



Welcome to [E-XFL.COM](https://www.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	20MHz
Connectivity	IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.600", 15.24mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f041apj020ec

Overview

Zilog's Z8 Encore![®] MCU family of products are the first in a line of Zilog[®] microcontroller products based upon the 8-bit eZ8 CPU. Zilog's Z8 Encore! XP[®] F082A Series products expand upon Zilog's extensive line of 8-bit microcontrollers. The Flash in-circuit programming capability allows for faster development time and program changes in the field. The new eZ8 CPU is upward compatible with existing Z8[®] instructions. The rich peripheral set of the Z8 Encore! XP F082A Series makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Features

The key features of Z8 Encore! XP F082A Series products include:

- 20 MHz eZ8 CPU
- 1 KB, 2 KB, 4 KB, or 8 KB Flash memory with in-circuit programming capability
- 256 B, 512 B, or 1 KB register RAM
- Up to 128 B non-volatile data storage (NVDS)
- Internal precision oscillator trimmed to $\pm 1\%$ accuracy
- External crystal oscillator, operating up to 20 MHz
- Optional 8-channel, 10-bit analog-to-digital converter (ADC)
- Optional on-chip temperature sensor
- On-chip analog comparator
- Optional on-chip low-power operational amplifier (LPO)
- Full-duplex UART
- The UART baud rate generator (BRG) can be configured and used as a basic 16-bit timer
- Infrared Data Association (IrDA)-compliant infrared encoder/decoders, integrated with UART
- Two enhanced 16-bit timers with capture, compare, and PWM capability
- Watchdog Timer (WDT) with dedicated internal RC oscillator
- Up to 20 vectored interrupts
- 6 to 25 I/O pins depending upon package

Table 3. Pin Characteristics (20- and 28-pin Devices) (Continued)

Symbol Mnemonic	Direction	Reset Direction	Active Low or Active High	Tristate Output	Internal Pull-up or Pull-down	Schmitt-Trigger Input	Open Drain Output	5 V Tolerance
PC[7:0]	I/O	I	N/A	Yes	Programmable Pull-up	Yes	Yes, Programmable	PC[7:3] unless pullups enabled
RESET/PD0	I/O	I/O (defaults to RESET)	Low (in Reset mode)	Yes (PD0 only)	Programmable for PD0; always on for RESET	Yes	Programmable for PD0; always on for RESET	Yes, unless pullups enabled
VDD	N/A	N/A	N/A	N/A			N/A	N/A
VSS	N/A	N/A	N/A	N/A			N/A	N/A

► **Note:** PB6 and PB7 are available only in those devices without ADC.

Table 4. Pin Characteristics (8-Pin Devices)

Symbol Mnemonic	Direction	Reset Direction	Active Low or Active High	Tristate Output	Internal Pull-up or Pull-down	Schmitt-Trigger Input	Open Drain Output	5 V Tolerance
PA0/DBG	I/O	I (but can change during reset if key sequence detected)	N/A	Yes	Programmable Pull-up	Yes	Yes, Programmable	Yes, unless pull-ups enabled
PA1	I/O	I	N/A	Yes	Programmable Pull-up	Yes	Yes, Programmable	Yes, unless pull-ups enabled
RESET/PA2	I/O	I/O (defaults to RESET)	Low (in Reset mode)	Yes	Programmable for PA2; always on for RESET	Yes	Programmable for PA2; always on for RESET	Yes, unless pull-ups enabled
PA[5:3]	I/O	I	N/A	Yes	Programmable Pull-up	Yes	Yes, Programmable	Yes, unless pull-ups enabled
V _{DD}	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
V _{SS}	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 7. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
F0B	Timer 1 Reload Low Byte	T1RL	FF	88
F0C	Timer 1 PWM High Byte	T1PWMH	00	88
F0D	Timer 1 PWM Low Byte	T1PWML	00	89
F0E	Timer 1 Control 0	T1CTL0	00	83
F0F	Timer 1 Control 1	T1CTL1	00	84
F10–F6F	Reserved	—	XX	
UART				
F40	UART Transmit/Receive Data Registers	TXD, RXD	XX	113
F41	UART Status 0 Register	U0STAT0	00	111
F42	UART Control 0 Register	U0CTL0	00	108
F43	UART Control 1 Register	U0CTL1	00	108
F44	UART Status 1 Register	U0STAT1	00	112
F45	UART Address Compare Register	U0ADDR	00	114
F46	UART Baud Rate High Byte Register	U0BRH	FF	114
F47	UART Baud Rate Low Byte Register	U0BRL	FF	114
Analog-to-Digital Converter (ADC)				
F70	ADC Control 0	ADCCTL0	00	130
F71	ADC Control 1	ADCCTL1	80	130
F72	ADC Data High Byte	ADCD_H	XX	133
F73	ADC Data Low Bits	ADCD_L	XX	133
F74–F7F	Reserved	—	XX	
Low Power Control				
F80	Power Control 0	PWRCTL0	80	35
F81	Reserved	—	XX	
LED Controller				
F82	LED Drive Enable	LEDEN	00	52
F83	LED Drive Level High Byte	LEDLVLH	00	53
F84	LED Drive Level Low Byte	LEDLVLL	00	54
F85	Reserved	—	XX	
Oscillator Control				
F86	Oscillator Control	OSCCTL	A0	190
F87–F8F	Reserved	—	XX	
Comparator 0				
F90	Comparator 0 Control	CMP0	14	136
XX=Undefined				

