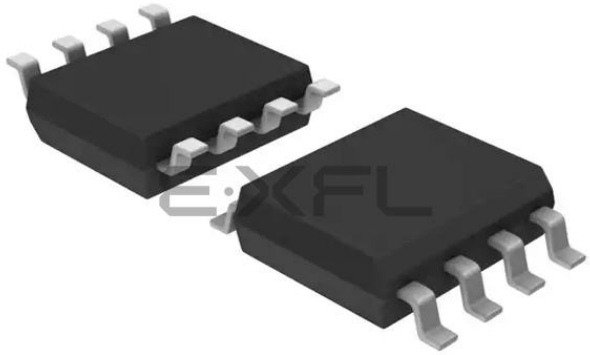


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

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

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | eZ8 |
| Core Size | 8-Bit |
| Speed | 5MHz |
| Connectivity | IrDA, UART/USART |
| Peripherals | Brown-out Detect/Reset, LED, POR, PWM, WDT |
| Number of I/O | 6 |
| 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 | 0°C ~ 70°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 8-SOIC (0.154", 3.90mm Width) |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/zilog/z8f0413sb005sc |

# Low-Power Modes

Z8 Encore! XP® F0823 Series products contain power-saving features. The highest level of power reduction is provided by the STOP mode, in which nearly all device functions are powered down. The next lower level of power reduction is provided by the HALT mode, in which the CPU is powered down.

Further power savings can be implemented by disabling individual peripheral blocks while in ACTIVE mode (defined as being in neither STOP nor HALT mode).

## STOP Mode

Executing the eZ8 CPU's Stop instruction places the device into STOP mode, powering down all peripherals except the Voltage Brownout detector, and the Watchdog Timer. These two blocks may also be disabled for additional power savings. In STOP mode, the operating characteristics are:

- Primary crystal oscillator and internal precision oscillator are stopped; XIN and XOUT (if previously enabled) are disabled, and PA0/PA1 revert to the states programmed by the GPIO registers.
- System clock is stopped.
- eZ8 CPU is stopped.
- Program counter (PC) stops incrementing.
- Watchdog Timer's internal RC oscillator continues to operate if enabled by the Oscillator Control Register.
- If enabled, the Watchdog Timer logic continues to operate.
- If enabled for operation in STOP mode by the associated Flash Option Bit, the Voltage Brownout protection circuit continues to operate.
- All other on-chip peripherals are idle.

To minimize current in STOP mode, all GPIO pins that are configured as digital inputs must be driven to one of the supply rails ($V_{CC}$ or GND). Additionally, any GPIOs configured as outputs must also be driven to one of the supply rails. The device can be brought out of STOP mode using Stop Mode Recovery. For more information on Stop Mode Recovery, see Reset and Stop Mode Recovery on page 21.

**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 |
| | | $\overline{\text{T0OUT}}$ | 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/VREF | AFS1[1]: 1 | AFS2[1]: 1 |
| | PA2 | DE0 | UART 0 Driver Enable | AFS1[2]: 0 | AFS2[2]: 0 |
| | | $\overline{\text{RESET}}$ | 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 | $\overline{\text{CTS0}}$ | 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 |
| | | $\overline{\text{T1OUT}}$ | 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 Port A–C Alternate Function Sub-Registers must be enabled.

## Port A–C Address Registers

The Port A–C Address registers select the GPIO Port functionality accessible through the Port A–C Control registers. The Port A–C Address and Control registers combine to provide access to all GPIO Port controls (Table 18).

**Table 18. Port A–C GPIO Address Registers (P*x*ADDR)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PADDR[7:0] | | | | | | | |
| RESET | 00H | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FD0H, FD4H, FD8H | | | | | | | |

PADDR[7:0]—Port Address
The Port Address selects one of the sub-registers accessible through the Port Control register.

| PADDR[7:0] | Port Control Sub-register Accessible Using the Port A–C Control Registers |
|---|---|
| 00H | No function. Provides some protection against accidental Port reconfiguration. |
| 01H | Data Direction. |
| 02H | Alternate Function. |
| 03H | Output Control (Open-Drain). |
| 04H | High Drive Enable. |
| 05H | Stop Mode Recovery Source Enable. |
| 06H | Pull-up Enable. |
| 07H | Alternate Function Set 1. |
| 08H | Alternate Function Set 2. |
| 09H–FFH | No function. |

## Port A–C Control Registers

The Port A–C Control registers set the GPIO port operation. The value in the corresponding Port A–C Address register determines which sub-register is read from or written to by a Port A–C Control register transaction (Table 19).

