Embedded in Life

An IXYS Company

High Performance 8-Bit Microcontrollers

Z8 Encore! XP® F0822 Series

Product Specification

PS022518-1011

www.zilog.com

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Revision History

Each instance in Revision History reflects a change to this document from its previous revision. For more details, refer to the corresponding pages and appropriate links in the table below.

Date	Revision Level	Description	Page
Oct 2011	18	Added LDWX information to Load Instructions table, eZ8 CPU Instruction Summary table and to Second Op Code Map after 1FH figure; revised Flash Sector Protect Register description; revised Packaging chapter.	206, 212, 220, 152, 221
May 2008	17	Removed *Flash Microcontrollers* from the title throughout the document.	All
Feb 2008	16	Updated the flag status for BCLR, BIT, and BSET in eZ8 CPU Instruction Summary table.	208
Dec 2007	15	Updated Zilog logo/text, Foreword section. Updated Z8 Encore! 8K Series to Z8 Encore! XP® F0822 Series Flash Microcontrollers throughout the document.	All

Table of Contents

List of Figures

List of Tables

Introduction

Zilog's Z8 Encore! XP® MCU product family is a line of Zilog microcontrollers based on the 8-bit eZ8 CPU. Z8 Encore! XP® F0822 Series of MCUs adds Flash memory to Zilog's extensive line of 8-bit microcontrollers. The Flash in-circuit programming allows faster development time and program changes in the field. The new eZ8 CPU is upward-compatible with the existing Z8® CPU instructions. The rich peripheral set of the Z8 Encore! XP® F0822 Series makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices and sensors.

Features

The Z8 Encore! XP® F0822 Series features:

- 20MHz eZ8 CPU core
- Up to 8KB Flash with in-circuit programming capability
- 1KB Register RAM
- Optional 2- to 5-channel, 10-bit Analog-to-Digital Converter (ADC)
- Full-duplex 9-bit Universal Asynchronous Receiver/Transmitter (UART) with bus transceiver Driver Enable Control
- Inter-Integrated Circuit (I^2C)
- Serial Peripheral Interface (SPI)
- Infrared Data Association (IrDA)-compliant infrared encoder/decoders
- Two 16-bit timers with Capture, Compare, and PWM capability
- Watchdog Timer (WDT) with internal RC oscillator
- 11 to 19 Input/Output pins depending upon package
- Up to 19 interrupts with configurable priority
- On-Chip Debugger (OCD)
- Voltage Brown-Out (VBO) protection
- Power-On Reset (POR)
- Crystal oscillator with three power settings and RC oscillator option
- 2.7V to 3.6V operating voltage with 5V-tolerant inputs
- 20-pin and 28-pin packages

- 0°C to +70°C standard temperature and –40°C to +105°C extended temperature operating ranges

Part Selection Guide

Table 1 identifies the basic features and package styles available for each device within the Z8 Encore! XP® F0822 Series.

Table 1. Z8 Encore! XP® F0822 Series Part Selection Guide

Part Number	Flash (KB)	RAM (KB)	I/O	16-bit Timers with PWM	ADC Inputs	UARTs with IrDA	I^2C	SPI	Package Pin Counts	
									20	28
Z8F0822	8	1	19	2	5	1	1	1		X
Z8F0821	8	1	11	2	2	1	1		X	
Z8F0812	8	1	19	2	0	1	1	1		X
Z8F0811	8	1	11	2	0	1	1		X	
Z8F0422	4	1	19	2	5	1	1	1		X
Z8F0421	4	1	11	2	2	1	1		X	
Z8F0412	4	1	19	2	0	1	1	1		X
Z8F0411	4	1	11	2	0	1	1		X	

Block Diagram

Figure 1 displays the block diagram of the architecture of Z8 Encore! XP® F0822 Series devices.

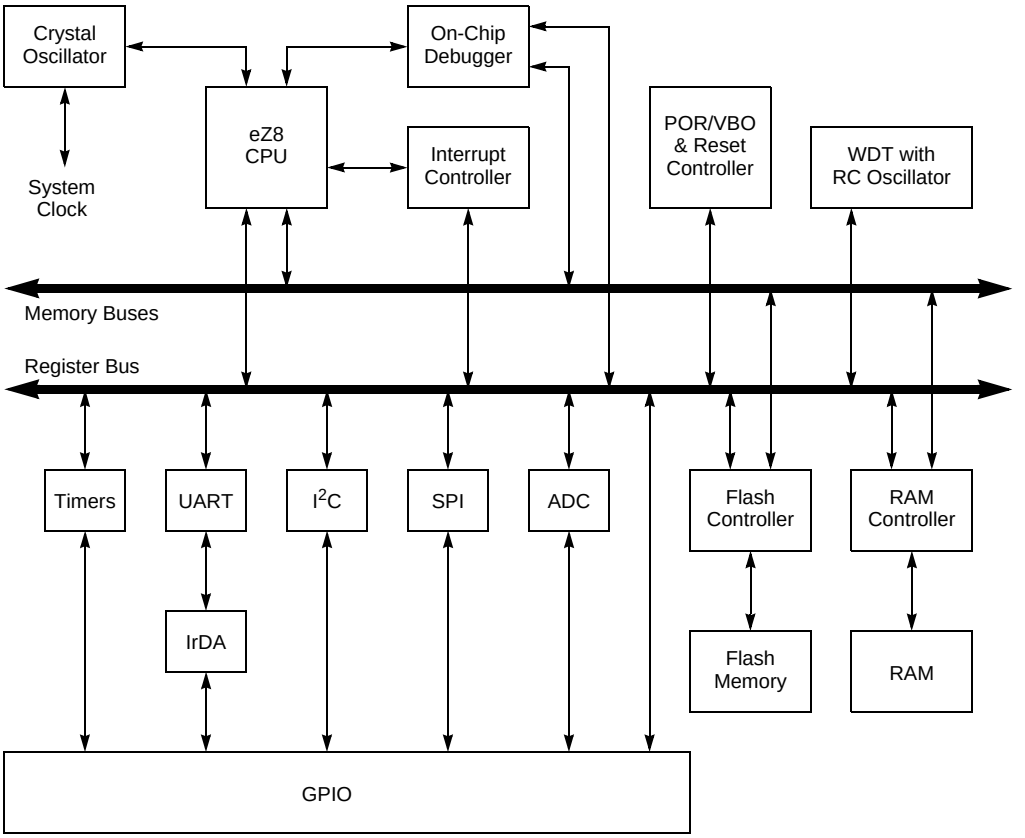


Figure 1. Z8 Encore! XP® F0822 Series Block Diagram

CPU and Peripheral Overview

Zilog's latest eZ8 8-bit CPU meets the continuing demand for faster and more code-efficient microcontrollers. The eZ8 CPU executes a superset of the original Z8® instruction set.

The eZ8 CPU features:

- Direct register-to-register architecture allows each register to function as an accumulator, improving execution time and decreasing the required Program memory
- Software stack allows much greater depth in subroutine calls and interrupts than hardware stacks
- Compatible with existing Z8® code
- Expanded internal Register File allows access of up to 4KB
- New instructions improve execution efficiency for code developed using higher-level programming languages, including C
- Pipelined instruction fetch and execution
- New instructions for improved performance including BIT, BSWAP, BTJ, CPC, LDC, LDCI, LEA, MULT and SRL
- New instructions support 12-bit linear addressing of the Register File
- Up to 10 MIPS operation
- C-Compiler friendly
- 2 to 9 clock cycles per instruction

For more information about the eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download at www.zilog.com.

General Purpose Input/Output

Z8 Encore! XP® F0822 Series features 11 to 19 port pins (Ports A–C) for General-Purpose Input/Output (GPIO). The number of available GPIO pins is a function of package type. Each pin is individually programmable. Ports A and C support 5V-tolerant inputs.

Flash Controller

The Flash Controller programs and erases the contents of Flash memory.

10-Bit Analog-to-Digital Converter

The optional Analog-to-Digital Converter (ADC) converts an analog input signal to a 10-bit binary number. The ADC accepts inputs from 2 to 5 different analog input sources.

UART

The Universal Asynchronous Receiver/Transmitter (UART) is full-duplex and capable of handling asynchronous data transfers. The UART supports 8-bit and 9-bit data modes and selectable parity.

I^2C

The Inter-Integrated Circuit (I^2C) controller makes the Z8 Encore! XP compatible with the I^2C protocol. The I^2C Controller consists of two bidirectional bus lines, a serial data (SDA) line, and a serial clock (SCL) line.

Serial Peripheral Interface

The Serial Peripheral Interface (SPI) allows the Z8 Encore! XP to exchange data between other peripheral devices such as EEPROMs, A/D converters, and ISDN devices. The SPI is a full-duplex, synchronous, and character-oriented channel that supports a four-wire interface.

Timers

Two 16-bit reloadable timers are used for timing/counting events or for motor control operations. These timers provide a 16-bit programmable reload counter and operate in One-Shot, Continuous, Gated, Capture, Compare, Capture and Compare, and PWM modes.

Interrupt Controller

Z8 Encore! XP® F0822 Series products support up to 18 interrupts. These interrupts consist of 7 internal peripheral interrupts and 11 GPIO pin interrupt sources. The interrupts have 3 levels of programmable interrupt priority.

Reset Controller

Z8 Encore! XP® F0822 Series products are reset using the $\overline{\text{RESET}}$ pin, POR, WDT, STOP Mode exit, or VBO warning signal.

On-Chip Debugger

Z8 Encore! XP® F0822 Series products feature an integrated On-Chip Debugger (OCD). The OCD provides a rich-set of debugging capabilities, such as, reading and writing registers, programming Flash memory, setting breakpoints, and executing code. A single-pin interface provides communication to the OCD.

Signal and Pin Descriptions

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

Available Packages

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

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

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

Pin Configurations

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

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

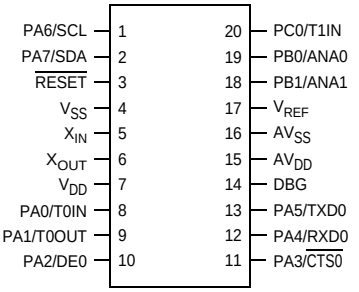


Figure 2. The Z8F0821 and Z8F0421 MCUs in 20-Pin SSOP and PDIP Packages

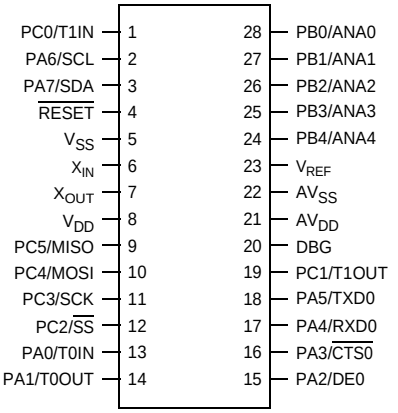


Figure 3. The Z8F0822 and Z8F0422 MCUs in 28-Pin SOIC and PDIP Packages

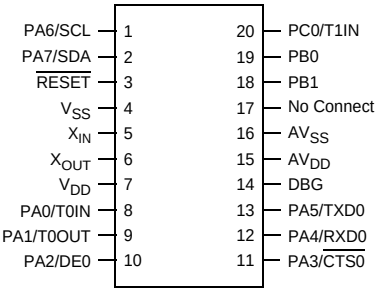


Figure 4. The Z8F0811 and Z8F0411 MCUs in 20-Pin SSOP and PDIP Packages

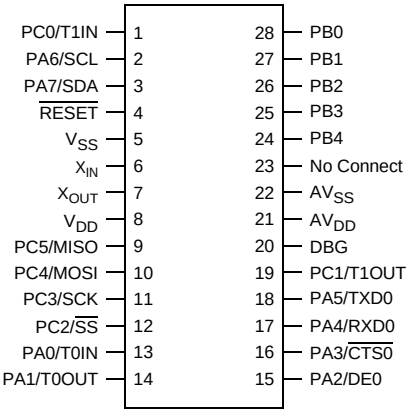


Figure 5. The Z8F0812 and Z8F0412 MCUs in 28-Pin SOIC and PDIP Packages

Signal Descriptions

Table 3 describes Z8 Encore! XP® F0822 Series signals. See the Pin Configurations section on page 7 to determine the signals available for the specific package styles

Table 3. Signal Descriptions

Signal Mnemonic	I/O	Description
General-Purpose I/O Ports A–H		
PA[7:0]	I/O	**Port C** These pins are used for general-purpose I/O and supports 5V-tolerant inputs.
PB[4:0]	I/O	**Port B** These pins are used for general-purpose I/O.
PC[5:0]	I/O	**Port C** These pins are used for general-purpose I/O and support 5V-tolerant inputs.
I^2C Controller		
SCL	I/O	**Serial Clock** This open-drain pin clocks data transfers in accordance with the I^2C standard protocol. This pin is multiplexed with a GPIO pin. When the GPIO pin is configured for alternate function to enable the SCL function, this pin is open-drain.
SDA	I/O	**Serial Data** This open-drain pin transfers data between the I^2C and a slave. This pin is multiplexed with a GPIO pin. When the GPIO pin is configured for alternate function to enable the SDA function, this pin is open-drain.
SPI Controller		
$\overline{SS}$	I/O	**Slave Select** This signal can be an output or an input. If the Z8 Encore! XP® F0822 Series MCU is the SPI Master, this pin can be configured as the Slave Select output. If the MCU is the SPI Slave, this pin is the input slave select. It is multiplexed with a GPIO pin.
SCK	I/O	**SPI Serial Clock** The SPI Master supplies this pin. If the Z8 Encore! XP® F0822 Series MCU is the SPI Master, this pin is the output. If the MCU is the SPI Slave, this pin is the input. It is multiplexed with a GPIO pin.
MOSI	I/O	**Master-Out/Slave-In** This signal is the data output from the SPI Master device and the data input to the SPI Slave device. It is multiplexed with a GPIO pin.
MISO	I/O	**Master-In/Slave-Out** This pin is the data input to the SPI Master device and the data output from the SPI Slave device. It is multiplexed with a GPIO pin.

Table 3. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
UART Controllers		
TXD0	O	**Transmit Data** This signal is the transmit output from the UART and IrDA. The TXD signals are multiplexed with GPIO pins.
RXD0	I	**Receive Data** This signal is the receiver input for the UART and IrDA. The RXD signals are multiplexed with GPIO pins.
$\overline{\text{CTS0}}$	I	**Clear To Send** This signal is control inputs for the UART. The $\overline{\text{CTS}}$ signals are multiplexed with GPIO pins.
DE0	O	**Driver Enable** This signal allows automatic control of external RS-485 drivers. This signal is approximately the inverse of the TXE (Transmit Empty) bit in the UART Status 0 Register. The DE signal can be used to ensure the external RS-485 driver is enabled when data is transmitted by the UART.
Timers		
T0OUT/ T1OUT	O	**Timer Output 0–1** These signals are output pins from the timers. The Timer Output signals are multiplexed with GPIO pins.
T0IN/T1IN	I	**Timer Input 0–1** These signals are used as the Capture, Gating and Counter inputs. The Timer Input signals are multiplexed with GPIO pins.
Analog		
ANA[4:0]	I	**Analog Input** These signals are inputs to the Analog-to-Digital Converter (ADC). The ADC analog inputs are multiplexed with GPIO pins.
V_{REF}	I	**Analog-to-Digital Converter Reference Voltage Input** As an output, Zilog does not recommend the V_{REF} signal for use as a reference voltage for external devices. If the ADC is configured to use the internal reference voltage generator, this pin should remain unconnected or capacitively coupled to analog ground (AV_{SS}).
Oscillators		
X_{IN}	I	**External Crystal Input** This pin is the input to the crystal oscillator. A crystal is connected between the external crystal input 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 to the system.

Table 3. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
X_{OUT}	O	**External Crystal Output** This pin is the output of the crystal oscillator. A crystal is connected between external crystal output and the X_{IN} pin to form the oscillator. When the system clock is referred in this manual, it refers to the frequency of the signal at this pin. This pin must remain unconnected when not using a crystal.
On-Chip Debugger		
DBG	I/O	**Debug** This pin is the control and data input and output to and from the OCD. The DBG pin is open-drain and must have an external pull-up resistor to ensure proper operation. **Caution:** For operation of the OCD, all power pins (V_{DD} and AV_{DD}) must be supplied with power and all ground pins (V_{SS} and AV_{SS}) must be properly grounded.
Reset		
$\overline{\text{RESET}}$	I	**RESET** Generates a Reset when asserted (driven Low).
Power Supply		
V_{DD}	I	Digital Power Supply.
AV_{DD}	I	**Analog Power Supply** Must be powered up and grounded to VDD, even if not using analog features.
V_{SS}	I	**Digital Ground.**
AV_{SS}	I	**Analog Ground** Must be grounded and connected to VSS, even if not using analog features.

Pin Characteristics

Table 4 provides detailed information about the characteristics for each pin available on Z8 Encore! XP® F0822 Series products. The data in Table 4 is sorted alphabetically by pin symbol mnemonic.

Table 4. Pin Characteristics

Symbol Mnemonic	Direction	Reset Direction	Active Low or Active High	Tri-State Output	Internal Pull-Up or Pull-Down	Schmitt-Trigger Input	Open Drain Output
AV_{DD}	N/A	N/A	N/A	N/A	No	No	N/A
AV_{SS}	N/A	N/A	N/A	N/A	No	No	N/A
DBG	I/O	I	N/A	Yes	No	Yes	Yes
PA[7:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable
PB[4:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable
PC[5:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable
RESET	I	I	Low	N/A	Pull-up	Yes	N/A
V_{DD}	N/A	N/A	N/A	N/A	No	No	N/A
V_{REF}	Analog	N/A	N/A	N/A	No	No	N/A
V_{SS}	N/A	N/A	N/A	N/A	No	No	N/A
X_{IN}	I	I	N/A	N/A	No	No	N/A
X_{OUT}	O	O	N/A	No	No	No	No

Address Space

The eZ8 CPU accesses three distinct address spaces:

- The Register File contains addresses for the general-purpose registers and the eZ8 CPU, Peripheral, and GPIO Port Control registers
- The Program Memory contains addresses for all memory locations having executable code and/or data
- The Data Memory contains addresses for all memory locations that hold data only

These three address spaces are covered briefly in the following sections. For more information about the eZ8 CPU and its address space, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download at www.zilog.com.

