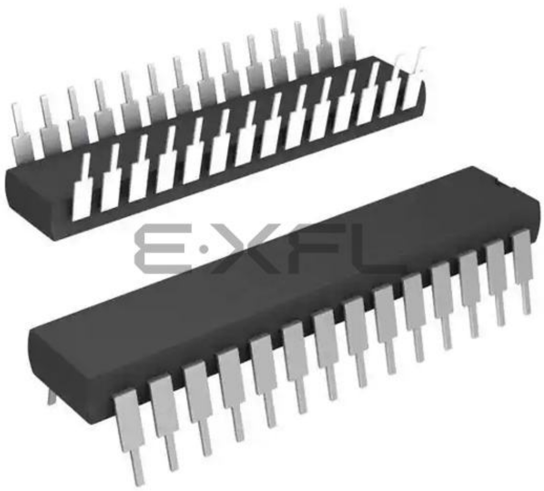

Welcome to E-XFL.COM

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	20MHz
Connectivity	IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	2KB (2K x 8)
Program Memory Type	FLASH
EEPROM Size	64 x 8
RAM Size	512 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/z8f021apj020eg

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:

- 20MHz eZ8 CPU
- 1KB, 2KB, 4KB, or 8KB Flash memory with in-circuit programming capability
- 256B, 512B, or 1KB register RAM
- Up to 128B nonvolatile data storage (NVDS)
- Internal precision oscillator trimmed to ±1% accuracy
- External crystal oscillator, operating up to 20MHz
- 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 the 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
- Up to thirteen 5 V-tolerant input pins

Table 2. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
Analog		
ANA[7:0]	I	Analog Port. These signals are used as inputs to the analog-to-digital converter (ADC).
VREF	I/O	Analog-to-digital converter reference voltage input, or buffered output for internal reference.
Low-Power Operational Amplifier (LPO)		
AMPINP/AMPINN	I	LPO inputs. If enabled, these pins drive the positive and negative amplifier inputs respectively.
AMPOUT	O	LPO output. If enabled, this pin is driven by the on-chip LPO.
Oscillators		
XIN	I	External Crystal Input. This is the input pin to the crystal oscillator. A crystal can be connected between it and the X_{OUT} pin to form the oscillator. In addition, this pin is used with external RC networks or external clock drivers to provide the system clock.
X_{OUT}	O	External Crystal Output. This pin is the output of the crystal oscillator. A crystal can be connected between it and the **XIN** pin to form the oscillator.
Clock Input		
CLKIN	I	Clock Input Signal. This pin may be used to input a TTL-level signal to be used as the system clock.
LED Drivers		
LED	O	Direct LED drive capability. All port C pins have the capability to drive an LED without any other external components. These pins have programmable drive strengths set by the GPIO block.
On-Chip Debugger		
DBG	I/O	Debug. This signal is the control and data input and output to and from the On-Chip Debugger. **Caution:** The DBG pin is open-drain and requires a pull-up resistor to ensure proper operation.

Notes:
1. PB6 and PB7 are only available in 28-pin packages without ADC. In 28-pin packages with ADC, they are replaced by AV_{DD} and AV_{SS}.
2. The AV_{DD} and AV_{SS} signals are available only in 28-pin packages with ADC. They are replaced by PB6 and PB7 on 28-pin packages without ADC.

Table 7. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page
F85	Reserved	—	XX	
Oscillator Control				
F86	Oscillator Control	OSCCTL	A0	196
F87–F8F	Reserved	—	XX	
Comparator 0				
F90	Comparator 0 Control	CMP0	14	141
F91–FBF	Reserved	—	XX	
Interrupt Controller				
FC0	Interrupt Request 0	IRQ0	00	60
FC1	IRQ0 Enable High Bit	IRQ0ENH	00	63
FC2	IRQ0 Enable Low Bit	IRQ0ENL	00	63
FC3	Interrupt Request 1	IRQ1	00	61
FC4	IRQ1 Enable High Bit	IRQ1ENH	00	65
FC5	IRQ1 Enable Low Bit	IRQ1ENL	00	65
FC6	Interrupt Request 2	IRQ2	00	62
FC7	IRQ2 Enable High Bit	IRQ2ENH	00	66
FC8	IRQ2 Enable Low Bit	IRQ2ENL	00	67
FC9–FCC	Reserved	—	XX	
FCD	Interrupt Edge Select	IRQES	00	68
FCE	Shared Interrupt Select	IRQSS	00	68
FCF	Interrupt Control	IRQCTL	00	69
GPIO Port A				
FD0	Port A Address	PAADDR	00	44
FD1	Port A Control	PACTL	00	46
FD2	Port A Input Data	PAIN	XX	46
FD3	Port A Output Data	PAOUT	00	46
GPIO Port B				
FD4	Port B Address	PBADDR	00	44
FD5	Port B Control	PBCTL	00	46
FD6	Port B Input Data	PBIN	XX	46
FD7	Port B Output Data	PBOUT	00	46
GPIO Port C				
FD8	Port C Address	PCADDR	00	44