**Caution:** *To avoid missing interrupts, use the following coding style to clear bits in the Interrupt Request 0 register:*

**Good coding style that avoids lost interrupt requests:**

```
ANDX IRQ0, MASK
```

## Software Interrupt Assertion

Program code generates interrupts directly. Writing a 1 to the correct bit in the Interrupt Request register triggers an interrupt (assuming that interrupt is enabled). When the interrupt request is acknowledged by the eZ8 CPU, the bit in the Interrupt Request register is automatically cleared to 0.

**Caution:** *The following coding style used to generate software interrupts by setting bits in the Interrupt Request registers is not recommended. All incoming interrupts received between execution of the first LDX command and the final LDX command are lost.*

**Poor coding style that can result in lost interrupt requests:**

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

**Caution:** *To avoid missing interrupts, use the following coding style to set bits in the Interrupt Request registers:*

**Good coding style that avoids lost interrupt requests:**

```
ORX IRQ0, MASK
```

## Watchdog Timer Interrupt Assertion

The Watchdog Timer interrupt behavior is different from interrupts generated by other sources. The Watchdog Timer continues to assert an interrupt as long as the timeout condition continues. As it operates on a different (and usually slower) clock domain than the rest of the device, the Watchdog Timer continues to assert this interrupt for many system clocks until the counter rolls over.

**Caution:** *To avoid re-triggerings of the Watchdog Timer interrupt after exiting the associated interrupt service routine, it is recommended that the service routine continues to read from the RSTSTAT register until the* `WDT` *bit is cleared as given in the following coding sample:*

```
CLEARWDT:
LDX r0, RSTSTAT ; read reset status register to clear wdt
   bit
BTJNZ 5, r0, CLEARWDT ; loop until bit is cleared
```

## Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) register (Table 35) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ1 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 Interrupt Request 1 register to determine if any interrupt requests are pending.

**Table 35. Interrupt Request 1 Register (IRQ1)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PA7VI | PA6CI | PA5I | PA4I | PA3I | PA2I | PA1I | PA0I |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FC3H | | | | | | | |

PA7VI—Port A7 Interrupt Request
0 = No interrupt request is pending for GPIO Port A
1 = An interrupt request from GPIO Port A

PA6CI—Port A6 or Comparator Interrupt Request
0 = No interrupt request is pending for GPIO Port A or Comparator
1 = An interrupt request from GPIO Port A or Comparator

PA*x*I—Port A Pin *x* Interrupt Request
0 = No interrupt request is pending for GPIO Port A pin *x*
1 = An interrupt request from GPIO Port A pin *x* is awaiting service

where *x* indicates the specific GPIO Port pin number (0–5)

## Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) register (Table 36) 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 can read the Interrupt Request 2 register to determine if any interrupt requests are pending.

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. Follow the steps below 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.
   - Set or clear CTSE to enable or disable control from the remote receiver using the $\overline{\text{CTS}}$ pin.
8. Execute an EI instruction to enable interrupts.

3. Clears the UART Receiver interrupt in the applicable Interrupt Request register.
4. Executes the IRET instruction to return from the interrupt-service routine and await more data.

## Clear To Send ($\overline{\text{CTS}}$) Operation

The CTS pin, if enabled by the CTSE bit of the UART Control 0 register, performs flow control on the outgoing transmit datastream. The Clear To Send ($\overline{\text{CTS}}$) input pin is sampled one system clock before beginning any new character transmission. To delay transmission of the next data character, an external receiver must deassert $\overline{\text{CTS}}$ at least one system clock cycle before a new data transmission begins. For multiple character transmissions, this action is typically performed during Stop Bit transmission. If $\overline{\text{CTS}}$ deasserts in the middle of a character transmission, the current character is sent completely.