Register File

The Register File address space in the Z8 Encore! XP® F0822 Series is 4KB (4096 bytes). It is composed of two sections: Control registers and General-Purpose registers. When instructions are executed, registers are read from when defined as sources and written to when defined as destinations. The architecture of the eZ8 CPU allows all general-purpose registers to function as accumulators, address pointers, index registers, stack areas, or scratch pad memory.

The upper 256 bytes of the 1KB Register File address space is reserved for control of the eZ8 CPU, the on-chip peripherals, and the I/O ports. These registers are located at addresses from F00H to FFFH. Some of the addresses within the 256-byte Control Register section is reserved (unavailable). Reading from the reserved Register File addresses returns an undefined value. Writing to reserved Register File addresses is not recommended by Zilog because it can produce unpredictable results.

The on-chip RAM always begins at address 000H in the Register File address space. Z8 Encore! XP® F0822 Series contains 1KB of on-chip RAM. Reading from Register File addresses outside the available RAM addresses (and not within the control register address space) returns an undefined value. Writing to these Register File addresses produces no effect.

Program Memory

The eZ8 CPU supports 64KB of Program memory address space. Z8 Encore! XP® F0822 Series contain 4KB to 8KB on-chip Flash in the Program memory address space, depending on the device. Reading from Program memory addresses outside the available Flash addresses returns FFH. Writing to unimplemented Program memory addresses produces no effect. Table 5 describes the Program memory Maps for Z8 Encore! XP® F0822 Series devices.

Table 5. Z8 Encore! XP® F0822 Series Program Memory Maps

Program Memory Address (Hex)	Function
Z8F082x and Z8F081x Products	
0000-0001	Option Bits
0002-0003	Reset Vector
0004-0005	WDT Interrupt Vector
0006-0007	Illegal Instruction Trap
0008-0037	Interrupt Vectors*
0038-1FFF	Program Memory
Z8F042x and Z8F041x Products	
0000-0001	Option Bits
0002-0003	Reset Vector
0004-0005	WDT Interrupt Vector
0006-0007	Illegal Instruction Trap
0008-0037	Interrupt Vectors*
0038-0FFF	Program Memory

Note: *See Table 24 on page 41 for a list of the interrupt vectors.

Data Memory

Z8 Encore! XP® F0822 Series does not use the eZ8 CPU's 64KB Data Memory address space.

Information Area

Table 6 describes the Z8 Encore! XP® F0822 Series Information Area. This 512-byte Information Area is accessed by setting bit 7 of the Page Select Register to 1. When access is enabled, the Information Area is mapped into the Program memory and overlays the 512 bytes at addresses FE00H to FFFFH. When the Information Area access is enabled, all

reads from these Program memory addresses return the Information Area data rather than the Program memory data. Access to the Information Area is read-only.

Table 6. Information Area Map

Program Memory Address (Hex)	Function
FE00H–FE3FH	Reserved
FE40H–FE53H	Part Number: 20-character ASCII alphanumeric code, left-justified and filled with zeros
FE54H–FFFFH	Reserved

Register File Address Map

Table 7 provides the address map for the Register File of the Z8 Encore! XP® F0822 Series products. Not all devices and package styles in the F0822 Series support the ADC, the SPI, or all of the GPIO Ports. Consider registers for unimplemented peripherals to be reserved.

Table 7. Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
General Purpose RAM				
000–3FF	General-Purpose Register File RAM	—	XX	
400–EFF	Reserved	—	XX	
Timer 0				
F00	Timer 0 High Byte	T0H	00	64
F01	Timer 0 Low Byte	T0L	01	64
F02	Timer 0 Reload High Byte	T0RH	FF	65
F03	Timer 0 Reload Low Byte	T0RL	FF	65
F04	Timer 0 PWM High Byte	T0PWMH	00	66
F05	Timer 0 PWM Low Byte	T0PWML	00	66
F06	Timer 0 Control 0	T0CTL0	00	68
F07	Timer 0 Control 1	T0CTL1	00	68
Timer 1				
F08	Timer 1 High Byte	T1H	00	64
F09	Timer 1 Low Byte	T1L	01	64
F0A	Timer 1 Reload High Byte	T1RH	FF	65
F0B	Timer 1 Reload Low Byte	T1RL	FF	65
F0C	Timer 1 PWM High Byte	T1PWMH	00	66
F0D	Timer 1 PWM Low Byte	T1PWML	00	66
F0E	Timer 1 Control 0	T1CTL0	00	68
F0F	Timer 1 Control 1	T1CTL1	00	68
F10–F3F	Reserved	—	XX	

Note: XX = undefined.

Table 7. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
UART 0				
F40	UART0 Transmit Data	U0TXD	XX	88
	UART0 Receive Data	U0RXD	XX	89
F41	UART0 Status 0	U0STAT0	0000011Xb	90
F42	UART0 Control 0	U0CTL0	00	92
F43	UART0 Control 1	U0CTL1	00	92
F44	UART0 Status 1	U0STAT1	00	90
F45	UART0 Address Compare Register	U0ADDR	00	94
F46	UART0 Baud Rate High Byte	U0BRH	FF	95
F47	UART0 Baud Rate Low Byte	U0BRL	FF	95
F48–F4F	Reserved	—	XX	
I^2C				
F50	I^2C Data	I2CDATA	00	129
F51	I^2C Status	I2CSTAT	80	129
F52	I^2C Control	I2CCTL	00	131
F53	I^2C Baud Rate High Byte	I2CBRH	FF	132
F54	I^2C Baud Rate Low Byte	I2CBRL	FF	132
F55	I^2C Diagnostic State	I2CDST	XX000000b	133
F56	I^2C Diagnostic Control	I2CDIAG	00	135
F57–F5F	Reserved	—	XX	
Serial Peripheral Interface (SPI) Unavailable in 20-Pin Package Devices				
F60	SPI Data	SPIDATA	01	109
F61	SPI Control	SPICTL	00	110
F62	SPI Status	SPISTAT	00	111
F63	SPI Mode	SPIMODE	00	112
F64	SPI Diagnostic State	SPIDST	00	113
F65	Reserved	—	XX	
F66	SPI Baud Rate High Byte	SPIBRH	FF	114
F67	SPI Baud Rate Low Byte	SPIBRL	FF	114
F68–F6F	Reserved	—	XX	

Note: XX = undefined.

Table 7. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
Analog-to-Digital Converter (ADC)				
F70	ADC Control	ADCCTL	20	139
F71	Reserved	—	XX	
F72	ADC Data High Byte	ADCD_H	XX	141
F73	ADC Data Low Bits	ADCD_L	XX	142
F74–FBF	Reserved	—	XX	
Interrupt Controller				
FC0	Interrupt Request 0	IRQ0	00	45
FC1	IRQ0 Enable High Bit	IRQ0ENH	00	48
FC2	IRQ0 Enable Low Bit	IRQ0ENL	00	48
FC3	Interrupt Request 1	IRQ1	00	46
FC4	IRQ1 Enable High Bit	IRQ1ENH	00	49
FC5	IRQ1 Enable Low Bit	IRQ1ENL	00	49
FC6	Interrupt Request 2	IRQ2	00	47
FC7	IRQ2 Enable High Bit	IRQ2ENH	00	51
FC8	IRQ2 Enable Low Bit	IRQ2ENL	00	51
FC9–FCC	Reserved	—	XX	
FCD	Interrupt Edge Select	IRQES	00	52
FCE	Reserved	—	00	
FCF	Interrupt Control	IRQCTL	00	53
GPIO Port A				
FD0	Port A Address	PAADDR	00	32
FD1	Port A Control	PACTL	00	33
FD2	Port A Input Data	PAIN	XX	38
FD3	Port A Output Data	PAOUT	00	39
GPIO Port B				
FD4	Port B Address	PBADDR	00	32
FD5	Port B Control	PBCTL	00	33
FD6	Port B Input Data	PBIN	XX	38
FD7	Port B Output Data	PBOUT	00	39

Note: XX = undefined.

Table 7. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
GPIO Port C				
FD8	Port C Address	PCADDR	00	32
FD9	Port C Control	PCCTL	00	33
FDA	Port C Input Data	PCIN	XX	38
FDB	Port C Output Data	PCOUT	00	39
FDC-FEF	Reserved	—	XX	
Watchdog Timer (WDT)				
FF0	Watchdog Timer Control	WDTCTL	XXX00000b	73
FF1	Watchdog Timer Reload Upper Byte	WDTU	FF	75
FF2	Watchdog Timer Reload High Byte	WDTH	FF	75
FF3	Watchdog Timer Reload Low Byte	WDTL	FF	75
FF4-FF7	Reserved	—	XX	
Flash Memory Controller				
FF8	Flash Control	FCTL	00	150
FF8	Flash Status	FSTAT	00	151
FF9	Page Select	FPS	00	152
FF9 (if enabled)	Flash Sector Protect	FPROT	00	153
FFA	Flash Programming Frequency High Byte	FFREQH	00	153
FFB	Flash Programming Frequency Low Byte	FFREQL	00	153
Read-Only Memory				
FF8	Reserved	—	XX	
FF9	Page Select	RPS	00	152
FFA-FFB	Reserved	—	XX	
eZ8 CPU				
FFC	Flags	—	XX	Refer to the eZ8 CPU Core User Manual (UM0128)
FFD	Register Pointer	RP	XX	
FFE	Stack Pointer High Byte	SPH	XX	
FFF	Stack Pointer Low Byte	SPL	XX	

Note: XX = undefined.

Reset and Stop Mode Recovery

The Reset Controller within the Z8 Encore! XP® F0822 Series controls Reset and Stop Mode Recovery operation. In typical operation, the following events cause a Reset to occur:

- Power-On Reset (POR)
- Voltage Brown-Out
- WDT time-out (when configured through the WDT_RES option bit to initiate a Reset)
- External $\overline{\text{RESET}}$ pin assertion
- On-Chip Debugger initiated Reset (OCDCTL[0] set to 1)

When the Z8 Encore! XP® F0822 Series device is in STOP Mode, a Stop Mode Recovery is initiated by any of the following events:

- WDT time-out
- GPIO Port input pin transition on an enabled Stop Mode Recovery source
- DBG pin driven Low

Reset Types

Z8 Encore! XP® F0822 Series provides two types of reset operation (System Reset and Stop Mode Recovery). The type of reset is a function of both the current operating mode of the Z8 Encore! XP® F0822 Series device and the source of the Reset. Table 8 lists the types of Resets and their operating characteristics.

Table 8. Reset and Stop Mode Recovery Characteristics and Latency

	Reset Characteristics and Latency		
Reset Type	**Control Registers**	**eZ8 CPU**	**Reset Latency (Delay)**
System Reset	Reset (as applicable)	Reset	66 WDT Oscillator cycles + 16 System Clock cycles
Stop Mode Recovery	Unaffected, except for the WDT_CTL Register	Reset	66 WDT Oscillator cycles + 16 System Clock cycles

System Reset

During a System Reset, a Z8 Encore! XP® F0822 Series device is held in Reset for 66 cycles of the WDT oscillator followed by 16 cycles of the system clock. At the beginning

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

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

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

Reset Sources

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

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

Table 9. Reset Sources and Resulting Reset Type

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

Power-On Reset

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

voltage threshold (V_{POR}), the POR Counter is enabled and counts 66 cycles of the WDT oscillator. After the POR counter times out, the XTAL Counter is enabled to count a total of 16 system clock pulses. The device is held in the Reset state until both the POR Counter and XTAL counter have timed out. After the Z8 Encore! XP® F0822 Series device exits the POR state, the eZ8 CPU fetches the Reset vector. Following POR, the POR status bit in the Watchdog Timer Control Register (WDTCTL) is set to 1.

Figure 6 displays POR operation. See the Electrical Characteristics chapter on page 176 for the POR threshold voltage (V_{POR}).

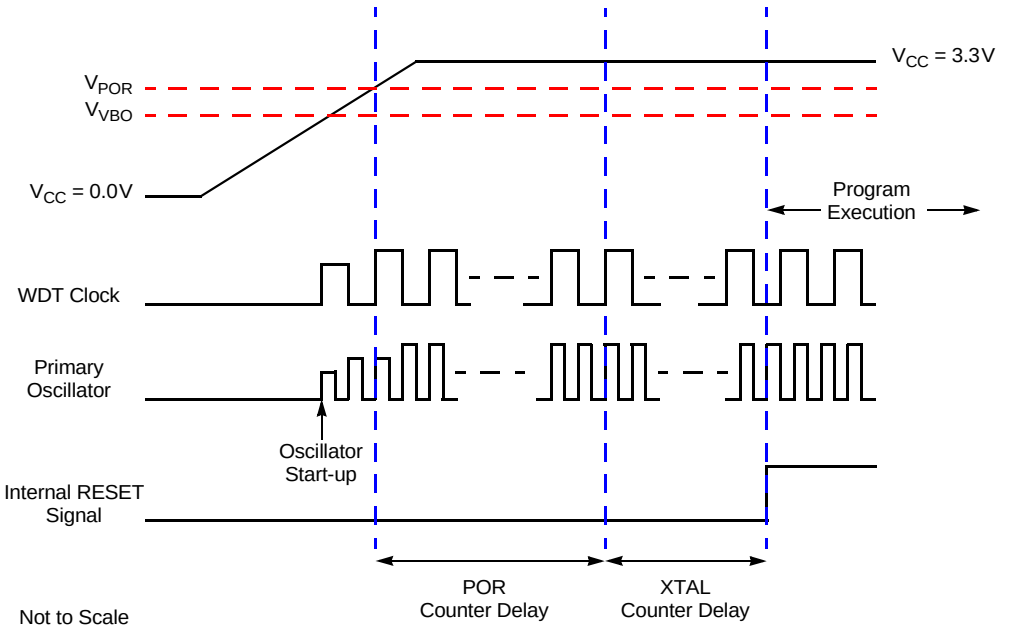


Figure 6. Power-On Reset Operation

Voltage Brown-Out Reset

The devices in Z8 Encore! XP® F0822 Series provides low-voltage brown-out protection. The VBO circuit senses when the supply voltage drops to an unsafe level (below the VBO threshold voltage) and forces the device into the Reset state. While the supply voltage remains below the POR voltage threshold (V_{POR}), the VBO block holds the device in the Reset state.

After the supply voltage again exceeds the POR voltage threshold, the device progresses through a full System Reset sequence as described in the POR section. Following POR,

the POR status bit in the Watchdog Timer Control Register (WDTCTL) is set to 1. Figure 7 displays the VBO operation. See the Electrical Characteristics chapter on page 176 for the VBO and POR threshold voltages (V_{VBO} and V_{POR}).

The VBO circuit can be either enabled or disabled during STOP Mode. Operation during STOP Mode is set by the VBO_AO option bit. For information about configuring VBO_AO, see the Option Bits chapter on page 155.

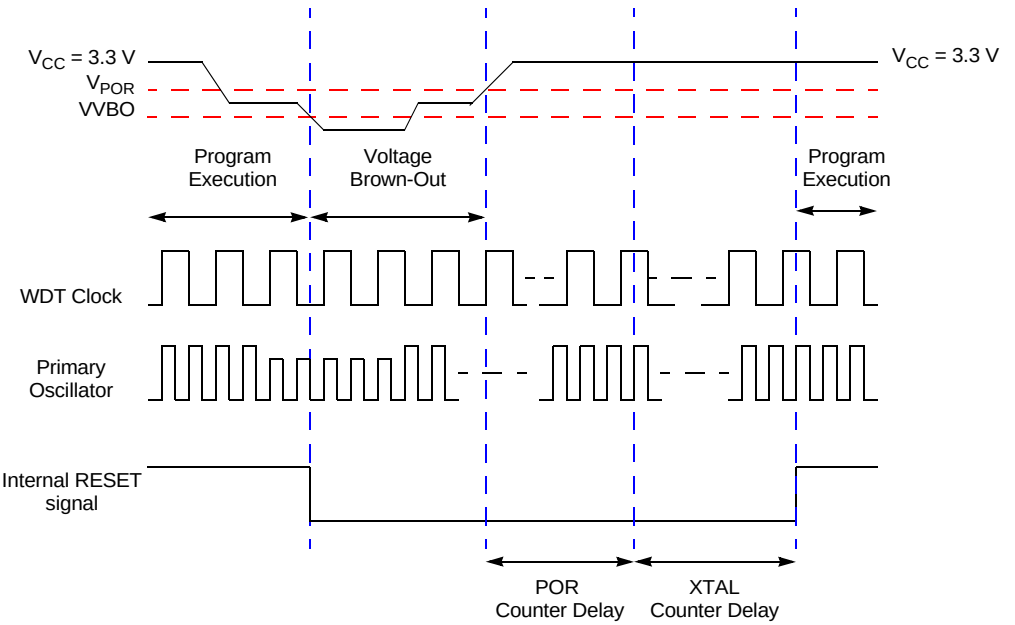


Figure 7. Voltage Brown-Out Reset Operation

Watchdog Timer Reset

If the device is in NORMAL or HALT Mode, WDT initiates a System Reset at time-out, if the WDT_RES option bit is set to 1. This setting is the default (unprogrammed) setting of the WDT_RES option bit. The WDT status bit in the WDT Control Register is set to signify that the reset was initiated by the WDT.

External Pin Reset

The $\overline{\text{RESET}}$ pin contains a Schmitt-triggered input, an internal pull-up, an analog filter, and a digital filter to reject noise. After the $\overline{\text{RESET}}$ pin is asserted for at least 4 system

clock cycles, the device progresses through the System Reset sequence. While the $\overline{\text{RESET}}$ input pin is asserted Low, Z8 Encore! XP® F0822 Series device continues to be held in the Reset state. If the $\overline{\text{RESET}}$ pin is held Low beyond the System Reset time-out, the device exits the Reset state immediately following $\overline{\text{RESET}}$ pin deassertion. Following a System Reset initiated by the external $\overline{\text{RESET}}$ pin, the EXT status bit in the Watchdog Timer Control Register (WDTCTL) is set to 1.

