|
| CLR | dst | Clear |
| LD | dst, src | Load |
| LDC | dst, src | Load Constant to/from Program Memory |
| LDCI | dst, src | Load Constant to/from Program Memory and Auto-Increment Addresses |
| LDE | dst, src | Load External Data to/from Data Memory |
| LDEI | dst, src | Load External Data to/from Data Memory and Auto-Increment Addresses |
| LDWX | dst, src | Load Word using Extended Addressing |
| LDX | dst, src | Load using Extended Addressing |
| LEA | dst, X(src) | Load Effective Address |
| POP | dst | Pop |
| POPX | dst | Pop using Extended Addressing |
| PUSH | src | Push |
| PUSHX | src | Push using Extended Addressing |

**Table 118. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LD dst, rc | dst ← src | r | IM | 0C-FC | – | – | – | – | – | – | 2 | 2 |
| | | r | X(r) | C7 | | | | | | | 3 | 3 |
| | | X(r) | r | D7 | | | | | | | 3 | 4 |
| | | r | Ir | E3 | | | | | | | 2 | 3 |
| | | R | R | E4 | | | | | | | 3 | 2 |
| | | R | IR | E5 | | | | | | | 3 | 4 |
| | | R | IM | E6 | | | | | | | 3 | 2 |
| | | IR | IM | E7 | | | | | | | 3 | 3 |
| | | Ir | r | F3 | | | | | | | 2 | 3 |
| | | IR | R | F5 | | | | | | | 3 | 3 |
| 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 |

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 118. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Flags Z | Flags S | Flags V | Flags D | Flags H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| SRL dst |  | R | | 1F C0 | * | * | 0 | * | – | – | 3 | 2 |
| | | IR | | 1F C1 | | | | | | | 3 | 3 |
| SRP src | RP ← src | | IM | 01 | – | – | – | – | – | – | 2 | 2 |
| STOP | STOP Mode | | | 6F | – | – | – | – | – | – | 1 | 2 |
| SUB dst, src | dst ← dst – src | r | r | 22 | * | * | * | * | 1 | * | 2 | 3 |
| | | r | Ir | 23 | | | | | | | 2 | 4 |
| | | R | R | 24 | | | | | | | 3 | 3 |
| | | R | IR | 25 | | | | | | | 3 | 4 |
| | | R | IM | 26 | | | | | | | 3 | 3 |
| | | IR | IM | 27 | | | | | | | 3 | 4 |
| 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 |

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 135. Z8 Encore! XP F0823 Series Ordering Matrix (Continued)**

| Part Number | Flash | RAM | I/O Lines | Interrupts | 16-Bit Timers w/PWM | 10-Bit A/D Channels | UART with IrDA | Description |
|---|---|---|---|---|---|---|---|---|
| **Z8 Encore! XP F0823 Series with 2 KB Flash** | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | |
| Z8F0213PB005SG | 2 KB | 512 B | 6 | 12 | 2 | 0 | 1 | PDIP 8-pin package |
| Z8F0213QB005SG | 2 KB | 512 B | 6 | 12 | 2 | 0 | 1 | QFN 8-pin package |
| Z8F0213SB005SG | 2 KB | 512 B | 6 | 12 | 2 | 0 | 1 | SOIC 8-pin package |
| Z8F0213SH005SG | 2 KB | 512 B | 16 | 18 | 2 | 0 | 1 | SOIC 20-pin package |
| Z8F0213HH005SG | 2 KB | 512 B | 16 | 18 | 2 | 0 | 1 | SSOP 20-pin package |
| Z8F0213PH005SG | 2 KB | 512 B | 16 | 18 | 2 | 0 | 1 | PDIP 20-pin package |
| Z8F0213SJ005SG | 2 KB | 512 B | 24 | 18 | 2 | 0 | 1 | SOIC 28-pin package |
| Z8F0213HJ005SG | 2 KB | 512 B | 24 | 18 | 2 | 0 | 1 | SSOP 28-pin package |
| Z8F0213PJ005SG | 2 KB | 512 B | 24 | 18 | 2 | 0 | 1 | PDIP 28-pin package |
| **Extended Temperature: –40°C to 105°C** | | | | | | | | |
| Z8F0213PB005EG | 2 KB | 512 B | 6 | 12 | 2 | 0 | 1 | PDIP 8-pin package |
| Z8F0213QB005EG | 2 KB | 512 B | 6 | 12 | 2 | 0 | 1 | QFN 8-pin package |
| Z8F0213SB005EG | 2 KB | 512 B | 6 | 12 | 2 | 0 | 1 | SOIC 8-pin package |
| Z8F0213SH005EG | 2 KB | 512 B | 16 | 18 | 2 | 0 | 1 | SOIC 20-pin package |
| Z8F0213HH005EG | 2 KB | 512 B | 16 | 18 | 2 | 0 | 1 | SSOP 20-pin package |
| Z8F0213PH005EG | 2 KB | 512 B | 16 | 18 | 2 | 0 | 1 | PDIP 20-pin package |
| Z8F0213SJ005EG | 2 KB | 512 B | 24 | 18 | 2 | 0 | 1 | SOIC 28-pin package |
| Z8F0213HJ005EG | 2 KB | 512 B | 24 | 18 | 2 | 0 | 1 | SSOP 28-pin package |
| Z8F0213PJ005EG | 2 KB | 512 B | 24 | 18 | 2 | 0 | 1 | PDIP 28-pin package |