## MULTIPROCESSOR (9-Bit) Mode

The UART has a MULTIPROCESSOR (9-bit) mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In MULTIPROCESSOR mode (also referred to as 9-bit mode), the multiprocessor bit (MP) is transmitted immediately following the 8-bits of data and immediately preceding the Stop bit(s) as displayed in Figure 13. The character format is given below:

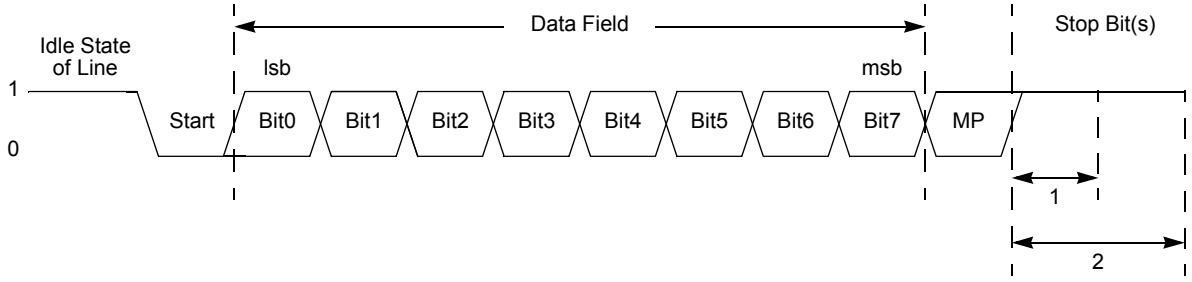



**Figure 13. UART Asynchronous MULTIPROCESSOR Mode Data Format**

In MULTIPROCESSOR (9-bit) mode, the Parity bit location (9th bit) becomes the Multiprocessor control bit. The UART Control 1 and Status 1 registers provide MULTIPROCESSOR (9-bit) mode control and status information. If an automatic address matching scheme is enabled, the UART Address Compare register holds the network address of the device.

### MULTIPROCESSOR (9-bit) Mode Receive Interrupts

When MULTIPROCESSOR mode is enabled, the UART only processes frames addressed to it. The determination of whether a frame of data is addressed to the UART can be made

## External Driver Enable

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

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

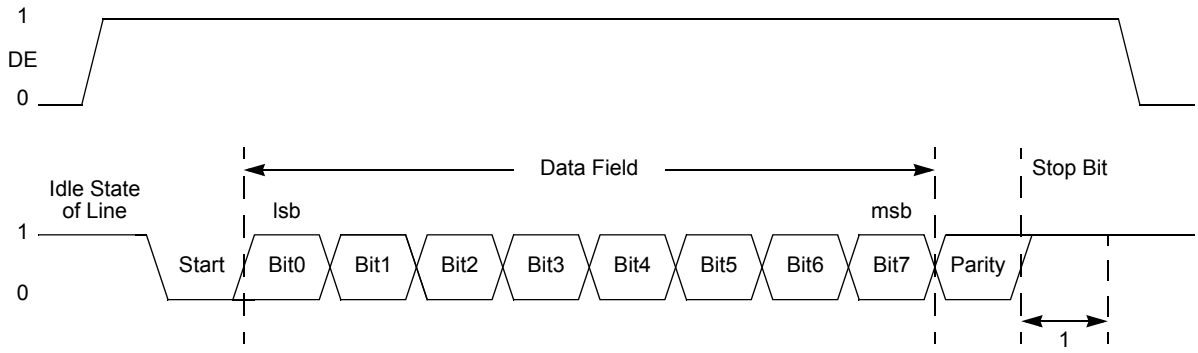



**Figure 14. UART Driver Enable Signal Timing (shown with 1 Stop Bit and Parity)**

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

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

## UART Interrupts

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

### Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register Empty bit (TDRE) is set to 1. This indicates that the transmitter is ready to accept new data for transmission. The TDRE interrupt occurs after the Transmit shift register has shifted the first bit of data out. The Transmit Data register can now be written with the next character to send. This action provides 7 bit periods of latency to load the Transmit Data register before the Transmit shift register completes shifting the current character. Writing to the UART Transmit Data register clears the TDRE bit to 0.

### Receiver Interrupts

The receiver generates an interrupt when any of the following occurs:

- A data byte is received and is available in the UART Receive Data register. This interrupt can be disabled independently of the other receiver interrupt sources. The received data interrupt occurs after the receive character has been received and placed in the Receive Data register. To avoid an overrun error, software must respond to this received data available condition before the next character is completely received.