Notes:
1. XX = Undefined.
2. Refer to the eZ8 CPU Core User Manual (UM0128).

Shared Debug Pin

On the 8-pin version of this device only, the Debug pin shares function with the PA0 GPIO pin. This pin performs as a general purpose input pin on power-up, but the debug logic monitors this pin during the reset sequence to determine if the unlock sequence occurs. If the unlock sequence is present, the debug function is unlocked and the pin no longer functions as a GPIO pin. If it is not present, the debug feature is disabled until/unless another reset event occurs. For more details, see the On-Chip Debugger chapter on page 180**.**

Crystal Oscillator Override

For systems using a crystal oscillator, PA0 and PA1 are used to connect the crystal. When the crystal oscillator is enabled, the GPIO settings are overridden and PA0 and PA1 are disabled. See the Oscillator Control Register Definitions section on page 196 for details.

5V Tolerance

All six I/O pins on the 8-pin devices are 5V-tolerant, unless the programmable pull-ups are enabled. If the pull-ups are enabled and inputs higher than V_{DD} are applied to these parts, excessive current flows through those pull-up devices and can damage the chip.

Note: In the 20- and 28-pin versions of this device, any pin which shares functionality with an ADC, crystal or comparator port is not 5 V-tolerant, including PA[1:0], PB[5:0] and PC[2:0]. All other signal pins are 5 V-tolerant and can safely handle inputs higher than V_{DD} except when the programmable pull-ups are 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 (OSCCTL) Register such that the external oscillator is selected as the system clock. See the Oscillator Control Register Definitions section on page 196 for details. For 8-pin devices, use PA1 instead of PB3.

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[1]	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	$\overline{\text{CTS0}}$	$\overline{\text{UART 0 Clear to Send}}$	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[2]	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[2]	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
		$\overline{\text{T1OUT}}$	Timer 1 Output Complement	AFS1[5]: 0	AFS2[5]: 1
		Reserved		AFS1[5]: 1	AFS2[5]: 0
		Analog Functions[2]	ADC/Comparator Input (P) LPO Output	AFS1[5]: 1	AFS2[5]: 1

Notes:
1. Analog functions include ADC inputs, ADC reference, comparator inputs and LPO ports.
2. The alternate function selection must be enabled; see the Port A–D Alternate Function Subregisters (PxAF) section on page 47 for details.

Table 27. Port A–D Alternate Function Set 1 Subregisters (PxAFS1)

Bit	7	6	5	4	3	2	1	0
Field	PAFS17	PAFS16	PAFS15	PAFS14	PAFS13	PAFS12	PAFS11	PAFS10
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	If 07H in Port A–D Address Register, accessible through the Port A–D Control Register							

Bit	Description
[7:0] PAFSx	**Port Alternate Function Set 1** 0 = Port Alternate Function selected, as defined in Tables 15 and 16 on page 43. 1 = Port Alternate Function selected, as defined in Tables 15 and 16 on page 43.

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

Port A–D Alternate Function Set 2 Subregisters

The Port A–D Alternate Function Set 2 Subregister, shown in Table 28, is accessed through the Port A–D Control Register by writing `08H` to the Port A–D Address Register. The Alternate Function Set 2 subregisters selects the alternate function available at a port pin. Alternate Functions selected by setting or clearing bits of this register is defined in Table 16 on page 43.

Note: Alternate function selection on the port pins must also be enabled. See the Port A–D Alternate Function Subregisters section on page 47 for details.