On-Chip Debugger Initiated Reset

A POR is initiated using the OCD by setting the RST bit in the OCD Control Register. The OCD block is not reset but the rest of the chip goes through a normal system reset. The RST bit automatically clears during the system reset. Following the system reset, the POR bit in the WDT Control Register is set.

Stop Mode Recovery

STOP Mode is entered by execution of a stop instruction by the eZ8 CPU. For detailed information about STOP Mode, see the Low-Power Modes chapter on page 27. During Stop Mode Recovery, the device is held in reset for 66 cycles of the WDT oscillator followed by 16 cycles of the system clock. Stop Mode Recovery only affects the contents of the WDT Control Register and does not affect any other values in the Register File including the Stack Pointer, Register Pointer, Flags, Peripheral Control registers, and General-Purpose RAM.

The eZ8 CPU fetches the Reset vector at Program memory addresses 0002H and 0003H and loads that value into the Program Counter. Program execution begins at the Reset vector address. Following Stop Mode Recovery, the stop bit in the WDT Control Register is set to 1. Table 10 lists the Stop Mode Recovery sources and resulting actions. The text following provides more detailed information about each of the Stop Mode Recovery sources.

Table 10. Stop Mode Recovery Sources and Resulting Action

Operating Mode	Stop Mode Recovery Source	Action
STOP Mode	WDT time-out when configured for Reset	Stop Mode Recovery
	WDT time-out when configured for interrupt	Stop Mode Recovery followed by interrupt (if interrupts are enabled)
	Data transition on any GPIO Port pin enabled as a Stop Mode Recovery source	Stop Mode Recovery

Stop Mode Recovery Using WDT Time-Out

If the WDT times out during STOP Mode, the device undergoes a Stop Mode Recovery sequence. In the WDT Control Register, the WDT and stop bits are set to 1. If the WDT is configured to generate an interrupt upon time-out and the Z8 Encore! XP® F0822 Series device is configured to respond to interrupts, the eZ8 CPU services the WDT interrupt request following the normal Stop Mode Recovery sequence.

Stop Mode Recovery Using a GPIO Port Pin Transition

Each of the GPIO port pins can be configured as a Stop Mode Recovery input source. On any GPIO pin enabled as a STOP Mode Recover source, a change in the input pin value (from High to Low or from Low to High) initiates Stop Mode Recovery. The GPIO Stop Mode Recovery signals are filtered to reject pulses less than 10ns (typical) in duration. In the WDT Control Register, the stop bit is set to 1.

Caution: In STOP Mode, the GPIO Port Input Data registers (PxIN) are disabled. The Port Input Data registers record the Port transition only if the signal stays on the port pin through the end of the Stop Mode Recovery delay. Therefore, short pulses on the port pin initiates Stop Mode Recovery without being written to the Port Input Data Register or without initiating an interrupt (if enabled for that pin).

Low-Power Modes

Z8 Encore! XP® F0822 Series products contain power-saving features. The highest level of power reduction is provided by STOP Mode. The next level of power reduction is provided by the HALT Mode.

STOP Mode

Execution of the eZ8 CPU's stop instruction places the device into STOP Mode. In STOP Mode, the operating characteristics are:

- Primary crystal oscillator is stopped; the X_{IN} pin is driven High and the X_{OUT} pin is driven Low
- System clock is stopped
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- If enabled for operation in STOP Mode, the WDT and its internal RC oscillator continue to operate
- If enabled for operation in STOP Mode through the associated option bit, the VBO protection circuit continues to operate
- All other on-chip peripherals are idle

To minimize current in STOP Mode, WDT must be disabled and all GPIO pins configured as digital inputs must be driven to one of the supply rails (V_{CC} or GND). The device can be brought out of STOP Mode using Stop Mode Recovery. For more information about Stop Mode Recovery, see the Reset and Stop Mode Recovery chapter on page 21.

Caution: STOP Mode must not be used when driving the Z8F082x family devices with an external clock driver source.

HALT Mode

Execution of the eZ8 CPU's HALT instruction places the device into HALT Mode. In HALT Mode, the operating characteristics are:

- Primary crystal oscillator is enabled and continues to operate

- System clock is enabled and continues to operate
- eZ8 CPU is stopped
- Program counter stops incrementing
- WDT's internal RC oscillator continues to operate
- If enabled, the WDT continues to operate
- All other on-chip peripherals continue to operate

The eZ8 CPU can be brought out of HALT Mode by any of the following operations:

- Interrupt
- WDT time-out (interrupt or reset)
- Power-On Reset
- Voltage Brown-Out reset
- External $\overline{\text{RESET}}$ pin assertion

To minimize current in HALT Mode, all GPIO pins which are configured as inputs must be driven to one of the supply rails (V_{CC} or GND).

General-Purpose Input/Output

Z8 Encore! XP® F0822 Series products support a maximum of 19 Port A–C pins for General-Purpose Input/Output (GPIO) operations. Each port consists Control and Data registers. The GPIO Control registers are used to determine data direction, open-drain, output drive current, programmable pull-ups, Stop Mode Recovery functionality, and alternate pin functions. Each port pin is individually programmable. Ports A and C support 5 V-tolerant inputs.

GPIO Port Availability by Device

Table 11 lists the port pins available with each device and package type.

Table 11. Port Availability by Device and Package Type

Devices	Package	Port A	Port B	Port C
Z8X0821, Z8X0811, Z8X0421, Z8X0411	20-pin	[7:0]	[1:0]	[0]
Z8X0822, Z8X0812, Z8X0422, Z8X0412	28-pin	[7:0]	[4:0]	[5:0]

Architecture

Figure 8 displays a simplified block diagram of a GPIO port pin. It does not display the ability to accommodate alternate functions, variable port current drive strength, and programmable pull-up.

GPIO Alternate Functions

Many of the GPIO port pins are used as both general-purpose I/O and to provide access to on-chip peripheral functions such as timers and serial communication devices. The Port A–C Alternate Function subregisters configure these pins for either general-purpose I/O or alternate function operation. When a pin is configured for alternate function, control of the port-pin direction (input/output) is passed from the Port A–C Data Direction registers to the alternate function assigned to this pin. Table 12 lists the alternate functions associated with each port pin.

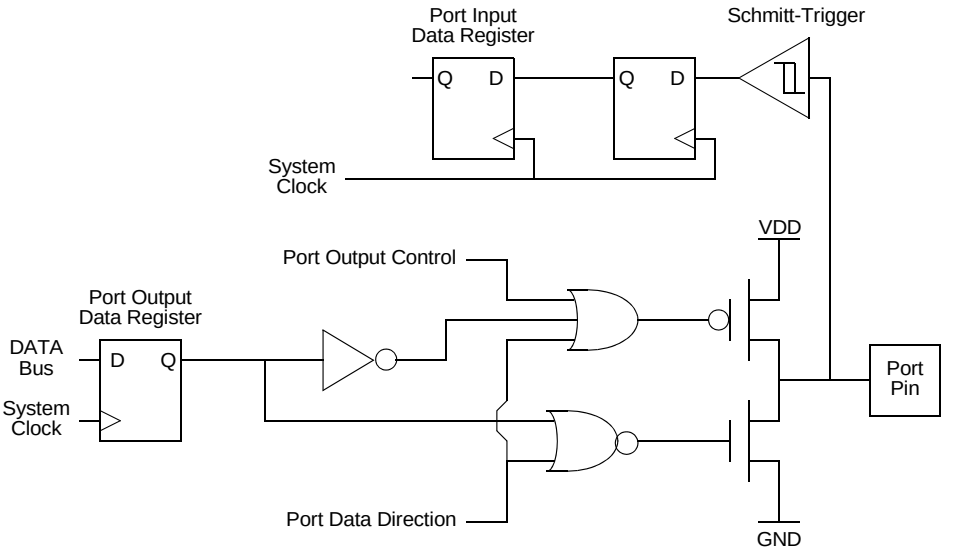


Figure 8. GPIO Port Pin Block Diagram

Table 12. Port Alternate Function Mapping

Port	Pin	Mnemonic	Alternate Function Description
Port A	PA0	T0IN	Timer 0 Input
	PA1	T0OUT	Timer 0 Output
	PA2	DE	UART 0 Driver Enable
	PA3	$\overline{\text{CTS0}}$	$\overline{\text{UART 0 Clear to Send}}$
	PA4	RXD0/IRRX0	UART 0/IrDA 0 Receive Data
	PA5	TXD0/IRTX0	UART 0/IrDA 0 Transmit Data
	PA6	SCL	I^2C Clock (automatically open-drain)
	PA7	SDA	I^2C Data (automatically open-drain)
Port B	PB0	ANA0	ADC Analog Input 0
	PB1	ANA1	ADC Analog Input 1
	PB2	ANA2	ADC Analog Input 2
	PB3	ANA3	ADC Analog Input 3
	PB4	ANA4	ADC Analog Input 4

Table 12. Port Alternate Function Mapping (Continued)

Port	Pin	Mnemonic	Alternate Function Description
Port C	PC0	T1IN	Timer 1 Input
	PC1	T1OUT	Timer 1 Output
	PC2	$\overline{SS}$	SPI Slave Select
	PC3	SCK	SPI Serial Clock
	PC4	MOSI	SPI Master Out Slave In
	PC5	MISO	SPI Master In Slave Out

GPIO Interrupts

Many of GPIO port pins are used as interrupt sources. Some port pins are configured to generate an interrupt request on either the rising edge or falling edge of the pin input signal. Other port pin interrupts generate an interrupt when any edge occurs (both rising and falling). For more details about interrupts using the GPIO pins, see Figure 8.

GPIO Control Register Definitions

Four registers for each port provide access to GPIO control, input data, and output data. Table 13 lists the GPIO port registers and subregisters. Use the Port A–C Address and Control registers together to provide access to subregisters for Port configuration and control.

Table 13. GPIO Port Registers and Subregisters

Port Register Mnemonic	Port Register Name
P*x*ADDR	Port A–C Address Register (selects subregisters)
P*x*CTL	Port A–C Control Register (provides access to subregisters)
P*x*IN	Port A–C Input Data Register
P*x*OUT	Port A–C Output Data Register
Port Subregister Mnemonic	**Port Register Name**
P*x*DD	Data Direction
P*x*AF	Alternate Function
P*x*OC	Output Control (Open-Drain)
P*x*HDE	High Drive Enable
P*x*SMRE	Stop Mode Recovery Source Enable
P*x*PUE	Pull-up Enable

Port A–C Address Registers

The Port A–C Address registers, shown in Table 14, 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 control.

Table 14. Port A–C GPIO Address Registers (PxADDR)

Bit	7	6	5	4	3	2	1	0
Field	PADDR[7:0]							
RESET	00H							
R/W	R/W							
Address	FD0H, FD4H, FD8H							

Bit	Description
[7:0] PADDR	**Port Address** The Port Address selects one of the subregisters accessible through the Port Control Register. 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–FFH = no function.

Port A–C Control Registers

The Port A–C Control registers, shown in Table 15, set the GPIO port operation. The value in the corresponding Port A–C Address Register determines the control subregisters accessible using the Port A–C Control Register.

Table 15. Port A–C Control Registers (PxCTL)

Bit	7	6	5	4	3	2	1	0
Field	PCTL							
RESET	00H							
R/W	R/W							
Address	FD1H, FD5H, FD9H							

Bit	Description
[7:0] PCTL	**Port Control** The Port Control Register provides access to all subregisters that configure the GPIO port operation.

Port A–C Data Direction Subregisters

The Port A–C Data Direction Subregister, shown in Table 16, is accessed through the Port A–C Control Register by writing 01H to the Port A–C Address Register.

Table 16. Port A–C Data Direction Subregisters

Bit	7	6	5	4	3	2	1	0
Field	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
RESET	1							
R/W	R/W							
Address	See footnote.							

Note: If 01H is written to the Port A–C Address Register, then it is accessible via the Port A–C Control Register.

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

Note: x indicates register bits in the range [7:0].

Port A–C Alternate Function Subregisters

The Port A–C Alternate Function Subregister, shown in Table 17, is accessed through the Port A–C Control Register by writing 02H to the Port A–C Address Register. The Port A–C Alternate Function subregisters select the alternate functions for the selected pins.To determine the alternate function associated with each port pin, see Figure 8 on page 30.

Caution: Do not enable alternate functions for GPIO port pins which do not have an associated alternate function. Failure to follow this guideline can result in unpredictable operation.

Table 17. Port A–CA–C Alternate Function Subregisters

Bit	7	6	5	4	3	2	1	0
Field	AF7	AF6	AF5	AF4	AF3	AF2	AF1	AF0
RESET	0							
R/W	R/W							
Address	See footnote.							

Note: If 02H is written to the Port A–C Address Register, then it is accessible via the Port A–C Control Register.

Bit	Description
[7:0] AFx	**Port Alternate Function enabled** 0 = The port pin is in NORMAL Mode and the DDx bit in the Port A–C Data Direction Subregister determines the direction of the pin. 1 = The alternate function is selected. Port pin operation is controlled by the alternate function.

Note: x indicates register bits in the range [7:0].

Port A–C Output Control Subregisters

The Port A–C Output Control Subregister, shown in Table 18, is accessed through the Port A–C Control Register by writing 03H to the Port A–C Address Register. Setting the bits in the Port A–C Output Control subregisters to 1 configures the specified port pins for open-drain operation. These subregisters affect the pins directly and, as a result, alternate functions are also affected.

Table 18. Port A–C Output Control Subregisters

Bit	7	6	5	4	3	2	1	0
Field	POC7	POC6	POC5	POC4	POC3	POC2	POC1	POC0
RESET	0							
R/W	R/W							
Address	See footnote.							

Note: If 03H is written to the Port A–C Address Register, then it is accessible via the Port A–C Control Register.

Bit	Description
[7:0] POCx	**Port Output Control** These bits function independently of the alternate function bit and always disable the drains if set to 1. 0 = The drains are enabled for any output mode (unless overridden by the alternate function). 1 = The drain of the associated pin is disabled (open-drain mode).

Note: x indicates register bits in the range [7:0].

Port A–C High Drive Enable Subregisters

The Port A–C High Drive Enable Subregister, shown in Table 19, is accessed through the Port A–C Control Register by writing 04H to the Port A–C Address Register. Setting the bits in the Port A–C High Drive Enable subregisters to 1 configures the specified port pins for high-output current drive operation. The Port A–C High Drive Enable Subregister affects the pins directly and, as a result, alternate functions are also affected.

Table 19. Port A–C High Drive Enable Subregisters

Bit	7	6	5	4	3	2	1	0
Field	PHDE7	PHDE6	PHDE5	PHDE4	PHDE3	PHDE2	PHDE1	PHDE0
RESET	0							
R/W	R/W							
Address	See footnote.							

Note: If 04H is written to the Port A–C Address Register, then it is accessible via the Port A–C Control Register.

Bit	Description
[7:0] PHDEx	**Port High Drive Enabled** 0 = The port pin is configured for standard-output current drive. 1 = The port pin is configured for high-output current drive.

Note: x indicates register bits in the range [7:0].

Port A–C Stop Mode Recovery Source Enable Subregisters

The Port A–C Stop Mode Recovery Source Enable Subregister, shown in Table 20, is accessed through the Port A–C Control Register by writing 05H to the Port A–C Address Register. Setting the bits in the Port A–C Stop Mode Recovery Source Enable subregisters to 1 configures the specified Port pins as a Stop Mode Recovery source. During STOP Mode, any logic transition on a Port pin enabled as a Stop Mode Recovery source initiates Stop Mode Recovery.

Table 20. Port A–C Stop Mode Recovery Source Enable Subregisters

Bit	7	6	5	4	3	2	1	0
Field	PSMRE7	PSMRE6	PSMRE5	PSMRE4	PSMRE3	PSMRE2	PSMRE1	PSMRE0
RESET	0							
R/W	R/W							
Address	See footnote.							

Note: If 05H is written to the Port A–C Address Register, then it is accessible through the Port A–C Control Register.

Bit	Description
[7:0] PSMREx	**Port Stop Mode Recovery Source Enabled** 0 = The port pin is not configured as a Stop Mode Recovery source. Transitions on this pin during STOP Mode does not initiate Stop Mode Recovery. 1 = The port pin is configured as a Stop Mode Recovery source. Any logic transition on this pin during STOP Mode initiates Stop Mode Recovery.

Note: x indicates register bits in the range [7:0].

Port A–C Pull-up Enable Subregisters

The Port A–C Pull-Up Enable Subregister, shown in Table 21, is accessed through the Port A–C Control Register by writing 06H to the Port A–C Address Register. Setting the bits in the Port A–C Pull-Up Enable subregisters enables a weak internal resistive pull-up on the specified Port pins.

Table 21. Port A–C Pull-Up Enable Subregisters

Bit	7	6	5	4	3	2	1	0
Field	PPUE7	PPUE6	PPUE5	PPUE4	PPUE3	PPUE2	PPUE1	PPUE0
RESET	0							
R/W	R/W							
Address	See footnote.							

Note: If 06H is written to the Port A–C Address Register, then it is accessible through the Port A–C Control Register.

Bit	Description
[7:0] PPUEx	**Port Pull-up Enabled** 0 = The weak pull-up on the port pin is disabled. 1 = The weak pull-up on the port pin is enabled.

Note: x indicates register bits in the range [7:0].

Port A–C Input Data Registers

Reading from the Port A–C Input Data registers, shown in Table 22, returns the sampled values from the corresponding port pins. The Port A–C Input Data registers are read-only.

Table 22. Port A–C Input Data Registers (PxIN)

Bit	7	6	5	4	3	2	1	0
Field	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X							
R/W	R							
Address	FD2H, FD6H, FDAH							

Bit	Description
[7:0] PxIN	**Port Input Data** Sampled data from the corresponding port pin input. 0 = Input data is logical 0 (Low). 1 = Input data is logical 1 (High).

Note: x indicates register bits in the range [7:0].

Port A–C Output Data Register

The Port A–C Output Data Register, shown in Table 23, controls the output data to the pins.