**Note:** *In MULTIPROCESSOR mode (MPEN = 1), the receive data interrupts are dependent on the multiprocessor configuration and the most recent address byte.*

- A break is received
- An overrun is detected
- A data framing error is detected

### UART Overrun Errors

When an overrun error condition occurs the UART prevents overwriting of the valid data currently in the Receive Data register. The Break Detect and Overrun status bits are not displayed until after the valid data has been read.

After the valid data has been read, the UART Status 0 register is updated to indicate the overrun condition (and Break Detect, if applicable). The RDA bit is set to 1 to indicate that the Receive Data register contains a data byte. However, because the overrun error occurred, this byte cannot contain valid data and must be ignored. The BRKD bit indicates if the overrun was caused by a break condition on the line. After reading the status byte indicating an overrun error, the Receive Data register must be read again to clear the error bits is the UART Status 0 register. Updates to the Receive Data register occur only when the next data word is received.

### UART Data and Error Handling Procedure

Figure 15 displays the recommended procedure for use in UART receiver interrupt service routines.

Endec, and passed to the UART. Communication is half-duplex, which means simultaneous data transmission and reception is not allowed.

The baud rate is set by the UART's baud rate generator and supports IrDA standard baud rates from 9600 baud to 115.2 kbaud. Higher baud rates are possible, but do not meet IrDA specifications. The UART must be enabled to use the Infrared Endec. The Infrared Endec data rate is calculated using the following equation:

$$\text{Infrared Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Baud Rate Divisor Value}}$$

## Transmitting IrDA Data

The data to be transmitted using the infrared transceiver is first sent to the UART. The UART's transmit signal (TXD) and baud rate clock are used by the IrDA to generate the modulation signal (IR_TXD) that drives the infrared transceiver. Each UART/Infrared data bit is 16 clocks wide. If the data to be transmitted is 1, the IR_TXD signal remains low for the full 16 clock period. If the data to be transmitted is 0, the transmitter first outputs a 7 clock low period, followed by a 3 clock high pulse. Finally, a 6 clock low pulse is output to complete the full 16 clock data period. Figure 17 displays IrDA data transmission. When the Infrared Endec is enabled, the UART's TXD signal is internal to Z8 Encore! XP® F0823 Series products while the IR_TXD signal is output through the TXD pin.

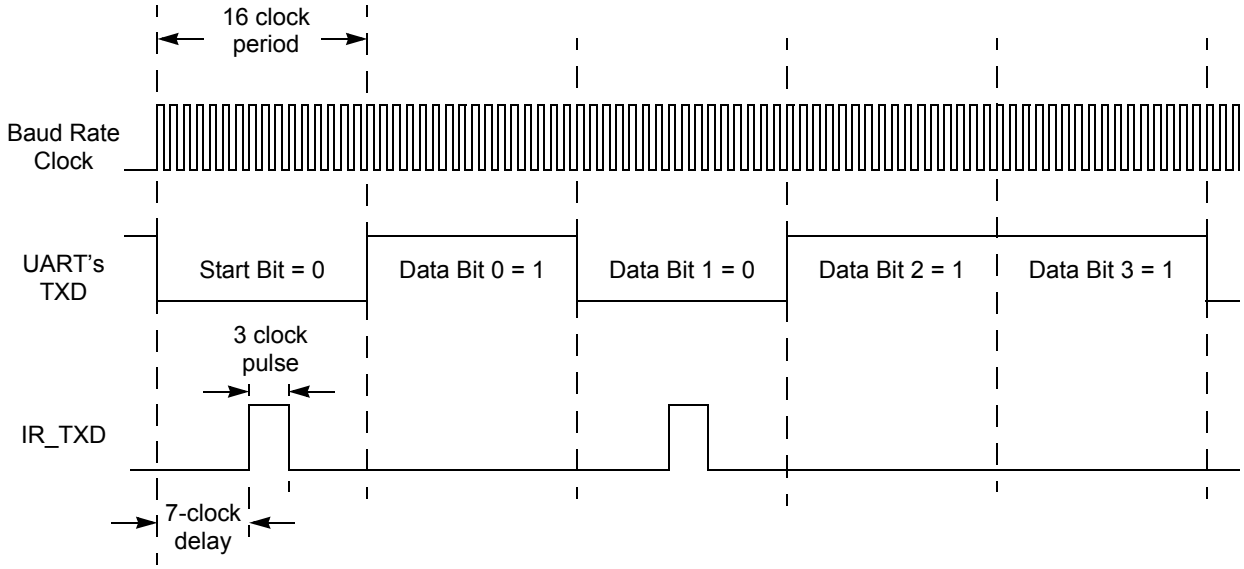



**Figure 17. Infrared Data Transmission**

configurations. The information contained here is lost when page 0 of the Program Memory is erased.