Table 28. Port A–D Alternate Function Set 2 Subregisters (PxAFS2)

Bit	7	6	5	4	3	2	1	0
Field	PAFS27	PAFS26	PAFS25	PAFS24	PAFS23	PAFS22	PAFS21	PAFS20
RESET	00H (all ports of 20/28 pin devices); 04H (Port A of 8-pin device)							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 08H in Port A–D Address Register, accessible through the Port A–D Control Register							

Bit	Description
[7] PAFS2x	**Port Alternate Function Set 2** 0 = Port Alternate Function selected, as defined in Table 16. 1 = Port Alternate Function selected, as defined in Table 16.

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

LED Drive Enable Register

The LED Drive Enable Register, shown in Table 31, activates the controlled current drive. The Alternate Function Register has no control over the LED function; therefore, setting the Alternate Function Register to select the LED function is not required. LEDEN bits [7:0] correspond to Port C bits [7:0], respectively.

Table 31. LED Drive Enable (LEDEN)

Bit	7	6	5	4	3	2	1	0
Field	LEDEN[7:0]							
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	F82H							

Bit	Description
[7:0] LEDENx	**LED Drive Enable** These bits determine which Port C pins are connected to an internal current sink. 0 = Tristate the Port C pin. 1 = Enable controlled current sink on the Port C pin. **Note:** *x* indicates the specific GPIO port pin number (7–0).

LED Drive Level High Register

The LED Drive Level registers contain two control bits for each Port C pin, as shown in Table 32. These two bits select between four programmable drive levels. Each pin is individually programmable.

Table 32. LED Drive Level High Register (LEDLVLH)

Bit	7	6	5	4	3	2	1	0
Field	LEDLVLH[7:0]							
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	F83H							

Bit	Description
[7:0] LEDLVLHx	**LED Level High Bit** {LEDLVLH, LEDLVLL} select one of four programmable current drive levels for each Port C pin. 00 = 3mA 01 = 7mA 10 = 13mA 11 = 20mA
Note:	x indicates the specific GPIO port pin number (7–0).

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

Interrupts are globally disabled by any of the following actions:

- Execution of a Disable Interrupt (DI) instruction
- eZ8 CPU acknowledgement of an interrupt service request from the interrupt controller
- Writing a 0 to the `IRQE` bit in the Interrupt Control Register
- Reset
- Execution of a Trap instruction
- Illegal Instruction Trap
- Primary Oscillator Fail Trap
- Watchdog Oscillator Fail Trap

Interrupt Vectors and Priority

The interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority and Level 1 is the lowest priority. If all of the interrupts are enabled with identical interrupt priority (all as Level 2 interrupts, for example), the interrupt priority is assigned from highest to lowest as specified in Table 34 on page 56. Level 3 interrupts are always assigned higher priority than Level 2 interrupts which, in turn, always are assigned higher priority than Level 1 interrupts. Within each interrupt priority level (Level 1, Level 2, or Level 3), priority is assigned as specified in Table 34, above. Reset, Watchdog Timer interrupt (if enabled), Primary Oscillator Fail Trap, Watchdog Oscillator Fail Trap and Illegal Instruction Trap always have highest (level 3) priority.

Interrupt Assertion

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

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

---

delay ensures a time gap between the deassertion of one PWM output to the assertion of its complement.

Observe the following steps for configuring 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 by writing the TMODE bits in the TxCTL1 Register and the TMODEHI bit in TxCTL0 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 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.

   – 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 TxCTL0 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.
7. Counting begins on the first appropriate transition of the Timer Input signal. No interrupt is generated by this first edge.

In CAPTURE/COMPARE Mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

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

Reading the Timer Count Values

The current count value in the timers can be read while counting (enabled). This capability has no effect on timer operation. When the timer is enabled and the Timer High Byte Register is read, the contents of the Timer Low Byte Register are placed in a holding register. A subsequent read from the Timer Low Byte Register returns the value in the holding register. This operation allows accurate reads of the full 16-bit timer count value while enabled. When the timers are not enabled, a read from the Timer Low Byte Register returns the actual value in the counter.

Timer Pin Signal Operation

The timer output function is a GPIO port pin alternate function. The Timer Output is toggled every time the counter is reloaded.

6. Read data from the UART Receive Data Register. If operating in MULTIPROCESSOR (9-bit) Mode, further actions may be required depending on the MULTIPROCESSOR Mode bits MPMD[1:0].
7. Return to Step 4 to receive additional data.