Table 15. 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
		$\overline{T0OUT}$	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{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	$\overline{CTS0}$	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/LPO Input (P)	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)/LPO Input (N)	AFS1[4]: 1	AFS2[4]: 1
	PA5	TXD0	UART 0 Transmit Data	AFS1[5]: 0	AFS2[5]: 0
		$\overline{T1OUT}$	Timer 1 Output Complement	AFS1[5]: 0	AFS2[5]: 1
		Reserved		AFS1[5]: 1	AFS2[5]: 0
		Analog Functions*	ADC/Comparator Input (P) LPO Output	AFS1[5]: 1	AFS2[5]: 1

*Analog Functions include ADC inputs, ADC reference, comparator inputs and LPO ports.

Note: Also, alternate function selection as described in [Port A–D Alternate Function Sub-Registers](#) on page 47 must be enabled.



Caution: *The following coding style that clears 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
AND r0, MASK
LDX IRQ0, r0
```



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 can generate 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.

U0RXI—UART 0 Receiver Interrupt Request

0 = No interrupt request is pending for the UART 0 receiver.

1 = An interrupt request from the UART 0 receiver is awaiting service.

U0TXI—UART 0 Transmitter Interrupt Request

0 = No interrupt request is pending for the UART 0 transmitter.

1 = An interrupt request from the UART 0 transmitter is awaiting service.

ADCI—ADC Interrupt Request

0 = No interrupt request is pending for the analog-to-digital Converter.

1 = An interrupt request from the Analog-to-Digital Converter is awaiting service.

Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) register ([Table 34](#)) 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 can read the Interrupt Request 1 register to determine if any interrupt requests are pending.

Table 34. 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 A Pin 7 or LVD Interrupt Request

0 = No interrupt request is pending for GPIO Port A or LVD.

1 = An interrupt request from GPIO Port A or LVD.

PA6CI—Port A Pin 6 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.

PAxI—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).

Follow the steps below for configuring a timer for COMPARE mode and initiating the count:

1. Write to the Timer Control register to:
 - Disable the timer.
 - Configure the timer for COMPARE mode.
 - Set the prescale value.
 - Set the initial logic level (High or Low) for the Timer Output alternate function, if appropriate.
2. Write to the Timer High and Low Byte registers to set the starting count value.
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.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control register to enable the timer and initiate counting.

In COMPARE mode, the system clock always provides the timer input. The Compare time can be calculated by the following equation:

$$\text{COMPARE Mode Time (s)} = \frac{(\text{Compare Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

GATED Mode

In GATED mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the TPOL bit in the Timer Control register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is deasserted or a timer reload occurs. To determine if a Timer Input signal deassertion generated the interrupt, read the associated GPIO input value and compare to the value stored in the TPOL bit.

The timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. When reaching the Reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes (assuming the Timer Input signal remains asserted). Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) at timer reset.

Follow the steps below for configuring a timer for GATED mode and initiating the count:

1. Write to the Timer Control register to:
 - Disable the timer.
 - Configure the timer for GATED mode.
 - Set the prescale value.

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.

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 may not 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 in 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.

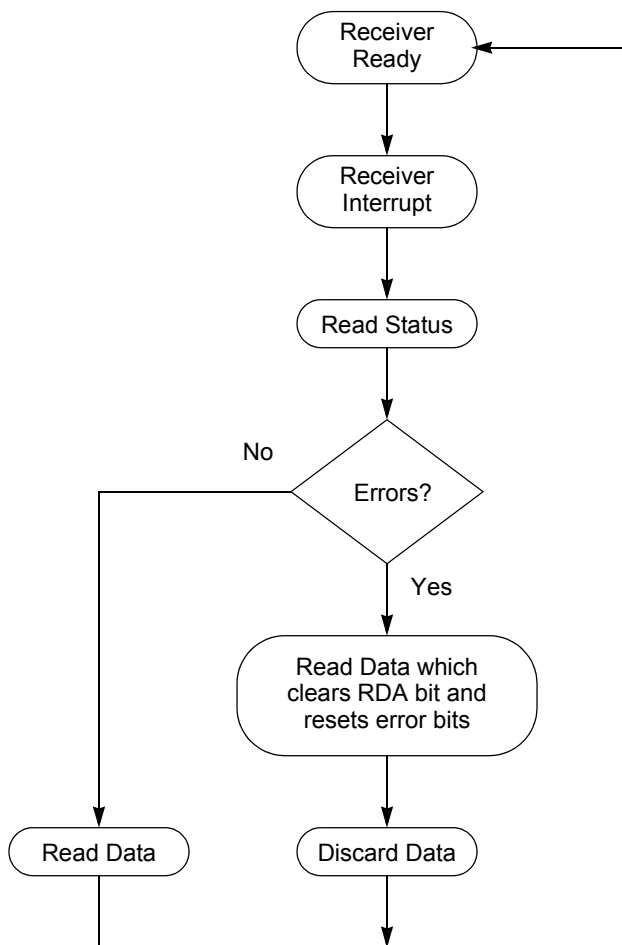


Figure 15. UART Receiver Interrupt Service Routine Flow

Baud Rate Generator Interrupts

If the baud rate generator (BRG) interrupt enable is set, the UART Receiver interrupt asserts when the UART Baud Rate Generator reloads. This condition allows the Baud Rate Generator to function as an additional counter if the UART functionality is not employed.

UART Baud Rate Generator

The UART Baud Rate Generator creates a lower frequency baud rate clock for data transmission. The input to the Baud Rate Generator is the system clock. The UART Baud Rate High and Low Byte registers combine to create a 16-bit baud rate divisor value

(BRG[15:0]) that sets the data transmission rate (baud rate) of the UART. The UART data rate is calculated using the following equation:

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

When the UART is disabled, the Baud Rate Generator functions as a basic 16-bit timer with interrupt on time-out. Follow the steps below to configure the Baud Rate Generator as a timer with interrupt on time-out:

1. Disable the UART by clearing the REN and TEN bits in the UART Control 0 register to 0.
2. Load the acceptable 16-bit count value into the UART Baud Rate High and Low Byte registers.
3. Enable the Baud Rate Generator timer function and associated interrupt by setting the BRGCTL bit in the UART Control 1 register to 1.

When configured as a general purpose timer, the interrupt interval is calculated using the following equation:

$$\text{Interrupt Interval(s)} = \text{System Clock Period (s)} \times \text{BRG[15:0]}$$

UART Control Register Definitions

The UART control registers support the UART and the associated Infrared Encoder/Decoders. For more information on infrared operation, see [Infrared Encoder/Decoder](#) on page 117.

UART Control 0 and Control 1 Registers

The UART Control 0 (UxCTL0) and Control 1 (UxCTL1) registers ([Table 61](#) and [Table 62](#)) configure the properties of the UART's transmit and receive operations. The UART Control registers must not be written while the UART is enabled.

Table 61. UART Control 0 Register (U0CTL0)

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
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	F42H							

TEN—Transmit Enable

This bit enables or disables the transmitter. The enable is also controlled by the $\overline{\text{CTS}}$ signal

Table 70. UART Baud Rates (Continued)

3.579545 MHz System Clock				1.8432 MHz System Clock			
Acceptable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Acceptable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A	625.0	N/A	N/A	N/A
250.0	1	223.72	-10.51	250.0	N/A	N/A	N/A