### Trim Option Bits

The trim option bits are contained in the information page of the Flash memory. These bits are factory programmed values required to optimize the operation of onboard analog circuitry and cannot be permanently altered. Program Memory may be erased without endangering these values. It is possible to alter working values of these bits by accessing the Trim Bit Address and Data Registers, but these working values are lost after a power loss or any other reset event.

There are 32 bytes of trim data. To modify one of these values the user code must first write a value between 00H and 1FH into the Trim Bit Address Register. The next write to the Trim Bit Data register changes the working value of the target trim data byte.

Reading the trim data requires the user code to write a value between 00H and 1FH into the Trim Bit Address Register. The next read from the Trim Bit Data register returns the working value of the target trim data byte.

**Note:** *The trim address range is from information address 20-3F only. The remainder of the information page is not accessible through the trim bit address and data registers.*

### Calibration Option Bits

The calibration option bits are also contained in the information page. These bits are factory programmed values intended for use in software correcting the device's analog performance. To read these values, the user code must employ the LDC instruction to access the information area of the address space as defined in Flash Information Area on page 15

### Serialization Bits

As an optional feature, Zilog® is able to provide factory-programmed serialization. For serialized products, the individual devices are programmed with unique serial numbers. These serial numbers are binary values, four bytes in length. The numbers increase in size with each device, but gaps in the serial sequence may exist.

These serial numbers are stored in the Flash information page (for more details, see Reading the Flash Information Page on page 143 and Serialization Data on page 148) and are unaffected by mass erasure of the device's Flash memory.

### Randomized Lot Identification Bits

As an optional feature, Zilog is able to provide a factory-programmed random lot identifier. With this feature, all devices in a given production lot are programmed with the same random number. This random number is uniquely regenerated for each successive production lot and is not likely to be repeated.

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

### Trim Bit Address 0002H

**Table 91. Trim Option Bits at 0002H (TIPO)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | IPO_TRIM | | | | | | | |
| RESET | U | | | | | | | |
| R/W | R/W | | | | | | | |
| ADDR | Information Page Memory 0022H | | | | | | | |

**Note:** U = Unchanged by Reset. R/W = Read/Write.

IPO_TRIM—Internal Precision Oscillator Trim Byte
Contains trimming bits for Internal Precision Oscillator.

### Trim Bit Address 0003H—Reserved

### Trim Bit Address 0004H—Reserved

## Zilog Calibration Data

### ADC Calibration Data

**Table 92. ADC Calibration Bits**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | ADC_CAL | | | | | | | |
| RESET | U | U | U | U | U | U | U | U |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | Information Page Memory 0060H–007DH | | | | | | | |

**Note:** U = Unchanged by Reset. R/W = Read/Write.

ADC_CAL—Analog-to-Digital Converter Calibration Values
Contains factory calibrated values for ADC gain and offset compensation. Each of the ten supported modes has one byte of offset calibration and two bytes of gain calibration. These values are read by the software to compensate ADC measurements as detailed in

## OCD Unlock Sequence (8-Pin Devices Only)

Because of pin-sharing on the 8-pin device, an unlock sequence must be performed to access the DBG pin. If this sequence is not completed during a system reset, then the PA0/DBG pin functions only as a GPIO pin.

The following sequence unlocks the DBG pin:

1. Hold PA2/$\overline{\text{RESET}}$ Low.
2. Wait 5 ms for the internal reset sequence to complete.
3. Send the following bytes serially to the debug pin:

```
DBG ← 80H (autobaud)
DBG ← EBH
DBG ← 5AH
DBG ← 70H
DBG ← CDH (32-bit unlock key)
```

4. Release PA2/$\overline{\text{RESET}}$. The PA0/DBG pin is now identical in function to that of the DBG pin on the 20- or 28-pin device. To enter DEBUG mode, re-autobaud and write 80H to the OCD control register (see On-Chip Debugger Commands on page 157).