Receiving Data using the Interrupt-Driven Method

The UART Receiver interrupt indicates the availability of new data (and error conditions). Observe the following steps to configure the UART receiver for interrupt-driven operation:

1. Write to the UART Baud Rate High and Low Byte registers to set the acceptable 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 Receiver interrupt and set the acceptable priority.
5. Clear the UART Receiver interrupt in the applicable Interrupt Request Register.
6. Write to the UART Control 1 Register to enable Multiprocessor (9-bit) mode functions, if appropriate.
   - Set the Multiprocessor Mode Select (`MPEN`) to Enable MULTIPROCESSOR Mode.
   - Set the Multiprocessor Mode Bits, MPMD[1:0], to select the acceptable address matching scheme.
   - Configure the UART to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for Z8 Encore! devices without a DMA block)
7. Write the device address to the Address Compare Register (automatic MULTIPROCESSOR Modes only).
8. Write to the UART Control 0 Register to:
   - Set the receive enable bit (`REN`) to enable the UART for data reception
   - Enable parity, if appropriate and if multiprocessor mode is not enabled and select either even or odd parity
9. Execute an EI instruction to enable interrupts.

Receiving IrDA Data

Data received from the infrared transceiver using the IR_RXD signal through the RXD pin is decoded by the infrared endec and passed to the UART. The UART's baud rate clock is used by the infrared endec to generate the demodulated signal (RXD) that drives the UART. Each UART/Infrared data bit is 16-clocks wide. Figure 18 displays data reception. When the infrared endec is enabled, the UART's RXD signal is internal to the Z8 Encore! XP F082A Series products while the IR_RXD signal is received through the RXD pin.

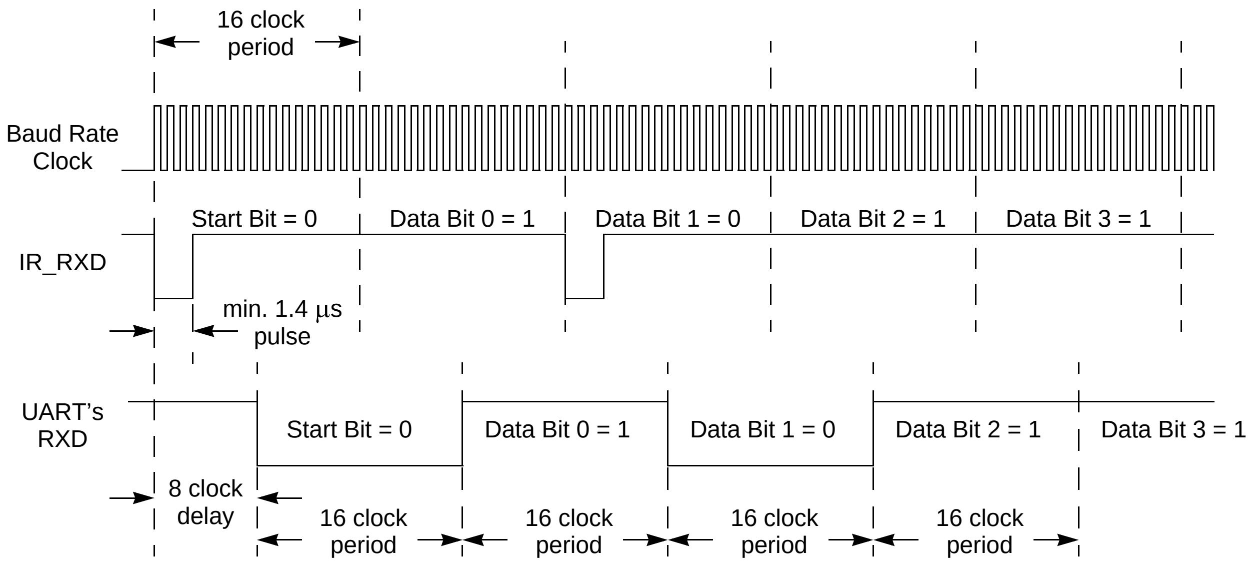

Figure 18. IrDA Data Reception

Infrared Data Reception

! **Caution:** The system clock frequency must be at least 1.0 MHz to ensure proper reception of the 1.4 µs minimum width pulses allowed by the IrDA standard.