115.2	2	111.9	-2.90	115.2	1	115.2	0.00
57.6	4	55.9	-2.90	57.6	2	57.6	0.00
38.4	6	37.3	-2.90	38.4	3	38.4	0.00
19.2	12	18.6	-2.90	19.2	6	19.2	0.00
9.60	23	9.73	1.32	9.60	12	9.60	0.00
4.80	47	4.76	-0.83	4.80	24	4.80	0.00
2.40	93	2.41	0.23	2.40	48	2.40	0.00
1.20	186	1.20	0.23	1.20	96	1.20	0.00
0.60	373	0.60	-0.04	0.60	192	0.60	0.00
0.30	746	0.30	-0.04	0.30	384	0.30	0.00

Infrared Encoder/Decoder

The Z8 Encore! XP[®] F082A Series products contain a fully-functional, high-performance UART to Infrared Encoder/Decoder (Endec). The Infrared Endec is integrated with an on-chip UART to allow easy communication between the Z8 Encore! 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.

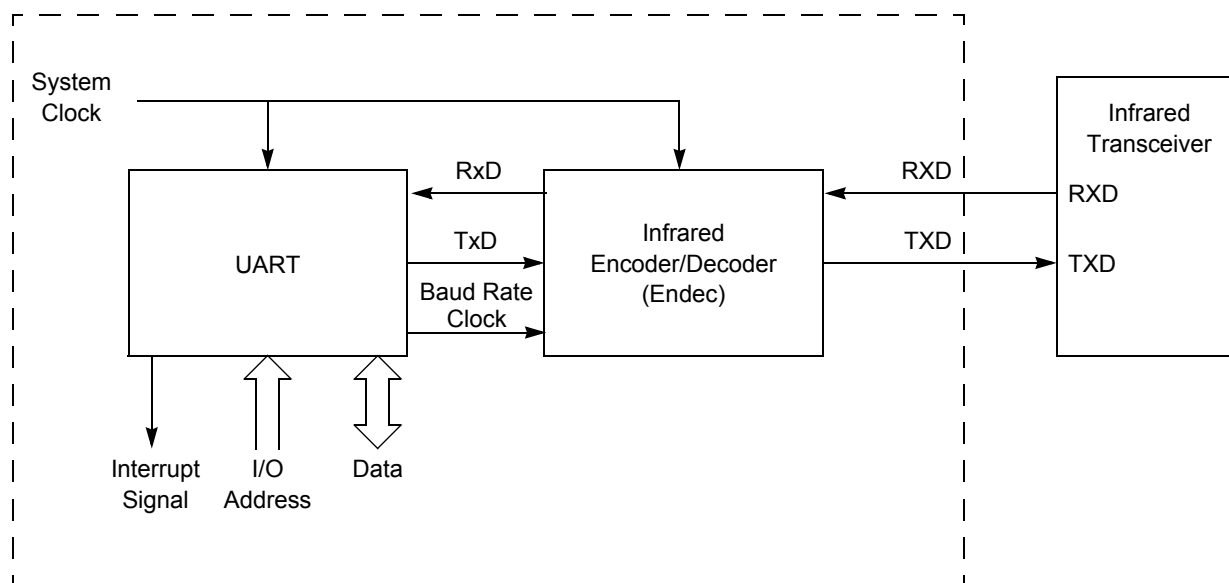


Figure 16. Infrared Data Communication System Block Diagram

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. Likewise, data received from the infrared transceiver is passed to the Infrared Endec through the RXD pin, decoded by the Infrared



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

ADC Compensation Details

High efficiency assembly code that performs this compensation is available for download on www.zilog.com. The following is a bit-specific description of the ADC compensation process used by this code.

The following data bit definitions are used:

0-9, a-f = bit indices in hexadecimal

s = sign bit

v = overflow bit

- = unused

Input Data

MSB	LSB	
s b a 9 8 7 6 5	4 3 2 1 0 - - v	(ADC) ADC Output Word; if v = 1, the data is invalid

s 6 5 4 3 2 1 0	Offset Correction Byte
-----------------	------------------------

s s s s s 7 6 5	4 3 2 1 0 0 0 0	(Offset)	Offset Byte shifted to align with ADC data
<div style="border: 1px solid black; width: 200px; height: 20px; margin: 0 auto;"></div>			

s e d c b a 9 8	7 6 5 4 3 2 1 0	(Gain)	Gain Correction Word
<div style="border: 1px solid black; width: 150px; height: 20px; margin: 0 auto;"></div>			

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

On-Chip Debugger Commands

The host communicates to the on-chip debugger by sending OCD commands using the DBG interface. During normal operation, only a subset of the OCD commands are available. In DEBUG mode, all OCD commands become available unless the user code and control registers are protected by programming the Flash Read Protect Option bit (FRP). The Flash Read Protect Option bit prevents the code in memory from being read out of the Z8 Encore! XP F082A Series products. When this option is enabled, several of the OCD commands are disabled. Table 106 on page 184 is a summary of the On-chip debugger commands. Each OCD command is described in further detail in the bulleted list following this table. Table 106 on page 184 also indicates those commands that operate when the device is not in DEBUG mode (normal operation) and those commands that are disabled by programming the Flash Read Protect Option bit.

Debug Command	Command Byte	Enabled when NOT in DEBUG mode?	Disabled by Flash Read Protect Option Bit
Read OCD Revision	00H	Yes	—
Reserved	01H	—	—
Read OCD Status Register	02H	Yes	—
Read Runtime Counter	03H	—	—
Write OCD Control Register	04H	Yes	Cannot clear DBGMODE bit
Read OCD Control Register	05H	Yes	—
Write Program Counter	06H	—	Disabled
Read Program Counter	07H	—	Disabled

Write Memory, Read Memory, Write Register, Read Register, Read Memory CRC, Step Instruction, Stuff Instruction, and Execute Instruction commands.