Table 23. Port A–C Output Data Register (PxOUT)

Bit	7	6	5	4	3	2	1	0
Field	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0							
R/W	R/W							
Address	FD3H, FD7H, FDBH							

Bit	Description
[7:0] PxOUT	**Port Output Data** These bits contain the data to be driven to the port pins. The values are only driven if the corresponding pin is configured as an output and the pin is not configured for alternate function operation. 0 = Drive a logical 0 (Low). 1 = Drive a logical 1 (High). This High value is not driven if the drain has been disabled by setting the corresponding Port Output Control Register bit to 1.

Note: x indicates register bits in the range [7:0].

Interrupt Controller

The interrupt controller on Z8 Encore! XP® F0822 Series products prioritizes the interrupt requests from the on-chip peripherals and the GPIO port pins. The features of the interrupt controller include the following:

- 19 unique interrupt vectors:
  - 12 GPIO port pin interrupt sources
  - 7 On-chip peripheral interrupt sources
- Flexible GPIO interrupts:
  - 8 selectable rising and falling edge GPIO interrupts
  - 4 dual-edge interrupts
- Three levels of individually programmable interrupt priority
- WDT is configured to generate an interrupt

Interrupt Requests (IRQs) allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start an Interrupt Service Routine (ISR). Usually this ISR is involved with the exchange of data, status information, or control information between the CPU and the interrupting peripheral. When the service routine is completed, the CPU returns to the operation from which it was interrupted.

The eZ8 CPU supports both vectored and polled interrupt handling. For polled interrupts, the interrupt control has no effect on operation. For more information about interrupt servicing, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download at www.zilog.com.

Interrupt Vector Listing

Table 24 lists all of the interrupts available in order of priority. The interrupt vector is stored with the most significant byte (MSB) at the even program memory address and the least significant byte (LSB) at the following odd program memory address.

Table 24. Interrupt Vectors in Order of Priority

Priority	Program Memory Vector Address	Interrupt Source
Highest	0002H	Reset (not an interrupt)
	0004H	WDT (see the Watchdog Timer chapter on page 70)
	0006H	Illegal Instruction Trap (not an interrupt)
	0008H	Reserved
	000AH	Timer 1
	000CH	Timer 0
	000EH	UART 0 receiver
	0010H	UART 0 transmitter
	0012H	I^2C
	0014H	SPI
	0016H	ADC
	0018H	Port A7, rising or falling input edge
	001AH	Port A6, rising or falling input edge
	001CH	Port A5, rising or falling input edge
	001EH	Port A4, rising or falling input edge
	0020H	Port A3, rising or falling input edge
	0022H	Port A2, rising or falling input edge
	0024H	Port A1, rising or falling input edge
	0026H	Port A0, rising or falling input edge
	0028H	Reserved
	002AH	Reserved
	002CH	Reserved
	002EH	Reserved
	0030H	Port C3, both input edges
	0032H	Port C2, both input edges
	0034H	Port C1, both input edges
Lowest	0036H	Port C0, both input edges

Architecture

Figure 9 displays a block diagram of the interrupt controller.

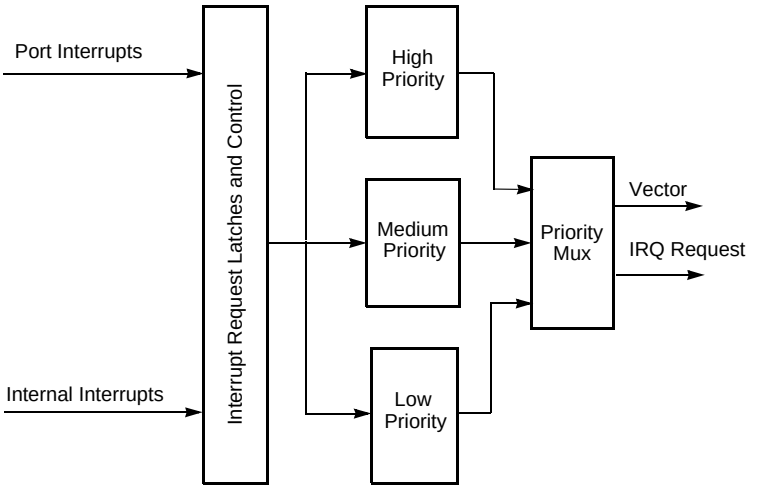


Figure 9. Interrupt Controller Block Diagram

Operation

This section describes the operational aspects of the following functions.

Master Interrupt Enable: see page 42

Interrupt Vectors and Priority: see page 43

Interrupt Assertion: see page 43

Software Interrupt Assertion: see page 44

Master Interrupt Enable

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

Interrupts are globally enabled by any of the following actions:

- Execution of an Enable Interrupt (EI) instruction
- Execution of an Return from Interrupt (IRET) instruction

- 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

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 were enabled with identical interrupt priority (all as Level 2 interrupts), then interrupt priority would be assigned from highest to lowest as specified in Table 24. Level 3 interrupts always have higher priority than Level 2 interrupts which in turn always have 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 24. A Reset, WDT interrupt (if enabled), and an Illegal Instruction Trap will always have the highest 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.

Example 1. A poor coding style that can result in lost interrupt requests:

```
LDX r0, IRQ0
AND r0, MASK
LDX IRQ0, r0
```

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

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

```
ANDX IRQ0, MASK
```

Software Interrupt Assertion

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

Caution: Zilog recommends not using a coding style to generate software interrupts by setting bits in the Interrupt Request registers. All incoming interrupts received between execution of the first LDX command and the final LDX command are lost. See Example 3, which follows.

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

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

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

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

```
ORX IRQ0, MASK
```

Interrupt Control Register Definitions

For all interrupts other than the WDT interrupt, the Interrupt Control registers enable individual interrupts, set interrupt priorities, and indicate interrupt requests.

Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) Register, shown in Table 25, stores the interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ0 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 IRQ0 Register to determine if any interrupt requests are pending.

Table 25. Interrupt Request 0 Register (IRQ0)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	T1I	T0I	U0RXI	U0TXI	I2CI	SPII	ADCI
RESET	0							
R/W	R/W							
Address	FC0H							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] T1I	**Timer 1 Interrupt Request** 0 = No interrupt request is pending for Timer 1. 1 = An interrupt request from Timer 1 is awaiting service.
[5] T0I	**Timer 0 Interrupt Request** 0 = No interrupt request is pending for Timer 0. 1 = An interrupt request from Timer 0 is awaiting service.
[4] 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.
[3] 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.
[2] I2CI	**I^2C Interrupt Request** 0 = No interrupt request is pending for the I^2C. 1 = An interrupt request from the I^2C is awaiting service.

Bit	Description (Continued)
[1] SPII	**SPI Interrupt Request** 0 = No interrupt request is pending for the SPI. 1 = An interrupt request from the SPI is awaiting service.
[0] ADCI	**ADC Interrupt Request** 0 = No interrupt request is pending for the ADC. 1 = An interrupt request from the ADC is awaiting service.

Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) Register, shown in Table 26, 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 IRQ1 Register to determine if any interrupt requests are pending.

Table 26. Interrupt Request 1 Register (IRQ1)

Bit	7	6	5	4	3	2	1	0
Field	PA7I	PA6I	PA5I	PA4I	PA3I	PA2I	PA1I	PA0I
RESET	0							
R/W	R/W							
Address	FC3H							

Bit	Description
[7:0] 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.

Note: x indicates register bits in the range [7:0].

Interrupt Request 2 Register

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

Table 27. Interrupt Request 2 Register (IRQ2)

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

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

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

IRQ0 Enable High and Low Bit Registers

Table 28 describes the priority control for IRQ0. The IRQ0 Enable High and Low Bit registers, shown in Tables 29 and 30, form a priority-encoded enabling for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register.

Table 28. IRQ0 Enable and Priority Encoding

IRQ0ENH[*x*]	IRQ0ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

Note: *x* indicates register bits in the range [7:0].

Table 29. IRQ0 Enable High Bit Register (IRQ0ENH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	T1ENH	T0ENH	U0RENH	U0TENH	I2CENH	SPIENH	ADCENH
RESET	0							
R/W	R/W							
Address	FC1H							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] T1ENH	**Timer 1 Interrupt Request Enable High Bit**
[5] T0ENH	**Timer 0 Interrupt Request Enable High Bit**
[4] U0RENH	**UART 0 Receive Interrupt Request Enable High Bit**
[3] U0TENH	**UART 0 Transmit Interrupt Request Enable High Bit**
[2] I2CENH	**I^2C Interrupt Request Enable High Bit**
[1] SPIENH	**SPI Interrupt Request Enable High Bit**
[0] ADCENH	**ADC Interrupt Request Enable High Bit**

Table 30. IRQ0 Enable Low Bit Register (IRQ0ENL)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	T1ENL	T0ENL	U0RENL	U0TENL	I2CENL	SPIENL	ADCENL
RESET	0							
R/W	R/W							
Address	FC2H							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] T1ENL	**Timer 1 Interrupt Request Enable Low Bit**
[5] T0ENL	**Timer 0 Interrupt Request Enable Low Bit**
[4] U0RENL	**UART 0 Receive Interrupt Request Enable Low Bit**
[3] U0TENL	**UART 0 Transmit Interrupt Request Enable Low Bit**
[2] I2CENL	**I^2C Interrupt Request Enable Low Bit**
[1] SPIENL	**SPI Interrupt Request Enable Low Bit**
[0] ADCENL	**ADC Interrupt Request Enable Low Bit**

IRQ1 Enable High and Low Bit Registers

Table 31 describes the priority control for IRQ1. The IRQ1 Enable High and Low Bit registers, shown in Tables 32 and 33, form a priority-encoded enabling for interrupts in the Interrupt Request 1 Register. Priority is generated by setting bits in each register.

Table 31. IRQ1 Enable and Priority Encoding

IRQ1ENH[*x*]	IRQ1ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

Note: *x* indicates register bits in the range [7:0].

Table 32. IRQ1 Enable High Bit Register (IRQ1ENH)

Bit	7	6	5	4	3	2	1	0
Field	PA7ENH	PA6ENH	PA5ENH	PA4ENH	PA3ENH	PA2ENH	PA1ENH	PA0ENH
RESET	0							
R/W	R/W							
Address	FC4H							

Bit	Description
[7:0] PAxENH	**Port A Bit[*x*] Interrupt Request Enable High Bit**

Note: *x* indicates register bits in the range [7:0].

Table 33. IRQ1 Enable Low Bit Register (IRQ1ENL)

Bit	7	6	5	4	3	2	1	0
Field	PA7ENL	PA6ENL	PA5ENL	PA4ENL	PA3ENL	PA2ENL	PA1ENL	PA0ENL
RESET	0							
R/W	R/W							
Address	FC5H							

Bit	Description
[7:0] PAxENL	**Port A Bit[*x*] Interrupt Request Enable Low Bit**

Note: *x* indicates register bits in the range [7:0].

IRQ2 Enable High and Low Bit Registers

Table 34 describes the priority control for IRQ2. The IRQ2 Enable High and Low Bit registers, shown in Tables 35 and 36, form a priority-encoded enabling for interrupts in the Interrupt Request 2 Register. Priority is generated by setting bits in each register.

Table 34. IRQ2 Enable and Priority Encoding

IRQ2ENH[*x*]	IRQ2ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

Note: *x* indicates register bits in the range [7:0].

Table 35. IRQ2 Enable High Bit Register (IRQ2ENH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				C3ENH	C2ENH	C1ENH	C0ENH
RESET	0							
R/W	R/W							
Address	FC7H							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] C3ENH	**Port C3 Interrupt Request Enable High Bit**
[2] C2ENH	**Port C2 Interrupt Request Enable High Bit**
[1] C1ENH	**Port C1 Interrupt Request Enable High Bit**
[0] C0ENH	**Port C0 Interrupt Request Enable High Bit**

Table 36. IRQ2 Enable Low Bit Register (IRQ2ENL)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				C3ENL	C2ENL	C1ENL	C0ENL
RESET	0							
R/W	R/W							
Address	FC8H							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] C3ENL	**Port C3 Interrupt Request Enable Low Bit**
[2] C2ENL	**Port C2 Interrupt Request Enable Low Bit**
[1] C1ENL	**Port C1 Interrupt Request Enable Low Bit**
[0] C0ENL	**Port C0 Interrupt Request Enable Low Bit**

Interrupt Edge Select Register

The Interrupt Edge Select (IRQES) Register, shown in Table 37, determines whether an interrupt is generated for the rising edge or falling edge on the selected GPIO Port input pin. The minimum pulse width must be greater than 1 system clock to guarantee capture of the edge triggered interrupt. Edge detection for pulses less than 1 system clock are not guaranteed.

Table 37. Interrupt Edge Select Register (IRQES)

Bit	7	6	5	4	3	2	1	0
Field	IES7	IES6	IES5	IES4	IES3	IES2	IES1	IES0
RESET	0							
R/W	R/W							
Address	FCDH							

Bit	Description
[7:0] IESx	**Interrupt Edge Select *x*** 0 = An interrupt request is generated on the falling edge of the PA*x* input. 1 = An interrupt request is generated on the rising edge of the PA*x* input.

Note: *x* indicates register bits in the range [7:0].

Interrupt Control Register

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

Table 38. Interrupt Control Register (IRQCTL)

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

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

Timers

Z8 Encore! XP® F0822 Series products contain up to two 16-bit reloadable timers that can be used for timing, event counting, or generation of pulse-width modulated signals. The timer features include:

- 16-bit reload counter
- Programmable prescaler with prescale values from 1 to 128
- PWM output generation
- Capture and compare capability
- External input pin for timer input, clock gating, or capture signal; external input pin signal frequency is limited to a maximum of one-fourth the system clock frequency
- Timer output pin
- Timer interrupt

In addition to the timers described in this chapter, the Baud Rate Generators for any unused UART, SPI, or I^2C peripherals can also be used to provide basic timing functionality. See the respective serial communication peripheral chapters for information about using the Baud Rate Generators as timers.

Architecture

Figure 10 displays the architecture of the timers.

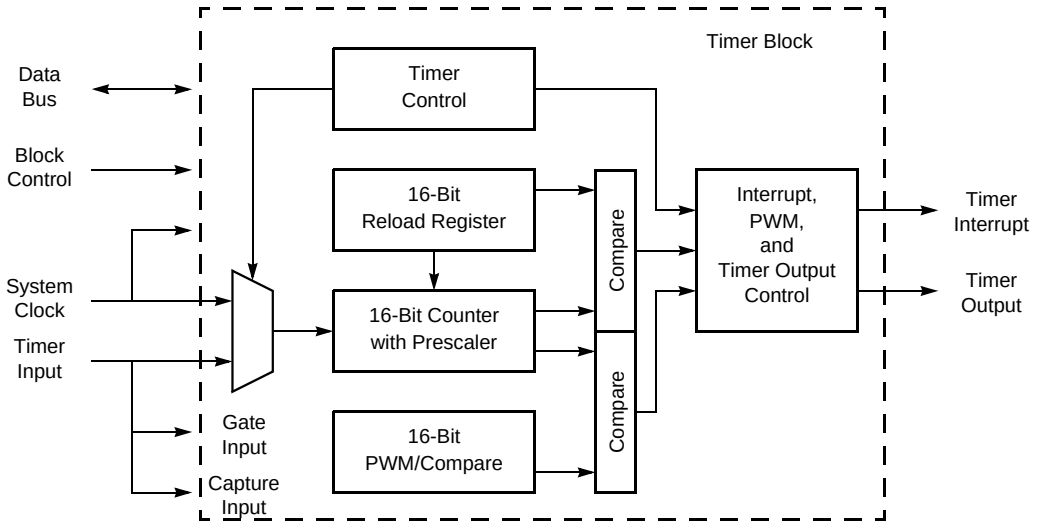


Figure 10. Timer Block Diagram

Operation

The timers are 16-bit up-counters. Minimum time-out delay is set by loading the value 0001H into the Timer Reload High and Low Byte registers and setting the prescale value to 1. Maximum time-out delay is set by loading the value 0000H into the Timer Reload High and Low Byte registers and setting the prescale value to 128. If the Timer reaches FFFFH, the timer rolls over to 0000H and continues counting.

Timer Operating Modes

The timers are configured to operate in the following modes:

ONE-SHOT Mode

In ONE-SHOT Mode, 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. Upon reaching the reload value, the timer generates an interrupt and the count value in the Timer High

and Low Byte registers is reset to 0001H. Then, the timer is automatically disabled and stops counting.

Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state for one system clock cycle (from Low to High or vice-versa) on timer reload. If it is required for the Timer Output to make a permanent state change on One-Shot time-out, first set the TPOL bit in the Timer Control Register to the start value before beginning ONE-SHOT Mode. Then, after starting the timer, set TPOL to the opposite bit value.

Observe the following procedure for configuring a timer for ONE-SHOT Mode and initiating the count:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for ONE-SHOT Mode
   - Set the prescale value
   - If using the Timer Output alternate function, set the initial output level (High or Low)
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 reload value.
4. If appropriate, enable the timer interrupt 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 ONE-SHOT Mode, the system clock always provides the timer input. The timer period is calculated using the following equation:

$$\text{ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

CONTINUOUS Mode

In CONTINUOUS Mode, 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. Upon 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. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) upon timer reload.

Observe the following procedure for configuring a timer for CONTINUOUS Mode and initiating the count:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for CONTINUOUS Mode
   - Set the prescale value.
   - If using the Timer Output alternate function, set the initial output level (High or Low)
2. Write to the Timer High and Low Byte registers to set the starting count value (usually 0001H). This starting count value only affects the first pass in CONTINUOUS Mode. After the first timer reload in CONTINUOUS Mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt 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 CONTINUOUS Mode, the system clock always provides the timer input. The timer period is calculated using the following equation:

$$\text{CONTINUOUS Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

If an initial starting value other than 0001H is loaded into the Timer High and Low Byte registers, the ONE-SHOT Mode equation must be used to determine the first time-out period.

COUNTER Mode

In COUNTER Mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO Port pin Timer Input alternate function. The TPOL bit in the Timer Control Register selects whether the count occurs on the rising edge or the falling edge of the Timer Input signal. In COUNTER Mode, the prescaler is disabled.

Caution: The input frequency of the Timer Input signal must not exceed one-fourth system clock frequency.

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. 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 reload.