Endec Receiver Synchronization

The IrDA receiver uses a local baud rate clock counter (0 to 15 clock periods) to generate an input stream for the UART and to create a sampling window for detection of incoming pulses. The generated UART input (UART RXD) is delayed by 8 baud rate clock periods with respect to the incoming IrDA data stream. When a falling edge in the input data stream is detected, the Endec counter is reset. When the count reaches a value of 8, the UART RXD value is updated to reflect the value of the decoded data. When the count reaches 12 baud clock periods, the sampling window for the next incoming pulse opens.

Flash Operation Timing Using the Flash Frequency Registers

Before performing either a program or erase operation on Flash memory, you must first configure the Flash Frequency High and Low Byte registers. The Flash Frequency registers allow programming and erasing of the Flash with system clock frequencies ranging from 32kHz (32768Hz) through 20MHz.

The Flash Frequency High and Low Byte registers combine to form a 16-bit value, `FFREQ`, to control timing for Flash program and erase operations. The 16-bit binary Flash Frequency value must contain the system clock frequency (in kHz). This value is calculated using the following equation:

$$\text{FFREQ[15:0]} = \frac{\text{System Clock Frequency (Hz)}}{1000}$$

! **Caution:** Flash programming and erasure are not supported for system clock frequencies below 32kHz (32768Hz) or above 20MHz. The Flash Frequency High and Low Byte registers must be loaded with the correct value to ensure operation of the Z8 Encore! XP F082A Series devices.

Flash Code Protection Against External Access

The user code contained within the Flash memory can be protected against external access by the on-chip debugger. Programming the FRP Flash option bit prevents reading of the user code with the On-Chip Debugger. See the Flash Option Bits chapter on page 159 and the On-Chip Debugger chapter on page 180 for more information.

Flash Code Protection Against Accidental Program and Erasure

The Z8 Encore! XP F082A Series provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by a combination of the Flash option bits, the register locking mechanism, the page select redundancy and the sector level protection control of the Flash Controller.

Flash Code Protection Using the Flash Option Bits

The FRP and FWP Flash option bits combine to provide three levels of Flash Program Memory protection, as shown in Table 79. See the Flash Option Bits chapter on page 159 for more information.

Trim Bit Address Space

All available Trim bit addresses and their functions are listed in Table 90 through Table 95.

Trim Bit Address 0000H

Table 90. Trim Options Bits at Address 0000H

Bit	7	6	5	4	3	2	1	0
Field	Reserved							
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
Address	Information Page Memory 0020H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

Bit	Description
[7:0]	**Reserved** These bits are reserved; altering this register may result in incorrect device operation.

Trim Bit Address 0001H

Table 91. Trim Option Bits at 0001H

Bit	7	6	5	4	3	2	1	0
Field	Reserved							
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
Address	Information Page Memory 0021H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

Bit	Description
[7:0]	**Reserved** These bits are reserved; altering this register may result in incorrect device operation.

Byte Write

To write a byte to the NVDS array, the user code must first push the address, then the data byte onto the stack. The user code issues a `CALL` instruction to the address of the byte-write routine (0x10B3). At the return from the sub-routine, the write status byte resides in working register R0. The bit fields of this status byte are defined in Table 106. The contents of the status byte are undefined for write operations to illegal addresses. Also, user code must pop the address and data bytes off the stack.

The write routine uses 13 bytes of stack space in addition to the two bytes of address and data pushed by the user. Sufficient memory must be available for this stack usage.

Because of the Flash memory architecture, NVDS writes exhibit a nonuniform execution time. In general, a write takes 251 µs (assuming a 20 MHz system clock). Every 400 to 500 writes, however, a maintenance operation is necessary. In this rare occurrence, the write takes up to 61 ms to complete. Slower system clock speeds result in proportionally higher execution times.

NVDS byte writes to invalid addresses (those exceeding the NVDS array size) have no effect. Illegal write operations have a 2 µs execution time.

Table 106. Write Status Byte

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				RCPY	PF	AWE	DWE
Default Value	0	0	0	0	0	0	0	0

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] RCPY	**Recopy Subroutine Executed** A recopy subroutine was executed. These operations take significantly longer than a normal write operation.
[2] PF	**Power Failure Indicator** A power failure or system reset occurred during the most recent attempted write to the NVDS array.
[1] AWE	**Address Write Error** An address byte failure occurred during the most recent attempted write to the NVDS array.
[0] DWE	**Data Write Error** A data byte failure occurred during the most recent attempted write to the NVDS array.