```
DBG ← 03H
DBG → RuntimeCounter[15:8]
DBG → RuntimeCounter[7:0]
```

- **Write OCD Control Register (04H)**—The Write OCD Control Register command writes the data that follows to the OCDCTL register. When the Flash Read Protect Option Bit is enabled, the DBGMODE bit (OCDCTL[7]) can only be set to 1, it cannot be cleared to 0 and the only method of returning the device to normal operating mode is to reset the device.

```
DBG ← 04H
DBG ← OCDCTL[7:0]
```

- **Read OCD Control Register (05H)**—The Read OCD Control Register command reads the value of the OCDCTL register.

```
DBG ← 05H
DBG → OCDCTL[7:0]
```

- **Write Program Counter (06H)**—The Write Program Counter command writes the data that follows to the eZ8 CPU's Program Counter (PC). If the device is not in DEBUG mode or if the Flash Read Protect Option bit is enabled, the Program Counter (PC) values are discarded.

```
DBG ← 06H
DBG ← ProgramCounter[15:8]
DBG ← ProgramCounter[7:0]
```

- **Read Program Counter (07H)**—The Read Program Counter command reads the value in the eZ8 CPU's Program Counter (PC). If the device is not in DEBUG mode or if the Flash Read Protect Option bit is enabled, this command returns FFFFH.

```
DBG ← 07H
DBG → ProgramCounter[15:8]
DBG → ProgramCounter[7:0]
```

- **Write Register (08H)**—The Write Register command writes data to the Register File. Data can be written 1–256 bytes at a time (256 bytes can be written by setting size to 0). If the device is not in DEBUG mode, the address and data values are discarded. If the Flash Read Protect Option bit is enabled, only writes to the Flash Control Registers are allowed and all other register write data values are discarded.