Observe the following procedure for configuring a timer for COUNTER Mode and initiating the count:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for COUNTER Mode
   - Select either the rising edge or falling edge of the Timer Input signal for the count. This selection also sets the initial logic level (High or Low) for the Timer Output alternate function; however, the Timer Output function does not have to be enabled.
2. Write to the Timer High and Low Byte registers to set the starting count value. This only affects the first pass in COUNTER Mode. After the first timer reload in COUNTER Mode, counting always begins at the reset value of 0001H. Generally, in COUNTER Mode the Timer High and Low Byte registers must be written with the value 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control Register to enable the timer.

In COUNTER Mode, the number of timer input transitions since the timer start is calculated using the following equation:

COUNTER Mode Timer Input Transitions = Current Count Value – Start Value

PWM Mode

In PWM Mode, the timer outputs a Pulse-Width Modulator output signal through a GPIO port pin. The timer input is the system clock. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM High and Low Byte registers. When the timer count

value matches the PWM value, the Timer Output toggles. The timer continues counting until it reaches the reload value stored in the Timer Reload High and Low Byte registers. Upon 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.

If the TPOL bit in the Timer Control Register is set to 1, the Timer Output signal begins as a High (1) and then transitions to a Low (0) when the timer value matches the PWM value. The Timer Output signal returns to a High (1) after the timer reaches the reload value and is reset to 0001H.

If the TPOL bit in the Timer Control Register is set to 0, the Timer Output signal begins as a Low (0) and then transitions to a High (1) when the timer value matches the PWM value. The Timer Output signal returns to a Low (0) after the timer reaches the reload value and is reset to 0001H.

Observe the following procedure for configuring a timer for PWM Mode and initiating the PWM operation:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for PWM Mode
   - 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 Timer Reload High and Low Byte registers to set the reload value (PWM period). The reload value must be greater than the PWM value.
5. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control Register to enable the timer and initiate counting.

The PWM period is calculated using 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 is used to determine the first PWM time-out period.

If TPOL is set to 0, the ratio of the PWM output High time to the total period is calculated using the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{Reload Value} - \text{PWM Value}}{\text{Reload Value}} \times 100$$

If TPOL is set to 1, the ratio of the PWM output High time to the total period is calculated using the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

CAPTURE Mode

In CAPTURE Mode, the current timer count value is recorded when the appropriate external Timer Input transition occurs. The capture count value is written to the Timer PWM High and Low Byte registers. The timer input is the system clock. The TPOL bit in the Timer Control Register determines if the Capture occurs on a rising edge or a falling edge of the Timer Input signal. When the capture event occurs, an interrupt is generated and the timer continues counting.

The timer continues counting up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt and continues counting.

Observe the following procedure for configuring a timer for CAPTURE Mode and initiating the count:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for CAPTURE Mode
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the Timer Input
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. Clear the Timer PWM High and Low Byte registers to 0000H. This allows user software to determine if interrupts were generated by either a capture event or a reload. If

the PWM High and Low Byte registers still contains 0000H after the interrupt, then the interrupt was generated by a reload.

5. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin for the Timer Input alternate function.
7. Write to the Timer Control Register to enable the timer and initiate counting.

In CAPTURE Mode, the elapsed time from timer start to capture event is calculated using the following equation:

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

COMPARE Mode

In COMPARE Mode, the timer counts up to the 16-bit maximum Compare value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0001H). Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) upon Compare.

If the Timer reaches FFFFH, the timer rolls over to 0000H and continue counting.

Observe the following procedure 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 required
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. If required, enable the timer interrupt 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 is 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 is still 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.

Observe the following procedure 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
2. Write to the Timer High and Low Byte registers to set the starting count value. This only affects the first pass in GATED Mode. After the first timer reset in GATED Mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control Register to enable the timer.
7. Assert the Timer Input signal to initiate the counting.

CAPTURE/COMPARE Mode

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

Every subsequent transition of the Timer Input signal (after the first, and if appropriate) captures the current count value. The Capture value is written to the Timer PWM High and Low Byte registers. When the capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes.

If no capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes.

Observe the following procedure for configuring a timer for CAPTURE/COMPARE Mode and initiating the count:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for CAPTURE/COMPARE Mode
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the Timer Input
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
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 is 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 Output Signal Operation

Timer Output is a GPIO port pin alternate function. Generally, the Timer Output is toggled every time the counter is reloaded.

Timer Control Register Definitions

This section defines the features of the following Timer Control registers.

Timer 0–1 High and Low Byte Registers: see page 64

Timer Reload High and Low Byte Registers: see page 65

Timer 0–1 PWM High and Low Byte Registers: see page 66

Timer 0–3 Control 0 Registers: see page 67

Timer 0–1 Control 1 Registers: see page 68

Timer 0–1 High and Low Byte Registers

The Timer 0–1 High and Low Byte (TxH and TxL) registers, shown in Tables 39 and 40, contain the current 16-bit timer count value. When the timer is enabled, a read from TxH causes the value in TxL to be stored in a temporary holding register. A read from TMRL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TMRL reads the register directly.

Zilog does not recommend writing to the Timer High and Low Byte registers while the timer is enabled. There are no temporary holding registers available for write operations, so simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are written during counting, the 8-bit written value is placed in the counter (High or Low Byte) at the next clock edge. The counter continues counting from the new value.

Table 39. Timer 0–1 High Byte Register (TxH)

Bit	7	6	5	4	3	2	1	0
Field	TH							
RESET	0							
R/W	R/W							
Address	F00H, F08H							

Table 40. Timer 0–1 Low Byte Register (TxL)

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

Bit	Description
[7:0] TH, TL	**Timer High and Low Bytes** These 2 bytes, {TMRH[7:0], TMRL[7:0]}, contain the current 16-bit timer count value.

Timer Reload High and Low Byte Registers

The Timer 0–1 Reload High and Low Byte (TxRH and TxRL) registers, shown in Tables 41 and 42, store a 16-bit reload value, {TRH[7:0], TRL[7:0]}. Values written to the Timer Reload High Byte Register are stored in a temporary holding register. When a write to the Timer Reload Low Byte Register occurs, the temporary holding register value is written to the Timer High Byte Register. This operation allows simultaneous updates of the 16-bit Timer reload value.

In COMPARE Mode, the Timer Reload High and Low Byte registers store the 16-bit Compare value.

Table 41. Timer 0–1 Reload High Byte Register (TxRH)

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

Table 42. Timer 0–1 Reload Low Byte Register (TxRL)

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

Bit	Description
[7] TRH, TRL	**Timer Reload Register High and Low** These two bytes form the 16-bit reload value, {TRH[7:0], TRL[7:0]}. This value sets the maximum count value which initiates a timer reload to 0001H. In COMPARE Mode, these two bytes form the 16-bit Compare value.

Timer 0–1 PWM High and Low Byte Registers

The Timer 0–1 PWM High and Low Byte (TxPWMH and TxPWML) registers, shown in Tables 43 and 44, are used for Pulse-Width Modulator (PWM) operations. These registers also store the Capture values for the CAPTURE and CAPTURE/COMPARE modes.

Table 43. Timer 0–1 PWM High Byte Register (TxPWMH)

Bit	7	6	5	4	3	2	1	0
Field	PWMH							
RESET	0							
R/W	R/W							
Address	F04H, F0CH							

Table 44. Timer 0–1 PWM Low Byte Register (TxPWML)

Bit	7	6	5	4	3	2	1	0
Field	PWML							
RESET	0							
R/W	R/W							
Address	F05H, F0DH							

Bit	Description
[7:0] PWMH, PWML	**Pulse-Width Modulator High and Low Bytes** These two bytes, {PWMH[7:0], PWML[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the PWM output changes state. The PWM output value is set by the TPOL bit in the Timer Control (TxCTL) Register. The TxPWMH and TxPWML registers also store the 16-bit captured timer value when operating in CAPTURE or CAPTURE/COMPARE modes.

Timer 0–3 Control 0 Registers

The Timer 0–3 Control 0 (TxCTL0) registers, shown in Table 45, allow cascading of the Timers.

Table 45. Timer 0–3 Control 0 Registers (TxCTL0)

Bit	7	6	5	4	3	2	1	0
Field	Reserved			CSC	Reserved			
RESET	0							
R/W	R/W							
Address	F06H, F0EH, F16H, F1EH							

Bit	Description
[7:5]	**Reserved** These bits are reserved and must be programmed to 000.
[4] CSC	**Cascade Timers** 0 = Timer Input signal comes from the pin. 1 = For Timer 0, input signal is connected to Timer 1 output. For Timer 1, the input signal is connected to the Timer 0 output.
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

Timer 0–1 Control 1 Registers

The Timer 0–1 Control (TxCTL) registers, shown in Table 46, enable/disable the timers, set the prescaler value, and determine the timer operating mode.

Table 46. Timer 0–1 Control Registers (TxCTL)

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

Bit	Description
[7] TEN	**Timer Enable** 0 = Timer is disabled. 1 = Timer enabled to count.
[6] TPOL	**Timer Input/Output Polarity** Operation of this bit is a function of the current operating mode of the timer. **ONE-SHOT Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. **CONTINUOUS Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. **COUNTER Mode** If the timer is enabled the Timer Output signal is complemented after timer reload. 0 = Count occurs on the rising edge of the Timer Input signal. 1 = Count occurs on the falling edge of the Timer Input signal. **PWM Mode** 0 = Timer Output is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon reload. 1 = Timer Output is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon reload. **CAPTURE Mode** 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **COMPARE Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload.

Bit	Description (Continued)
[6] TPOL (cont'd.)	**GATED Mode** 0 = Timer counts when the Timer Input signal is High (1) and interrupts are generated on the falling edge of the Timer Input. 1 = Timer counts when the Timer Input signal is Low (0) and interrupts are generated on the rising edge of the Timer Input. **CAPTURE/COMPARE Mode** 0 = Counting is started on the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal. 1 = Counting is started on the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal.
[5:3] PRES	**Prescale Value** The timer input clock is divided by 2^{PRES}, where PRES is set from 0 to 7. The prescaler is reset each time the timer is disabled to ensure proper clock division each time the timer is restarted. 000 = Divide by 1. 001 = Divide by 2. 010 = Divide by 4. 011 = Divide by 8. 100 = Divide by 16. 101 = Divide by 32. 110 = Divide by 64. 111 = Divide by 128.
[2:0] TMODE	**Timer Mode** 000 = ONE-SHOT Mode. 001 = CONTINUOUS Mode. 010 = COUNTER Mode. 011 = PWM Mode. 100 = CAPTURE Mode. 101 = COMPARE Mode. 110 = GATED Mode. 111 = CAPTURE/COMPARE Mode.

Watchdog Timer

Watchdog Timer (WDT) protects against corrupt or unreliable software, power faults, and other system-level problems which can place the Z8 Encore! XP® F0822 Series device into unsuitable operating states. It includes the following features:

- On-chip RC oscillator
- A selectable time-out response; either Reset or Interrupt
- 24-bit programmable time-out value

Operation

WDT is a retriggerable one-shot timer that resets or interrupts the Z8 Encore! XP® F0822 Series device when the WDT reaches its terminal count. It uses its own dedicated on-chip RC oscillator as its clock source. The WDT has only two modes of operation: ON and OFF. When enabled, it always counts and must be refreshed to prevent a time-out. An enable is performed by executing the WDT instruction or by setting the WDT_AO option bit. The WDT_AO bit enables the WDT to operate all of the time, even if a WDT instruction has not been executed.

The WDT is a 24-bit reloadable downcounter that uses three 8-bit registers in the eZ8 CPU register space to set the reload value. The nominal WDT time-out period is calculated using the following equation:

$$\text{WDT Time-out Period (ms)} = \frac{\text{WDT Reload Value}}{10}$$

In this equation, the WDT reload value is the decimal value of the 24-bit value furnished by {WDTU[7:0], WDTH[7:0], WDTL[7:0]}; the typical Watchdog Timer RC oscillator frequency is 10kHz. WDT cannot be refreshed after it reaches 000002H. The WDT reload value must not be set to values below 000004H.

Table 47 lists the approximate time-out delays based on minimum and maximum WDT reload values.

Table 47. Watchdog Timer Approximate Time-Out Delays

WDT Reload Value (Hex)	WDT Reload Value (Decimal)	Approximate Time-Out Delay (with 10 kHz Typical WDT Oscillator Frequency)	
		Typical	Description
000004	4	400 µs	Minimum time-out delay
FFFFFF	16,777,215	1677.5 s	Maximum time-out delay

Watchdog Timer Refresh

When first enabled, the WDT is loaded with the value in the WDT Reload registers. The WDT then counts down to 000000H unless a WDT instruction is executed by the eZ8 CPU. Execution of the WDT instruction causes the downcounter to be reloaded with the WDT reload value stored in the WDT Reload registers. Counting resumes following the reload operation.

When Z8 Encore! XP® F0822 Series device is operating in DEBUG Mode (using the OCD), the WDT is continuously refreshed to prevent spurious WDT time-outs.

Watchdog Timer Time-Out Response

The WDT times out when the counter reaches 000000H. A WDT time-out generates either an Interrupt or a Reset. The WDT_RES option bit determines the time-out response of the WDT. For information regarding programming of the WDT_RES option bit, see the Option Bits chapter on page 155.

WDT Interrupt in Normal Operation

If configured to generate an interrupt when a time-out occurs, the WDT issues an interrupt request to the interrupt controller and sets the WDT status bit in the WDT Control Register. If interrupts are enabled, the eZ8 CPU responds to the interrupt request by fetching the WDT interrupt vector and executing the code from the vector address. After time-out and interrupt generation, the WDT counter rolls over to its maximum value of FFFFFH and continues counting. The WDT counter is not automatically returned to its reload value.

WDT Reset in STOP Mode

If enabled in STOP Mode and configured to generate a Reset when a time-out occurs and the device is in STOP Mode, the WDT initiates a Stop Mode Recovery. Both the WDT status bit and the stop bit in the WDT Control Register is set to 1 following the WDT time-out in STOP Mode. For more information, see the Reset and Stop Mode Recovery chapter on page 21. Default operation is for the WDT and its RC oscillator to be enabled during STOP Mode.

To minimize power consumption in STOP Mode, the WDT and its RC oscillator is disabled in STOP Mode. The following sequence configures the WDT to be disabled when the Z8F082x family device enters STOP Mode following execution of a stop instruction:

1. Write 55H to the Watchdog Timer Control Register (WDTCTL).
2. Write AAH to the Watchdog Timer Control Register (WDTCTL).
3. Write 81H to the Watchdog Timer Control Register (WDTCTL) to configure the WDT and its oscillator to be disabled during STOP Mode. Alternatively, write 00H to the WDTCTL as the third step in this sequence to reconfigure the WDT and its oscillator to be enabled during STOP Mode. This sequence only affects WDT operation in STOP Mode.

WDT Reset in Normal Operation

If configured to generate a Reset when a time-out occurs, the WDT forces the device into the Reset state. The WDT status bit in the WDT Control Register is set to 1. For more information about Reset, see the Reset and Stop Mode Recovery chapter on page 21.

WDT Reset in STOP Mode

If enabled in STOP Mode and configured to generate a Reset when a time-out occurs and the device is in STOP Mode, the WDT initiates a Stop Mode Recovery. Both the WDT status bit and the stop bit in the WDT Control Register is set to 1 following WDT time-out in STOP Mode. For more information about Reset, see the Reset and Stop Mode Recovery chapter on page 21. Default operation is for the WDT and its RC oscillator to be enabled during STOP Mode.

WDT RC Disable in STOP Mode

To minimize power consumption in STOP Mode, the WDT and its RC oscillator can be disabled in STOP Mode. The following sequence configures the WDT to be disabled when the Z8F082x family device enters STOP Mode following execution of a stop instruction:

1. Write 55H to the Watchdog Timer Control Register (WDTCTL).
2. Write AAH to the Watchdog Timer Control Register (WDTCTL).
3. Write 81H to the Watchdog Timer Control Register (WDTCTL) to configure the WDT and its oscillator to be disabled during STOP Mode. Alternatively, write 00H to the Watchdog Timer Control Register (WDTCTL) as the third step in this sequence to reconfigure the WDT and its oscillator to be enabled during STOP Mode. This sequence only affects WDT operation in STOP Mode.

Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the WDTCTL address unlocks the three Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) to allow changes to the time-out period. These write operations to the WDTCTL address produce no effect on the bits in the WDTCTL. The locking mechanism prevents spurious writes to the Reload registers. The following sequence is required to unlock the Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) for write access.

1. Write 55H to the Watchdog Timer Control Register (WDTCTL).
2. Write AAH to the Watchdog Timer Control Register (WDTCTL).
3. Write the Watchdog Timer Reload Upper Byte Register (WDTU).
4. Write the Watchdog Timer Reload High Byte Register (WDTH).
5. Write the Watchdog Timer Reload Low Byte Register (WDTL).

All three Watchdog Timer Reload registers must be written in this order. There must be no other register writes between each of these operations. If a register write occurs, the lock state machine resets and no further writes occur unless the sequence is restarted. The value in the Watchdog Timer Reload registers is loaded into the counter when the WDT is first enabled and every time a WDT instruction is executed.

Watchdog Timer Control Register Definitions

This section defines the features of the following Watchdog Timer Control registers.

Watchdog Timer Control Register (WDTCTL): see page 74

Watchdog Timer Reload Upper Byte Register (WDTU): see page 75

Watchdog Timer Reload High Byte Register (WDTH): see page 75

Watchdog Timer Reload Low Byte Register (WDTL): see page 76

Watchdog Timer Control Register

The Watchdog Timer Control Register (WDTCTL), shown in Table 48, is a read-only Register that indicates the source of the most recent Reset event, a Stop Mode Recovery event, and a WDT time-out. Reading this register resets the upper four bits to 0.

Writing the 55H, AAH unlock sequence to the Watchdog Timer Control Register (WDTCTL) address unlocks the three Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) to allow changes to the time-out period. These write operations to the WDTCTL address produce no effect on the bits in the WDTCTL. The locking mechanism prevents spurious writes to the Reload registers.