Oscillator Operation with an External RC Network

Figure 28 displays a recommended configuration for connection with an external resistor-capacitor (RC) network.

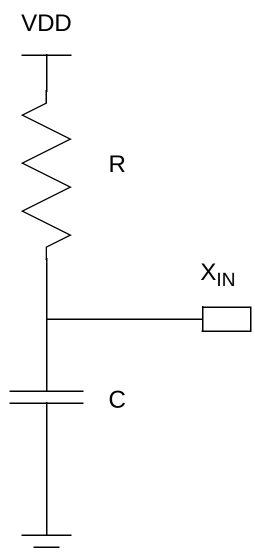

Figure 28. Connecting the On-Chip Oscillator to an External RC Network

An external resistance value of 45kΩ is recommended for oscillator operation with an external RC network. The minimum resistance value to ensure operation is 40 kΩ. The typical oscillator frequency can be estimated from the values of the resistor (*R* in kΩ) and capacitor (*C* in pF) elements using the following equation:

$$\text{Oscillator Frequency (kHz)} = \frac{1 \times 10^6}{(0.4 \times R \times C) + (4 \times C)}$$

Figure 29 displays the typical (3.3V and 25°C) oscillator frequency as a function of the capacitor (C, in pF) employed in the RC network assuming a 45KΩ external resistor. For very small values of C, the parasitic capacitance of the oscillator X_{IN} pin and the printed circuit board must be included in the estimation of the oscillator frequency.

It is possible to operate the RC oscillator using only the parasitic capacitance of the package and printed circuit board. To minimize sensitivity to external parasitics, external capacitance values in excess of 20pF are recommended.

Table 118. Notational Shorthand (Continued)

Notation	Description	Operand	Range
Vector	Vector address	Vector	Vector represents a number in the range of 00H to FFH.
X	Indexed	#Index	The register or register pair to be indexed is offset by the signed Index value (#Index) in a +127 to –128 range.

Table 119 lists additional symbols that are used throughout the Instruction Summary and Instruction Set Description sections.

Table 119. Additional Symbols

Symbol	Definition
dst	Destination Operand
src	Source Operand
@	Indirect Address Prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flags Register
RP	Register Pointer
#	Immediate Operand Prefix
B	Binary Number Suffix
%	Hexadecimal Number Prefix
H	Hexadecimal Number Suffix

Assignment of a value is indicated by an arrow, as shown in the following example.

dst ← dst + src

This example indicates that the source data is added to the destination data; the result is stored in the destination location.

eZ8 CPU Instruction Classes

eZ8 CPU instructions can be divided functionally into the following groups:

- Arithmetic
- Bit Manipulation

Table 123. CPU Control Instructions (Continued)

Mnemonic	Operands	Instruction
RCF	—	Reset Carry Flag
SCF	—	Set Carry Flag
SRP	src	Set Register Pointer
STOP	—	STOP Mode
WDT	—	Watchdog Timer Refresh

Table 124. 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 125. Logical Instructions

Mnemonic	Operands	Instruction
AND	dst, src	Logical AND
ANDX	dst, src	Logical AND using Extended Addressing
COM	dst	Complement
OR	dst, src	Logical OR
ORX	dst, src	Logical OR using Extended Addressing
XOR	dst, src	Logical Exclusive OR
XORX	dst, src	Logical Exclusive OR using Extended Addressing

Table 126. 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 127. Rotate and Shift Instructions

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

Table 128. 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
TMX dst, src	dst AND src	ER	ER	78	–	*	*	0	–	–	4	3
		ER	IM	79							4	3
TRAP Vector	SP ← SP – 2 @SP ← PC SP ← SP – 1 @SP ← FLAGS PC ← @Vector		Vector	F2	–	–	–	–	–	–	2	6
WDT				5F	–	–	–	–	–	–	1	2
XOR dst, src	dst ← dst XOR src	r	r	B2	–	*	*	0	–	–	2	3
		r	Ir	B3							2	4
		R	R	B4							3	3
		R	IR	B5							3	4
		R	IM	B6							3	3
		IR	IM	B7							3	4
XORX dst, src	dst ← dst XOR src	ER	ER	B8	–	*	*	0	–	–	4	3
		ER	IM	B9							4	3

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.