## Breakpoints

Execution breakpoints are generated using the BRK instruction (opcode 00H). When the eZ8 CPU decodes a BRK instruction, it signals the OCD. If breakpoints are enabled, the OCD enters DEBUG mode and idles the eZ8 CPU. If breakpoints are not enabled, the OCD ignores the BRK signal and the BRK instruction operates as an NOP instruction.

### Breakpoints in Flash Memory

The BRK instruction is opcode 00H, which corresponds to the fully programmed state of a byte in Flash memory. To implement a breakpoint, write 00H to the required break address, overwriting the current instruction. To remove a breakpoint, the corresponding page of Flash memory must be erased and reprogrammed with the original data.

## Runtime Counter

The OCD contains a 16-bit Runtime Counter. It counts system clock cycles between breakpoints. The counter starts counting when the OCD leaves DEBUG mode and stops counting when it enters DEBUG mode again or when it reaches the maximum count of FFFFH.

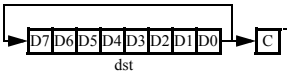
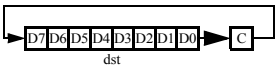
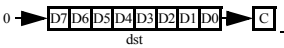
**Table 115. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AND dst, src | dst ← dst AND src | r | r | 52 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 53 | | | | | | | 2 | 4 |
| | | R | R | 54 | | | | | | | 3 | 3 |
| | | R | IR | 55 | | | | | | | 3 | 4 |
| | | R | IM | 56 | | | | | | | 3 | 3 |
| | | IR | IM | 57 | | | | | | | 3 | 4 |
| ANDX dst, src | dst ← dst AND src | ER | ER | 58 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 59 | | | | | | | 4 | 3 |
| ATM | Block all interrupt and DMA requests during execution of the next 3 instructions | | | 2F | – | – | – | – | – | – | 1 | 2 |
| BCLR bit, dst | dst[bit] ← 0 | r | | E2 | – | – | – | – | – | – | 2 | 2 |
| BIT p, bit, dst | dst[bit] ← p | r | | E2 | – | – | – | 0 | – | – | 2 | 2 |
| BRK | Debugger Break | | | 00 | – | – | – | – | – | – | 1 | 1 |
| BSET bit, dst | dst[bit] ← 1 | r | | E2 | – | – | – | 0 | – | – | 2 | 2 |
| BSWAP dst | dst[7:0] ← dst[0:7] | R | | D5 | X | * | * | 0 | – | – | 2 | 2 |
| BTJ p, bit, src, dst | if src[bit] = p<br>PC ← PC + X | | r | F6 | – | – | – | – | – | – | 3 | 3 |
| | | | Ir | F7 | | | | | | | 3 | 4 |
| BTJNZ bit, src, dst | if src[bit] = 1<br>PC ← PC + X | | r | F6 | – | – | – | – | – | – | 3 | 3 |
| | | | Ir | F7 | | | | | | | 3 | 4 |
| BTJZ bit, src, dst | if src[bit] = 0<br>PC ← PC + X | | r | F6 | – | – | – | – | – | – | 3 | 3 |
| | | | Ir | F7 | | | | | | | 3 | 4 |
| CALL dst | SP ← SP -2<br>@SP ← PC<br>PC ← dst | IRR | | D4 | – | – | – | – | – | – | 2 | 6 |
| | | DA | | D6 | | | | | | | 3 | 3 |
| CCF | C ← ~C | | | EF | * | – | – | – | – | – | 1 | 2 |
| CLR dst | dst ← 00H | R | | B0 | – | – | – | – | – | – | 2 | 2 |
| | | IR | | B1 | | | | | | | 2 | 3 |
| Flags Notation: | * = Value is a function of the result of the operation.<br>– = Unaffected<br>X = Undefined | | | | 0 = Reset to 0<br>1 = Set to 1 | | | | | | | |