Table 48. Watchdog Timer Control Register (WDTCTL)

Bit	7	6	5	4	3	2	1	0
Field	POR	STOP	WDT	EXT	Reserved			
RESET	See Table 49.				0			
R/W	R							
Address	FF0H							

Bit	Description
[7] POR	**Power-On Reset Indicator** If this bit is set to 1, a POR event occurred. This bit is reset to 0, if a WDT time-out or Stop Mode Recovery occurs. This bit is also reset to 0, when the register is read.
[6] STOP	**Stop Mode Recovery Indicator** If this bit is set to 1, a Stop Mode Recovery occurred. If the STOP and WDT bits are both set to 1, the Stop Mode Recovery occurred due to a WDT time-out. If the stop bit is 1 and the WDT bit is 0, the Stop Mode Recovery was not caused by a WDT time-out. This bit is reset by a POR or a WDT time-out that occurred while not in STOP Mode. Reading this register also resets this bit.
[5] WDT	**Watchdog Timer Time-Out Indicator** If this bit is set to 1, a WDT time-out occurred. A POR resets this pin. A Stop Mode Recovery due a change in an input pin also resets this bit. Reading this register resets this bit.
[4] EXT	**External Reset Indicator** If this bit is set to 1, a Reset initiated by the external $\overline{\text{RESET}}$ pin occurred. A POR or a Stop Mode Recovery from a change in an input pin resets this bit. Reading this register resets this bit.
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

Table 49. Watchdog Timer Events

Reset or Stop Mode Recovery Event	POR	STOP	WDT	EXT
Power-On Reset	1	0	0	0
Reset through $\overline{\text{RESET}}$ pin assertion	0	0	0	1
Reset through WDT time-out	0	0	1	0
Reset through the OCD (OCTCTL[1] set to 1)	1	0	0	0
Reset from STOP Mode through the DBG Pin driven Low	1	0	0	0
Stop Mode Recovery through GPIO pin transition	0	1	0	0
Stop Mode Recovery through WDT time-out	0	1	1	0

Watchdog Timer Reload Upper, High and Low Byte Registers

The Watchdog Timer Reload Upper, High and Low Byte (WDTU, WDTH, WDTL) registers, shown in Tables 50 through 52, form the 24-bit reload value that is loaded into the WDT, when a WDT instruction executes. The 24-bit reload value is {WDTU[7:0], WDTH[7:0], WDTL[7:0]}. Writing to these registers sets the required reload value. Reading from these registers returns the current WDT count value.

Caution: The 24-bit WDT reload value must not be set to a value less than 000004H.

Table 50. Watchdog Timer Reload Upper Byte Register (WDTU)

Bit	7	6	5	4	3	2	1	0
Field	WDTU							
RESET	1							
R/W	R/W*							
Address	FF1H							

Note: *R/W = a read returns the current WDT count value; a write sets the appropriate reload value.

Bit	Description
[7:0] WDTU	**WDT Reload Upper Byte** Most significant byte (MSB), bits [23:16] of the 24-bit WDT reload value.

Table 51. Watchdog Timer Reload High Byte Register (WDTH)

Bit	7	6	5	4	3	2	1	0
Field	WDTH							
RESET	1							
R/W	R/W*							
Address	FF2H							

Note: *R/W = a read returns the current WDT count value; a write sets the appropriate reload value.

Bit	Description
[7:0] WDTH	**WDT Reload High Byte** Middle byte, bits [15:8] of the 24-bit WDT reload value.

Table 52. Watchdog Timer Reload Low Byte Register (WDTL)

Bit	7	6	5	4	3	2	1	0
Field	WDTL							
RESET	1							
R/W	R/W*							
Address	FF3H							

Note: *R/W = a read returns the current WDT count value; a write sets the appropriate reload value.

Bit	Description
[7:0] WDTL	**WDT Reload Low** Least significant byte (LSB), bits [7:0] of the 24-bit WDT reload value.

Universal Asynchronous Receiver/ Transmitter

The Universal Asynchronous Receiver/Transmitter (UART) is a full-duplex communication channel capable of handling asynchronous data transfers. The UART uses a single 8-bit data mode with selectable parity. Features of the UART include:

- 8-bit asynchronous data transfer
- Selectable even- and odd-parity generation and checking
- Option of one or two stop bits
- Separate transmit and receive interrupts
- Framing, parity, overrun, and break detection
- Separate transmit and receive enables
- 16-bit Baud Rate Generator
- Selectable MULTIPROCESSOR (9-Bit) Mode with three configurable interrupt schemes
- BRG timer mode
- Driver Enable output for external bus transceivers

Architecture

The UART consists of three primary functional blocks: Transmitter, Receiver, and Baud Rate Generator. The UART's transmitter and receiver functions independently, but use the same baud rate and data format. Figure 11 displays the UART architecture.

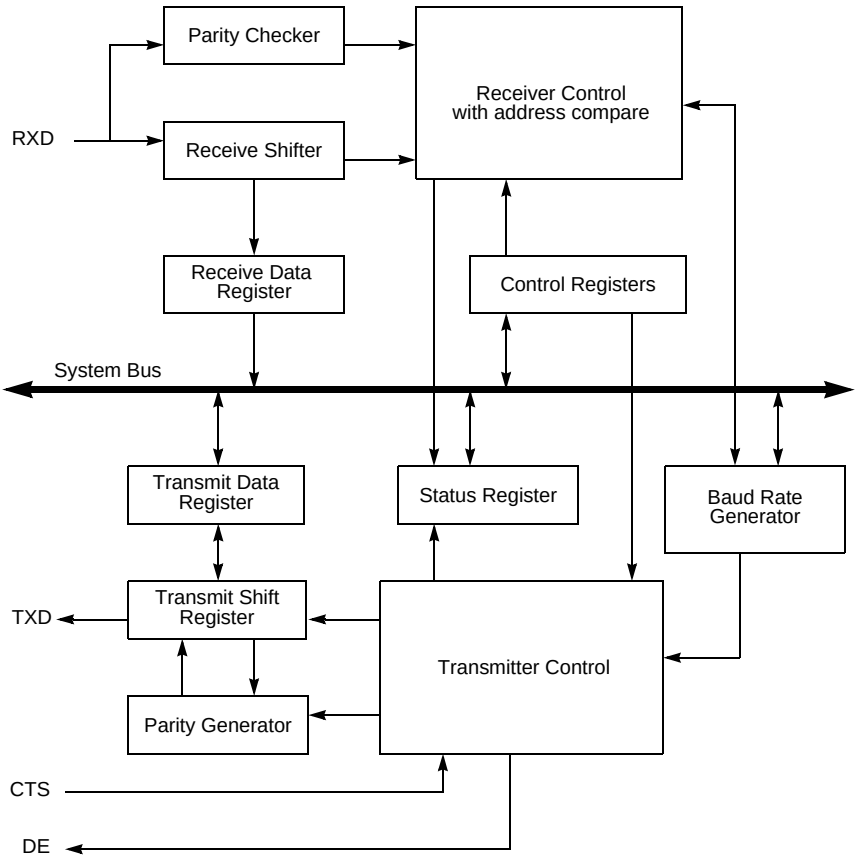


Figure 11. UART Block Diagram

Operation

The UART always transmits and receives data in an 8-bit data format, least-significant bit first. An even or odd parity bit is optionally added to the data stream. Each character begins with an active Low start bit and ends with either 1 or 2 active High stop bits. Figures 12 and 13 display the asynchronous data format used by the UART without parity and with parity, respectively.

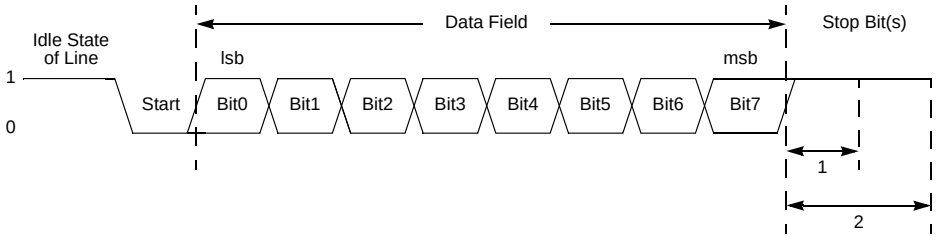


Figure 12. UART Asynchronous Data Format without Parity

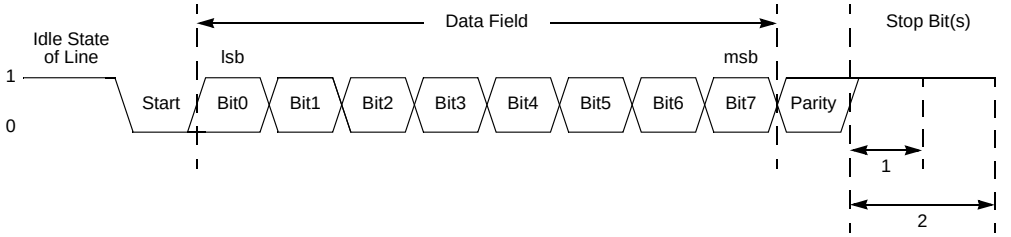


Figure 13. UART Asynchronous Data Format with Parity

Transmitting Data using Polled Method

Observe the following procedure to transmit data using polled method of operation:

1. Write to the UART Baud Rate High Byte and Low Byte registers to set the required baud rate.
2. Enable the UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. If MULTIPROCESSOR Mode is required, write to the UART Control 1 Register to enable multiprocessor (9-bit) mode functions.
   - Set the Multiprocessor Mode Select (MPEN) to enable MULTIPROCESSOR Mode.
4. Write to the UART Control 0 Register to:
   - Set the transmit enable bit (TEN) to enable the UART for data transmission
   - If parity is required, and MULTIPROCESSOR Mode is not enabled, set the parity enable bit (PEN) and select either even or odd parity (PSEL).

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

5. 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 6. 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.
6. Write the UART Control 1 Register to select the outgoing address bit:
   - Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clear it if sending a data byte.
7. Write data byte to the UART Transmit Data Register. The transmitter automatically transfers data to the Transmit Shift Register and then transmits the data.
8. If required, and multiprocessor mode is enabled, make any changes to the Multiprocessor Bit Transmitter (MPBT) value.
9. To transmit additional bytes, return to Step 5.

Transmitting Data Using Interrupt-Driven Method

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

1. Write to the UART Baud Rate High and Low Byte registers to set the required 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 required priority.
5. If MULTIPROCESSOR Mode is required, write to the UART Control 1 Register to enable MULTIPROCESSOR (9-Bit) Mode functions:
   - Set the Multiprocessor Mode Select (MPEN) to enable MULTIPROCESSOR Mode.
6. Write to the UART Control 0 Register to:
   - Set the transmit enable (TEN) bit to enable the UART for data transmission
   - Enable parity, if required, and if MULTIPROCESSOR Mode is not enabled, and select either even or odd parity.

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

7. Execute an EI instruction to enable interrupts.

The UART is now configured for interrupt-driven data transmission. Because the UART Transmit Data Register is empty, an interrupt is generated immediately. When the UART transmit interrupt is detected, the associated ISR performs the following:

1. Write the UART Control 1 Register to select the outgoing address bit:
   - Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte; clear it if sending a data byte.
2. Write the data byte to the UART Transmit Data Register. The transmitter automatically transfers data to the Transmit Shift Register and then transmits the data.
3. Clear the UART transmit interrupt bit in the applicable Interrupt Request Register.
4. Execute the IRET instruction to return from the ISR and waits for the Transmit Data Register to again become empty.

Receiving Data using the Polled Method

Observe the following procedure to configure the UART for polled data reception:

1. Write to the UART Baud Rate High and Low Byte registers to set the required baud rate.
2. Enable the UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Write to the UART Control 1 Register to enable Multiprocessor mode functions, if appropriate.
4. Write to the UART Control 0 Register to:
   - Set the receive enable bit (REN) to enable the UART for data reception
   - Enable parity, if required, and if MULTIPROCESSOR Mode is not enabled, and select either even or odd parity.
5. Check the RDA bit in the UART Status 0 Register to determine if the Receive Data Register contains a valid data byte (indicated by 1). If RDA is set to 1 to indicate available data, continue to Step 6. If the Receive Data Register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.

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 5 to receive additional data.

Receiving Data Using Interrupt-Driven Method

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

1. Write to the UART Baud Rate High and Low Byte registers to set the required 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 required 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 required 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! XP 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 required, and if MULTIPROCESSOR Mode is not enabled, and select either even or odd parity.
9. Execute an EI instruction to enable interrupts.

The UART is now configured for interrupt-driven data reception. When the UART Receiver Interrupt is detected, the associated ISR performs the following operations:

1. Check the UART Status 0 Register to determine the source of the interrupt, whether error, break or received data.
2. If the interrupt was due to data available, read the 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].
3. Clear the UART Receiver Interrupt in the applicable Interrupt Request Register.
4. Execute the IRET instruction to return from the ISR and await more data.

Clear To Send 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 would be done 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 features 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 is transmitted following the 8 bits of data and immediately preceding the stop bit(s); this character format is shown in Figure 14.

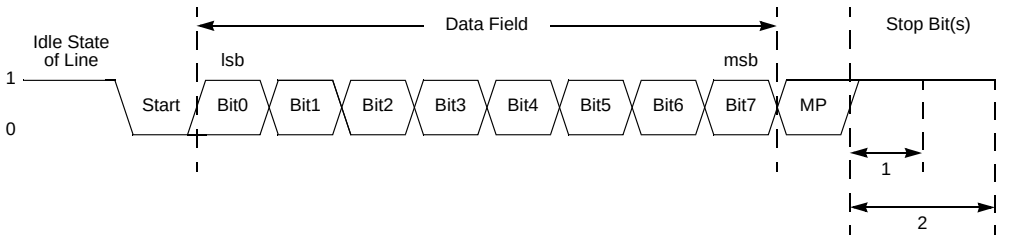


Figure 14. UART Asynchronous Multiprocessor Mode Data Format

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 in hardware, software, or combination of the two depending on the multiprocessor configuration bits. In general, the address compare feature reduces the load on the CPU, because it is not required to access the UART when it receives data directed to other devices on the multinode network. The following MULTIPROCESSOR modes are available in hardware:

- Interrupt on all address bytes
- Interrupt on matched address bytes and correctly framed data bytes
- Interrupt only on correctly framed data bytes

These modes are selected with MPMD[1:0] in the UART Control 1 Register. For all MULTIPROCESSOR modes, bit MPEN of the UART Control 1 Register must be set to 1.

The first scheme is enabled by writing 01b to MPMD[1:0]. In this mode, all incoming address bytes cause an interrupt, while data bytes never cause an interrupt. The ISR must manually check the address byte that caused triggered the interrupt. If it matches the UART address, the software should clear MPMD[0]. At this point, each new incoming byte interrupts the CPU. The software is then responsible for determining the end-of-frame. It checks for the end-of-frame by reading the MPRX bit of the UART Status 1 Register for each incoming byte. If MPRX=1, then a new frame begins. If the address of this new frame is different from the UART's address, then MPMD[0] must be set to 1 causing the UART interrupts to go inactive until the next address byte. If the new frame's address matches the UART's address, then the data in the new frame should be processed as well.

The second scheme is enabled by setting MPMD[1:0] to 10b and writing the UART's address into the UART Address Compare Register. This mode introduces more hardware control, interrupting only on frames that match the UART's address. When an incoming address byte does not match the UART's address, it is ignored. All successive data bytes in this frame are also ignored. When a matching address byte occurs, an interrupt is issued and further interrupts occur on each successive data byte. The first data byte in the frame contains the NEWFRM=1 in the UART Status 1 Register. When the next address byte occurs, the hardware compares it to the UART's address. If there is a match, the interrupts continue and the NEWFRM bit is set for the first byte of the new frame. If there is no match, then the UART ignores all incoming bytes until the next address match.

The third scheme is enabled by setting MPMD[1:0] to 11b and by writing the UART's address into the UART Address Compare Register. This mode is identical to the second scheme, except that there are no interrupts on address bytes. The first data byte of each frame is still accompanied by a NEWFRM assertion.

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 multitransceiver 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 shown in Figure 15. 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 format allows a setup time to enable the transceiver. The Driver Enable signal deasserts one system clock period after the last 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.

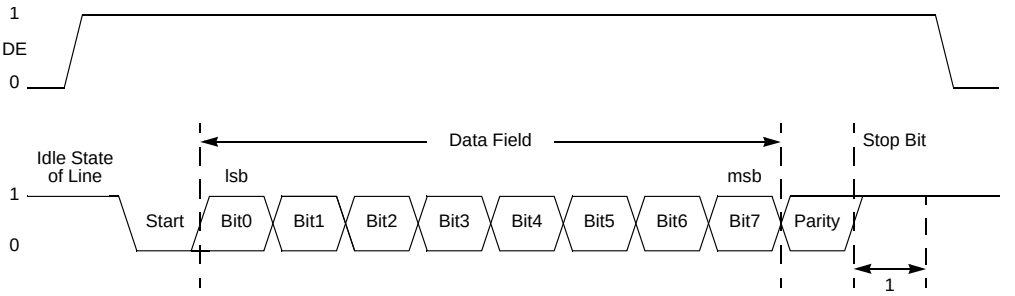


Figure 15. UART Driver Enable Signal Timing (with 1 Stop Bit and Parity)

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 BRG also functions 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. At this point, the Transmit Data Register can be written with the next character to send. This 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 independent of the other receiver interrupt sources. The received data interrupt occurs after the receive character is received and placed in the Receive Data Register. Software must respond to this received data available condition before the next character is completely received to avoid an overrun error. 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 the valid data is 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 should 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 16 displays the recommended procedure for UART receiver ISRs.