```
DBG ← 08H
DBG ← {4'h0, Register Address[11:8]}
DBG ← Register Address[7:0]
DBG ← Size[7:0]
DBG ← 1-256 data bytes
```

If the device is not in DEBUG mode or the Flash Read Protect Option bit is enabled, this command reads and discards one byte.

DBG ← 12H

DBG ← 1-5 byte opcode

On-Chip Debugger Control Register Definitions

OCD Control Register

The OCD Control register controls the state of the On-Chip Debugger. This register is used to enter or exit DEBUG mode and to enable the BRK instruction. It can also reset the Z8 Encore! XP[®] F082A Series device.

A reset and stop function can be achieved by writing 81H to this register. A reset and go function can be achieved by writing 41H to this register. If the device is in DEBUG mode, a run function can be implemented by writing 40H to this register.

Table 106. OCD Control Register (OCDCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	DBGMODE	BRKEN	DBGACK	Reserved				RST
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R	R	R/W

DBGMODE—DEBUG Mode

The device enters DEBUG mode when this bit is 1. When in DEBUG mode, the eZ8 CPU stops fetching new instructions. Clearing this bit causes the eZ8 CPU to restart. This bit is automatically set when a BRK instruction is decoded and Breakpoints are enabled. If the Flash Read Protect Option Bit is enabled, this bit can only be cleared by resetting the device. It cannot be written to 0.

0 = The Z8 Encore! XP F082A Series device is operating in NORMAL mode.

1 = The Z8 Encore! XP F082A Series device is in DEBUG mode.

BRKEN—Breakpoint Enable

This bit controls the behavior of the BRK instruction (opcode 00H). By default, Breakpoints are disabled and the BRK instruction behaves similar to an NOP instruction. If this bit is 1, when a BRK instruction is decoded, the DBGMODE bit of the OCDCTL register is automatically set to 1.

0 = Breakpoints are disabled.

1 = Breakpoints are enabled.



Caution: *It is possible to disable the clock failure detection circuitry as well as all functioning clock sources. In this case, the Z8 Encore! XP F082A Series device ceases functioning and can only be recovered by Power-On-Reset.*

Oscillator Control Register Definitions

Oscillator Control Register

The Oscillator Control Register (OSCCTL) enables/disables the various oscillator circuits, enables/disables the failure detection/recovery circuitry and selects the primary oscillator, which becomes the system clock.

The Oscillator Control Register must be unlocked before writing. Writing the two step sequence E7H followed by 18H to the Oscillator Control Register unlocks it. The register is locked at successful completion of a register write to the OSCCTL.

Table 109. Oscillator Control Register (OSCCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	INTEN	XTLEN	WDTEN	SOFEN	WDFEN	SCKSEL		
RESET	1	0	1	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F86H							

INTEN—Internal Precision Oscillator Enable

1 = Internal precision oscillator is enabled

0 = Internal precision oscillator is disabled

XTLEN—Crystal Oscillator Enable; this setting overrides the GPIO register control for PA0 and PA1

1 = Crystal oscillator is enabled

0 = Crystal oscillator is disabled

WDTEN—Watchdog Timer Oscillator Enable

1 = Watchdog Timer oscillator is enabled

0 = Watchdog Timer oscillator is disabled

SOFEN—System Clock Oscillator Failure Detection Enable

1 = Failure detection and recovery of system clock oscillator is enabled

0 = Failure detection and recovery of system clock oscillator is disabled

Table 121. Logical Instructions (Continued)

Mnemonic	Operands	Instruction
ORX	dst, src	Logical OR using Extended Addressing
XOR	dst, src	Logical Exclusive OR
XORX	dst, src	Logical Exclusive OR using Extended Addressing

Table 122. Program Control Instructions

Mnemonic	Operands	Instruction
BRK	—	On-Chip Debugger Break
BTJ	p, bit, src, DA	Bit Test and Jump
BTJNZ	bit, src, DA	Bit Test and Jump if Non-Zero
BTJZ	bit, src, DA	Bit Test and Jump if Zero
CALL	dst	Call Procedure
DJNZ	dst, src, RA	Decrement and Jump Non-Zero
IRET	—	Interrupt Return
JP	dst	Jump
JP cc	dst	Jump Conditional
JR	DA	Jump Relative
JR cc	DA	Jump Relative Conditional
RET	—	Return
TRAP	vector	Software Trap

Table 123. 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

Table 124. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
LDC dst, src	$\text{dst} \leftarrow \text{src}$	r	lrr	C2	–	–	–	–	–	–	2	5
		lr	lrr	C5							2	9
		lrr	r	D2							2	5
LDCI dst, src	$\text{dst} \leftarrow \text{src}$ $r \leftarrow r + 1$ $rr \leftarrow rr + 1$	lr	lrr	C3	–	–	–	–	–	–	2	9
		lrr	lr	D3							2	9
LDE dst, src	$\text{dst} \leftarrow \text{src}$	r	lrr	82	–	–	–	–	–	–	2	5
		lrr	r	92							2	5
LDEI dst, src	$\text{dst} \leftarrow \text{src}$ $r \leftarrow r + 1$ $rr \leftarrow rr + 1$	lr	lrr	83	–	–	–	–	–	–	2	9
		lrr	lr	93							2	9
LDWX dst, src	$\text{dst} \leftarrow \text{src}$	ER	ER	1FE8	–	–	–	–	–	–	5	4
LDX dst, src	$\text{dst} \leftarrow \text{src}$	r	ER	84	–	–	–	–	–	–	3	2
		lr	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	lr	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)	$\text{dst} \leftarrow \text{src} + X$	r	X(r)	98	–	–	–	–	–	–	3	3
		rr	X(rr)	99							3	5
MULT dst	$\text{dst}[15:0] \leftarrow \text{dst}[15:8] * \text{dst}[7:0]$	RR		F4	–	–	–	–	–	–	2	8
NOP	No operation			0F	–	–	–	–	–	–	1	2
Flags Notation:	* = Value is a function of the result of the operation. – = Unaffected X = Undefined				0 = Reset to 0 1 = Set to 1							