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

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COM dst | dst ← ~dst | R | | 60 | – | * | * | 0 | – | – | 2 | 2 |
| | | IR | | 61 | | | | | | | 2 | 3 |
| CP dst, src | dst - src | r | r | A2 | * | * | * | * | – | – | 2 | 3 |
| | | r | Ir | A3 | | | | | | | 2 | 4 |
| | | R | R | A4 | | | | | | | 3 | 3 |
| | | R | IR | A5 | | | | | | | 3 | 4 |
| | | R | IM | A6 | | | | | | | 3 | 3 |
| | | IR | IM | A7 | | | | | | | 3 | 4 |
| CPC dst, src | dst - src - C | r | r | 1F A2 | * | * | * | * | – | – | 3 | 3 |
| | | r | Ir | 1F A3 | | | | | | | 3 | 4 |
| | | R | R | 1F A4 | | | | | | | 4 | 3 |
| | | R | IR | 1F A5 | | | | | | | 4 | 4 |
| | | R | IM | 1F A6 | | | | | | | 4 | 3 |
| | | IR | IM | 1F A7 | | | | | | | 4 | 4 |
| CPCX dst, src | dst - src - C | ER | ER | 1F A8 | * | * | * | * | – | – | 5 | 3 |
| | | ER | IM | 1F A9 | | | | | | | 5 | 3 |
| CPX dst, src | dst - src | ER | ER | A8 | * | * | * | * | – | – | 4 | 3 |
| | | ER | IM | A9 | | | | | | | 4 | 3 |
| DA dst | dst ← DA(dst) | R | | 40 | * | * | * | X | – | – | 2 | 2 |
| | | IR | | 41 | | | | | | | 2 | 3 |
| DEC dst | dst ← dst - 1 | R | | 30 | – | * | * | * | – | – | 2 | 2 |
| | | IR | | 31 | | | | | | | 2 | 3 |
| DECW dst | dst ← dst - 1 | RR | | 80 | – | * | * | * | – | – | 2 | 5 |
| | | IRR | | 81 | | | | | | | 2 | 6 |
| DI | IRQCTL[7] ← 0 | | | 8F | – | – | – | – | – | – | 1 | 2 |
| DJNZ dst, RA | dst ← dst – 1<br>if dst ≠ 0<br>PC ← PC + X | r | | 0A-FA | – | – | – | – | – | – | 2 | 3 |
| EI | IRQCTL[7] ← 1 | | | 9F | – | – | – | – | – | – | 1 | 2 |
| Flags Notation: | * = Value is a function of the result of the operation.<br>– = Unaffected<br>X = Undefined | | | | 0 = Reset to 0<br>1 = Set to 1 | | | | | | | |

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

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 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 |

| Flags Notation: | * = Value is a function of the result of the operation.<br>– = Unaffected<br>X = Undefined | 0 = Reset to 0<br>1 = Set to 1 |
|---|---|---|

**Table 123. Flash Memory Electrical Characteristics and Timing**

| Parameter | $V_{DD}$ = 2.7 V to 3.6 V<br>$T_A$ = -40 °C to +105 °C<br>(unless otherwise stated)<br>Minimum | Typical | Maximum | Units | Notes |
|---|---|---|---|---|---|
| Flash Byte Read Time | 100 | – | – | ns | |
| Flash Byte Program Time | 20 | – | 40 | µs | |
| Flash Page Erase Time | 10 | – | – | ms | |
| Flash Mass Erase Time | 200 | – | – | ms | |
| Writes to Single Address Before Next Erase | – | – | 2 | | |
| Flash Row Program Time | – | – | 8 | ms | Cumulative program time for single row cannot exceed limit before next erase. This parameter is only an issue when bypassing the Flash Controller. |
| Data Retention | 100 | – | – | years | 25 °C |
| Endurance | 10,000 | – | – | cycles | Program/erase cycles |

**Table 124. Watchdog Timer Electrical Characteristics and Timing**

| Symbol | Parameter | $V_{DD}$ = 2.7 V to 3.6 V<br>$T_A$ = -40 °C to +105 °C<br>(unless otherwise stated)<br>Minimum | Typical | Maximum | Units | Conditions |
|---|---|---|---|---|---|---|
| $F_{WDT}$ | WDT Oscillator Frequency | | 10 | | kHz | |
| $F_{WDT}$ | WDT Oscillator Error | | | ±50 | % | |
| $T_{WDTCAL}$ | WDT Calibrated Timeout | 0.98 | 1 | 1.02 | s | $V_{DD}$ = 3.3 V;<br>$T_A$ = 30 °C |
| | | 0.70 | 1 | 1.30 | s | $V_{DD}$ = 2.7 V to 3.6 V<br>$T_A$ = 0 °C to 70 °C |
| | | 0.50 | 1 | 1.50 | s | $V_{DD}$ = 2.7 V to 3.6 V<br>$T_A$ = -40 °C to +105 °C |