Baud Rate Generator Interrupts

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

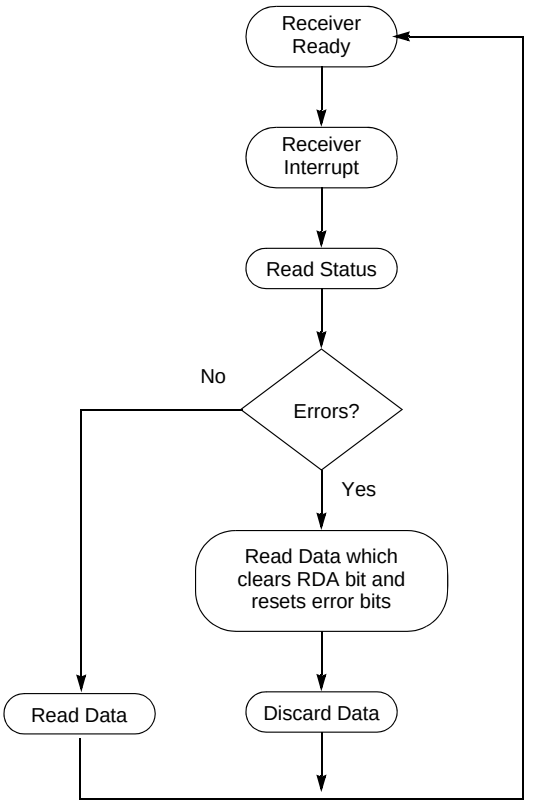


Figure 16. UART Receiver Interrupt Service Routine Flow

UART Baud Rate Generator

The UART Baud Rate Generator creates a lower frequency baud rate clock for data transmission. The input to the BRG 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 BRG functions as a basic 16-bit timer with interrupt upon time-out. Observe the following procedure to configure the BRG as a timer with interrupt upon time-out:

1. Disable the UART by clearing the REN and TEN bits in the UART Control 0 Register to 0.
2. Load the appropriate 16-bit count value into the UART Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and associated interrupt by setting the BKGCTL 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. See the Infrared Encoder/Decoder chapter on page 97 for more information about the infrared operation.

UART Transmit Data Register

Data bytes written to the UART Transmit Data Register, shown in Table 53, are shifted out on the TXD*x* pin. The write-only UART Transmit Data Register shares a Register File address with the read-only UART Receive Data Register.

Table 53. UART Transmit Data Register (U0TXD)

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

Bit	Description
[7:0] TXD	**Transmit Data** UART transmitter data byte to be shifted out through the TXD*x* pin.

UART Receive Data Register

Data bytes received through the RXD*x* pin are stored in the UART Receive Data Register, which is shown in Table 54. The read-only UART Receive Data Register shares a Register File address with the write-only UART Transmit Data Register.

Table 54. UART Receive Data Register (U0RXD)

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

Bit	Description
[7:0] RXD	**Receive Data** UART receiver data byte from the RXD*x* pin.

UART Status 0 Register

The UART Status 0 and Status 1 registers, shown in Tables 55 and 56, identify the current UART operating configuration and status.

Table 55. UART Status 0 Register (U0STAT0)

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

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

Bit	Description (Continued)
[1] TXE	**Transmitter Data Register Empty** This bit indicates that the UART Transmit Data Register is empty and ready for additional data. Writing to the UART Transmit Data Register resets this bit. 0 = Do not write to the UART Transmit Data Register. 1 = The UART Transmit Data Register is ready to receive an additional byte to be transmitted.
[0] CTS	**CTS Signal** When this bit is read it returns the level of the $\overline{\text{CTS}}$ signal.

UART Status 1 Register

The UART Status 1 Register, shown in Table 56, contains multiprocessor control and status bits.

Table 56. UART Status 1 Register (U0STAT1)

Bit	7	6	5	4	3	2	1	0
Field	Reserved						NEWFRM	MPRX
RESET	0							
R/W	R				R/W		R	
Address	F44H							

Bit	Description
[7:2]	**Reserved** These bits are reserved and must be programmed to 000000.
[1] NEWFRM	**New Frame** Status bit denoting the start of a new frame. Reading the UART Receive Data Register resets this bit to 0. 0 = The current byte is not the first data byte of a new frame. 1 = The current byte is the first data byte of a new frame.
[0] MPRX	**Multiprocessor Receive** Returns the value of the last multiprocessor bit received. Reading from the UART Receive Data Register resets this bit to 0.

UART Control 0 and Control 1 Registers

The UART Control 0 and Control 1 registers, shown in Tables 57 and 58, configure the properties of the UART's transmit and receive operations. The UART Control registers must not been written while the UART is enabled.

Table 57. UART Control 0 Register (U0CTL0)

Bit	7	6	5	4	3	2	1	0
Field	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
RESET	0							
R/W	R/W							
Address	F42H							

Bit	Description
[7] TEN	**Transmit Enable** This bit enables or disables the transmitter. The enable is also controlled by the $\overline{CTS}$ signal and the CTSE bit. If the $\overline{CTS}$ signal is Low and the CTSE bit is 1, the transmitter is enabled. 0 = Transmitter disabled. 1 = Transmitter enabled.
[6] REN	**Receive Enable** This bit enables or disables the receiver. 0 = Receiver disabled. 1 = Receiver enabled.
[5] CTSE	**CTS Enable** 0 = The $\overline{CTS}$ signal has no effect on the transmitter. 1 = The UART recognizes the $\overline{CTS}$ signal as an enable control from the transmitter.
[4] PEN	**Parity Enable** This bit enables or disables parity. Even or odd is determined by the PSEL bit. This bit is overridden by the MPEN bit. 0 = Parity is disabled. 1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit.
[3] PSEL	**Parity Select** 0 = Even parity is transmitted and expected on all received data. 1 = Odd parity is transmitted and expected on all received data.
[2] SBRK	**Send Break** This bit pauses or breaks data transmission by forcing the Transmit data output to 0. Sending a break interrupts any transmission in progress, so ensure that the transmitter has finished sending data before setting this bit. The UART does not automatically generate a stop bit when SBRK is deasserted. Software must time the duration of the break and the duration of any appropriate stop bit time following the break. 0 = No break is sent. 1 = The output of the transmitter is zero.

Bit	Description (Continued)
[1] STOP	**Stop Bit Select** 0 = The transmitter sends one stop bit. 1 = The transmitter sends two stop bits.
[0] LBEN	**Loop Back Enable** 0 = Normal operation. 1 = All transmitted data is looped back to the receiver.

Table 58. UART Control 1 Register (U0CTL1)

Bit	7	6	5	4	3	2	1	0
Field	MPMD[1]	MPEN	MPMD[0]	MPBT	DEPOL	BRGCTL	RDAIRQ	IREN
RESET	0							
R/W	R/W							
Address	F43H							

Bit	Description
[7,5] MPMD[1,0]	**Multiprocessor Mode** If MULTIPROCESSOR (9-Bit) Mode is enabled, 00 = The UART generates an interrupt request on all received bytes (data and address). 01 = The UART generates an interrupt request only on received address bytes. 10 = The UART generates an interrupt request when a received address byte matches the value stored in the Address Compare Register and on all successive data bytes until an address mismatch occurs. 11 = The UART generates an interrupt request on all received data bytes for which the most recent address byte matched the value in the Address Compare Register.
[6] MPEN	**Multiprocessor (9-Bit) Enable** This bit is used to enable MULTIPROCESSOR (9-Bit) Mode. 0 = Disable MULTIPROCESSOR (9-Bit) Mode. 1 = Enable MULTIPROCESSOR (9-Bit) Mode.
[4] MPBT	**Multiprocessor Bit Transmit** This bit is applicable only when MULTIPROCESSOR (9-Bit) Mode is enabled. 0 = Send a 0 in the multiprocessor bit location of the data stream (9th bit). 1 = Send a 1 in the multiprocessor bit location of the data stream (9th bit).
[3] DEPOL	**Driver Enable Polarity** 0 = DE signal is Active High. 1 = DE signal is Active Low.

Bit	Description (Continued)
[2] BRGCTL	**Baud Rate Control** This bit causes different UART behavior depending on whether the UART receiver is enabled (REN = 1 in the UART Control 0 Register). When the UART receiver is not enabled, this bit determines whether the BRG will issue interrupts. 0 = Reads from the Baud Rate High and Low Byte registers return the BRG reload value 1 = The BRG generates a receive interrupt when it counts down to zero. Reads from the Baud Rate High and Low Byte registers return the current BRG count value. When the UART receiver is enabled, this bit allows reads from the Baud Rate registers to return the BRG count value instead of the reload value. 0 = Reads from the Baud Rate High and Low Byte registers return the BRG reload value. 1 = Reads from the Baud Rate High and Low Byte registers return the current BRG count value. Unlike the Timers, there is no mechanism to latch the High Byte when the Low Byte is read.
[1] RDAIRQ	**Receive Data Interrupt Enable** 0 = Received data and receiver errors generates an interrupt request to the Interrupt Controller. 1 = Received data does not generate an interrupt request to the Interrupt Controller. Only receiver errors generate an interrupt request.
[0] IREN	**Infrared Encoder/Decoder Enable** 0 = Infrared Encoder/Decoder is disabled. UART operates normally operation. 1 = Infrared Encoder/Decoder is enabled. The UART transmits and receives data through the Infrared Encoder/Decoder.

UART Address Compare Register

The UART Address Compare Register, shown in Table 59, stores the multinode network address of the UART. When the MPMD[1] bit of UART Control Register 0 is set, all incoming address bytes will be compared to the value stored in the Address Compare Register. Receive interrupts and RDA assertions will only occur in the event of a match.

Table 59. UART Address Compare Register (U0ADDR)

Bit	7	6	5	4	3	2	1	0
Field	COMP_ADDR							
RESET	0							
R/W	R/W							
Address	F45H							

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

UART Baud Rate High and Low Byte Registers

The UART Baud Rate High and Low Byte registers, shown in Tables 60 and 61, combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the UART.

Table 60. UART Baud Rate High Byte Register (U0BRH)

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

Table 61. UART Baud Rate Low Byte Register (U0BRL)

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

The UART data rate is calculated using the following equation:

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

For a given UART data rate, the integer baud rate divisor value is calculated using the following equation:

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

The baud rate error relative to the desired baud rate is calculated using the following equation:

$$\text{UART Baud Rate Error (\%)} = 100 \times \left(\frac{\text{Actual Data Rate} - \text{Desired Data Rate}}{\text{Desired Data Rate}}\right)$$

For reliable communication, the UART baud rate error must never exceed 5 percent. Table 62 provides information about data rate errors for popular baud rates and commonly used crystal oscillator frequencies.

Table 62. UART Baud Rates

10.0 MHz System Clock				5.5296 MHz System Clock			
Desired Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Desired 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	1	625.0	0.00	625.0	N/A	N/A	N/A
250.0	3	208.33	–16.67	250.0	1	345.6	38.24
115.2	5	125.0	8.51	115.2	3	115.2	0.00
57.6	11	56.8	–1.36	57.6	6	57.6	0.00
38.4	16	39.1	1.73	38.4	9	38.4	0.00
19.2	33	18.9	0.16	19.2	18	19.2	0.00
9.60	65	9.62	0.16	9.60	36	9.60	0.00
4.80	130	4.81	0.16	4.80	72	4.80	0.00
2.40	260	2.40	–0.03	2.40	144	2.40	0.00
1.20	521	1.20	–0.03	1.20	288	1.20	0.00
0.60	1042	0.60	–0.03	0.60	576	0.60	0.00
0.30	2083	0.30	0.2	0.30	1152	0.30	0.00
3.579545 MHz System Clock				**1.8432 MHz System Clock**			
Desired Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Desired 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

Z8 Encore! XP® F0822 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! XP and IrDA Physical Layer Specification, v1.3-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 17 displays the architecture of the infrared endec.

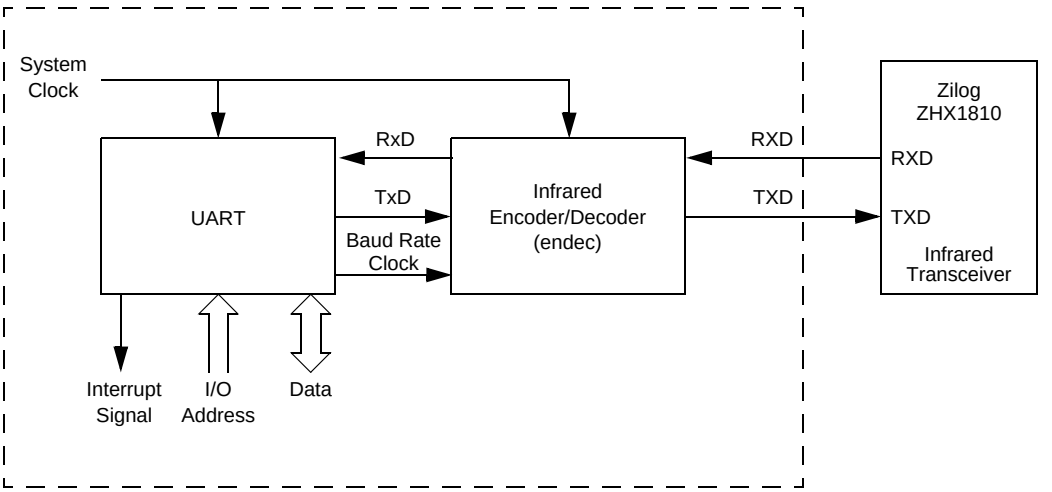


Figure 17. Infrared Data Communication System Block Diagram

Operation

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

then 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, a 3-clock High pulse is output following a 7-clock Low period. After the 3-clock High pulse, a 6-clock Low pulse is output to complete the full 16-clock data period. Figure 18 displays IrDA data transmission. When the infrared endec is enabled, the UART's TXD signal is internal to the Z8 Encore! XP® F0822 Series products while the IR_TXD signal is output through the TXD pin.

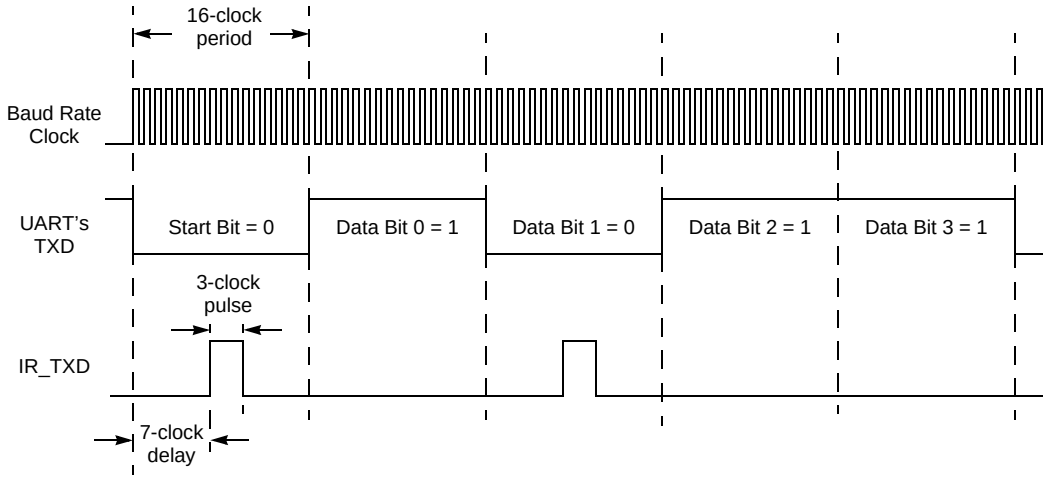


Figure 18. Infrared Data Transmission

Receiving IrDA Data

Data received from the infrared transceiver through 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 19 displays data reception. When the infrared endec is enabled, the UART's RXD signal is internal to the Z8 Encore! XP® F0822 Series products while the IR_RXD signal is received through the RXD pin.

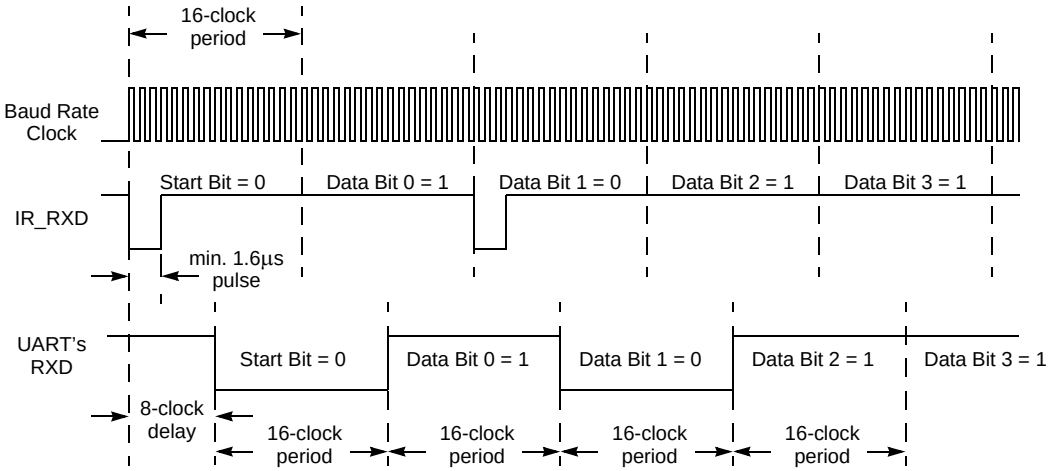


Figure 19. Infrared Data Reception

Caution: The system clock frequency must be at least 1.0MHz to ensure proper reception of the 1.6µs 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. The window remains open until the count again reaches 8 (or, in other words, 24 baud

clock periods since the previous pulse was detected). This period allows the endec a sampling window of –4 to +8 baud rate clocks around the expected time of an incoming pulse. If an incoming pulse is detected inside this window, this process is repeated. If the incoming data is a logical 1 (no pulse), the endec returns to its initial state and waits for the next falling edge. As each falling edge is detected, the endec clock counter is reset to resynchronize the endec to the incoming signal. This routine allows the endec to tolerate jitter and baud rate errors in the incoming data stream. Resynchronizing the endec does not alter the operation of the UART, which ultimately receives the data. The UART is only synchronized to the incoming data stream when a start bit is received.

Infrared Endec Control Register Definitions

All infrared endec configuration and status information is set by the UART control registers as defined in the UART Control Register Definitions section on page 88.

Caution: To prevent spurious signals during IrDA data transmission, set the IREN bit in the UART Control 1 Register to 1 to enable the infrared endec before enabling the GPIO port alternate function for the corresponding pin.

Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a synchronous interface allowing several SPI-type devices to be interconnected. SPI-compatible devices include EEPROMs, Analog-to-Digital Converters, and ISDN devices. Features of the SPI include:

- Full-duplex, synchronous, and character-oriented communication
- Four-wire interface
- Data transfers rates up to a maximum of one-half the system clock frequency
- Error detection
- Dedicated Baud Rate Generator

The SPI is not available in 20-pin package devices

Architecture

The SPI is be configured as either a Master (in single- or multiple-master systems) or a Slave, as shown in Figures 20 through 22.

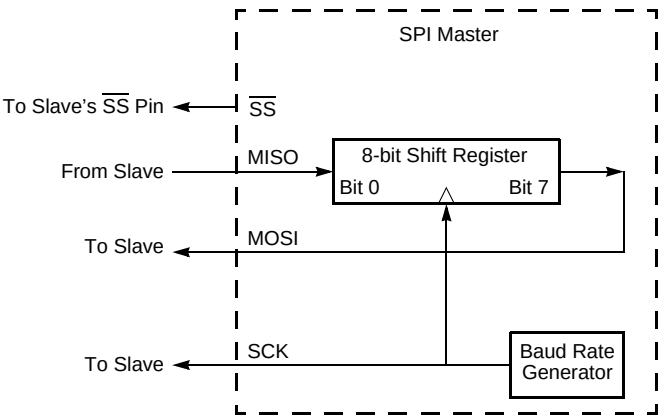


Figure 20. SPI Configured as a Master in a Single Master, Single Slave System

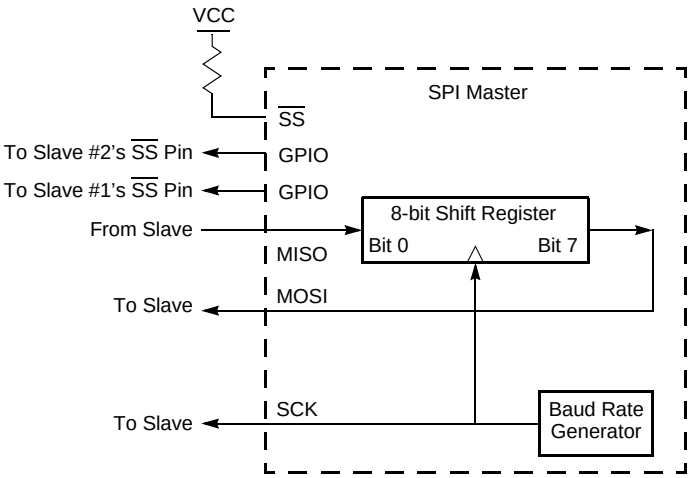


Figure 21. SPI Configured as a Master in a Single Master, Multiple Slave System

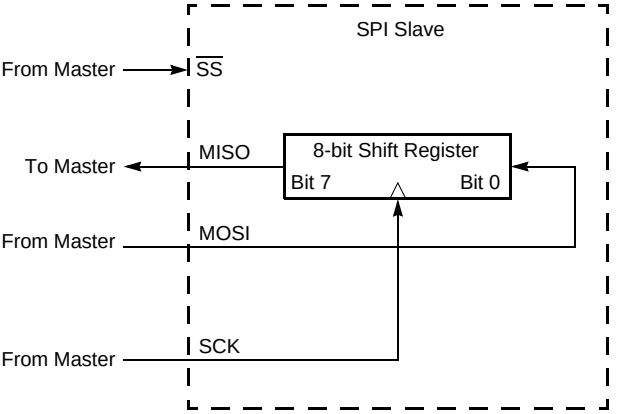


Figure 22. SPI Configured as a Slave

Operation

The SPI is a full-duplex, synchronous, and character-oriented channel that supports a four-wire interface (serial clock, transmit, receive and Slave select). The SPI block consists of a transmit/receive shift register, a Baud Rate (clock) Generator and a control unit.

During an SPI transfer, data is sent and received simultaneously by both the Master and the Slave SPI devices. Separate signals are required for data and the serial clock. When an SPI transfer occurs, a multibit (typically 8-bit) character is shifted out one data pin and an multibit character is simultaneously shifted in on a second data pin. An 8-bit shift register in the Master and another 8-bit shift register in the Slave are connected as a circular buffer. The SPI Shift Register is single-buffered in the transmit and receive directions. New data to be transmitted cannot be written into the shift register until the previous transmission is complete and receive data (if valid) has been read.

SPI Signals

The four basic SPI signals are:

- MISO (Master-In, Slave-Out)
- MOSI (Master-Out, Slave-In)
- SCK (Serial Clock)
- $\overline{SS}$ (Slave Select)

The following sections discuss these SPI signals. Each signal is described in both Master and Slave modes.

Master-In/Slave-Out

The Master-In/Slave-Out (MISO) pin is configured as an input in a Master device and as an output in a Slave device. It is one of the two lines that transfer serial data, with the most significant bit sent first. The MISO pin of a Slave device is placed in a high-impedance state if the Slave is not selected. When the SPI is not enabled, this signal is in a high-impedance state.

Master-Out/Slave-In

The Master-Out/Slave-In (MOSI) pin is configured as an output in a Master device and as an input in a Slave device. It is one of the two lines that transfer serial data, with the most significant bit sent first. When the SPI is not enabled, this signal is in a high-impedance state.

Serial Clock

The Serial Clock (SCK) synchronizes data movement both in and out of the device through its MOSI and MISO pins. In MASTER Mode, the SPI's Baud Rate Generator creates the serial clock. The Master drives the serial clock out its own SCK pin to the Slave's SCK pin. When the SPI is configured as a Slave, the SCK pin is an input and the clock signal from the Master synchronizes the data transfer between the Master and Slave devices. Slave devices ignore the SCK signal, unless the $\overline{SS}$ pin is asserted. When configured as a slave, the SPI block requires a minimum SCK period of greater than or equal to 8 times the system (X_{IN}) clock period.

The Master and Slave are each capable of exchanging a character of data during a sequence of NUMBITS clock cycles (see the NUMBITS field in the SPI Mode Register). In both Master and Slave SPI devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. Edge polarity is determined by the SPI phase and polarity control.

Slave Select

The active Low Slave Select ($\overline{SS}$) input signal selects a Slave SPI device. $\overline{SS}$ must be Low prior to all data communication to and from the Slave device. $\overline{SS}$ must stay Low for the full duration of each character transferred. The $\overline{SS}$ signal can stay Low during the transfer of multiple characters or can deassert between each character.

When the SPI is configured as the only Master in an SPI system, the $\overline{SS}$ pin is set as either an input or an output. For communication between the Z8 Encore! XP® F0822 Series device's SPI Master and external Slave devices, the $\overline{SS}$ signal, as an output, asserts the $\overline{SS}$ input pin on one of the Slave devices. Other GPIO output pins can also be employed to select external SPI Slave devices.

When the SPI is configured as one Master in a multimaster SPI system, the $\overline{SS}$ pin should be set as an input. The $\overline{SS}$ input signal on the Master must be High. If the $\overline{SS}$ signal goes Low (indicating another Master is driving the SPI bus), a Collision error flag is set in the SPI Status Register.

SPI Clock Phase and Polarity Control

The SPI supports four combinations of serial clock phase and polarity using two bits in the SPI Control Register. The clock polarity bit, CLKPOL, selects an active High or active Low clock and has no effect on the transfer format. Table 63 lists the SPI Clock Phase and Polarity Operation parameters. The clock phase bit, PHASE, selects one of two fundamentally different transfer formats. For proper data transmission, the clock phase and polarity must be identical for the SPI Master and the SPI Slave. The Master always places data on the MOSI line a half-cycle before the receive clock edge (SCK signal), in order for the Slave to latch the data.

Table 63. SPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation

PHASE	CLKPOL	SCK Transmit Edge	SCK Receive Edge	SCK Idle State
0	0	Falling	Rising	Low
0	1	Rising	Falling	High
1	0	Rising	Falling	Low
1	1	Falling	Rising	High

Transfer Format PHASE is 0

Figure 23 displays the timing diagram for an SPI transfer in which PHASE is cleared to 0. The two SCK waveforms show polarity with CLKPOL reset to 0 and with CLKPOL set to one. The diagram can be interpreted as either a Master or Slave timing diagram, because the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the Master and the Slave.

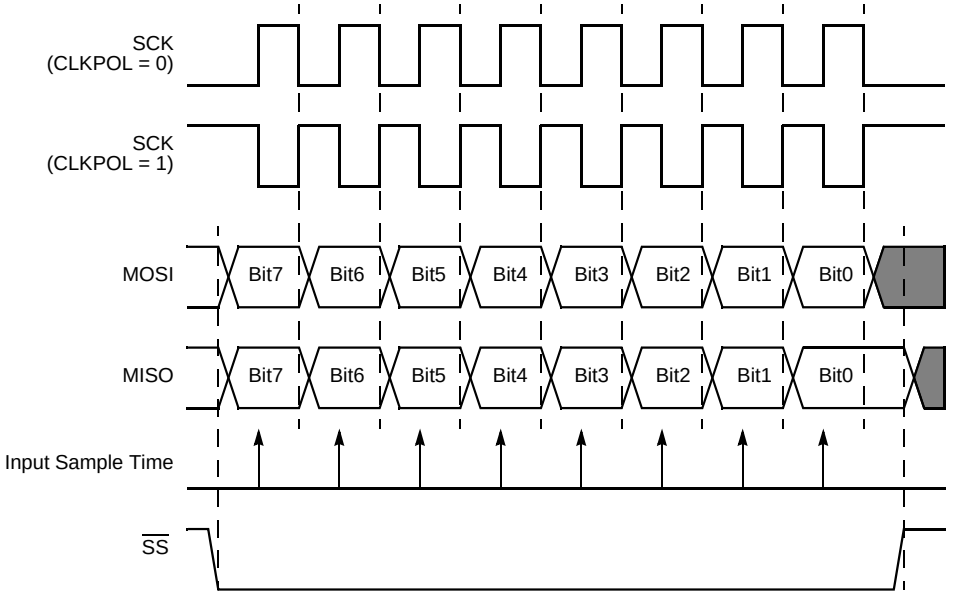


Figure 23. SPI Timing When PHASE is 0

Transfer Format PHASE is 1

Figure 24 displays the timing diagram for an SPI transfer in which PHASE is one. Two waveforms are depicted for SCK, one for CLKPOL reset to 0 and another for CLKPOL set to 1.

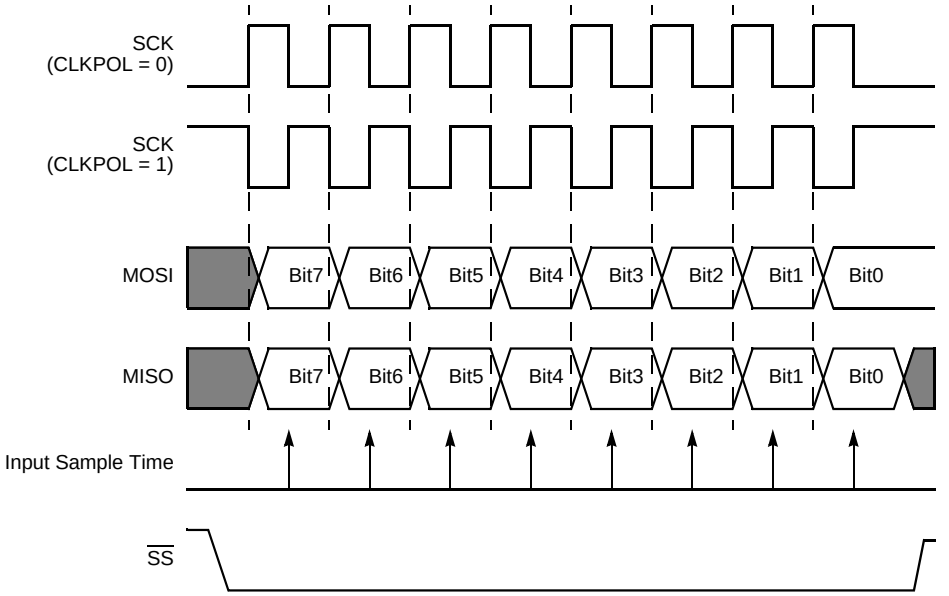


Figure 24. SPI Timing When PHASE is 1

Multimaster Operation

In a multimaster SPI system, all SCK pins are tied together, all MOSI pins are tied together and all MISO pins are tied together. All SPI pins must then be configured in OPEN-DRAIN Mode to prevent bus contention. At any one time, only one SPI device is configured as the Master and all other SPI devices on the bus are configured as Slaves. The Master enables a single Slave by asserting the $\overline{SS}$ pin on that Slave only. Then, the single Master drives data out its SCK and MOSI pins to the SCK and MOSI pins on the Slaves (including those which are not enabled). The enabled Slave drives data out its MISO pin to the MISO Master pin.

For a Master device operating in a multimaster system, if the $\overline{SS}$ pin is configured as an input and is driven Low by another Master, the COL bit is set to 1 in the SPI Status Register. The COL bit indicates the occurrence of a multimaster collision (mode fault error condition).

Slave Operation

The SPI block is configured for SLAVE Mode operation by setting the SPIEN bit to 1 and the MMEN bit to 0 in the SPICTL Register and setting the SSIO bit to 0 in the SPIMODE Register. The IRQE, PHASE, CLKPOL, and WOR bits in the SPICTL Register and the NUMBITS field in the SPIMODE Register must be set to be consistent with the other SPI devices. The STR bit in the SPICTL Register can be used, if appropriate, to force a *start-up* interrupt. The BIRQ bit in the SPICTL Register and the SSV bit in the SPIMODE Register is not used in SLAVE Mode. The SPI Baud Rate Generator is not used in SLAVE Mode; therefore, the SPIBRH and SPIBRL registers are not required to be initialized.

If the slave has data to send to the master, the data must be written to the SPIDAT Register before the transaction starts (first edge of SCK when $\overline{SS}$ is asserted). If the SPIDAT Register is not written prior to the slave transaction, the MISO pin outputs whatever value is currently in the SPIDAT Register.

Due to the delay resulting from synchronization of the SPI input signals to the internal system clock, the maximum SPICLK baud rate that can be supported in SLAVE Mode is the system clock frequency (X_{IN}) divided by 8. This rate is controlled by the SPI Master.

Error Detection

The SPI contains error detection logic to support SPI communication protocols and recognize when communication errors have occurred. The SPI Status Register indicates when a data transmission error has been detected.

Overrun (Write Collision)

An overrun error (write collision) indicates a write to the SPI Data Register was attempted while a data transfer is in progress (in either Master or Slave modes). An overrun sets the OVR bit in the SPI Status Register to 1. Writing a 1 to OVR clears this error flag. The data register is not altered when a write occurs while data transfer is in progress.

Mode Fault (Multimaster Collision)

A mode fault indicates when more than one Master is trying to communicate at the same time (a multimaster collision). The mode fault is detected when the enabled Master's $\overline{SS}$ pin is asserted. A mode fault sets the COL bit in the SPI Status Register to 1. Writing a 1 to COL clears this error flag.

SLAVE Mode Abort

In SLAVE Mode, if the $\overline{SS}$ pin deasserts before all bits in a character have been transferred, the transaction aborts. When this condition occurs, the ABT bit is set in the SPISTAT Register, and so is the IRQ bit (indicating that the transaction is complete). The next time $\overline{SS}$ asserts, the MISO pin outputs SPIDAT[7], regardless of where the previous transaction left off. Writing a 1 to ABT clears this error flag.

SPI Interrupts

When SPI interrupts are enabled, the SPI generates an interrupt after character transmission/reception completes in both Master and Slave modes. A character is defined to be 1 through 8 bits by the NUMBITS field in the SPI Mode Register. In SLAVE Mode it is not necessary for $\overline{SS}$ to deassert between characters to generate the interrupt. The SPI in SLAVE Mode also generates an interrupt if the $\overline{SS}$ signal deasserts prior to transfer of all of the bits in a character (see the previous paragraph). Writing a 1 to the IRQ bit in the SPI Status Register clears the pending SPI interrupt request. The IRQ bit must be cleared to 0 by the ISR to generate future interrupts. To start the transfer process, an SPI interrupt can be forced by software writing a 1 to the STR bit in the SPICTL Register.

If the SPI is disabled, an SPI interrupt can be generated by a BRG time-out. This timer function must be enabled by setting the BIRQ bit in the SPICTL Register. This BRG time-out does not set the IRQ bit in the SPISTAT Register, just the SPI interrupt bit in the interrupt controller.

SPI Baud Rate Generator

In SPI MASTER Mode, the BRG creates a lower frequency serial clock (SCK) for data transmission synchronization between the Master and the external Slave. The input to the BRG is the system clock. The SPI Baud Rate High and Low Byte registers combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator. The SPI baud rate is calculated using the following equation:

$$\text{SPI Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{2 \times \text{BRG[15:0]}}$$

The minimum baud rate is obtained by setting BRG[15:0] to 0000H for a clock divisor value of (2 x 65536 = 131072).

When the SPI is disabled, BRG functions as a basic 16-bit timer with interrupt upon time-out. Observe the following procedure to configure BRG as a timer with interrupt upon time-out:

1. Disable the SPI by clearing the SPIEN bit in the SPI Control Register to 0.
2. Load the appropriate 16-bit count value into the SPI Baud Rate High and Low Byte registers.
3. Enable BRG timer function and associated interrupt by setting the BIRQ bit in the SPI Control Register to 1.

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

Interrupt Interval (s) = System Clock Period (s) ×BRG[15:0]]

SPI Control Register Definitions

This section defines the features of the following Serial Peripheral Interface registers.

SPI Data Register: see page 109

SPI Control Register: see page 110

SPI Status Register: see page 111

SPI Mode Register: see page 112

SPI Diagnostic State Register: see page 113

SPI Baud Rate High and Low Byte Registers: see page 114

SPI Data Register

The SPI Data Register, shown in Table 64, stores both the outgoing (transmit) data and the incoming (receive) data. Reads from the SPI Data Register always return the current contents of the 8-bit Shift Register. Data is shifted out starting with bit 7. The last bit received resides in bit position 0.

With the SPI configured as a Master, writing a data byte to this register initiates the data transmission. With the SPI configured as a Slave, writing a data byte to this register loads the shift register in preparation for the next data transfer with the external Master. In either the Master or Slave modes, if a transmission is already in progress, writes to this register are ignored and the Overrun error flag, OVR, is set in the SPI Status Register.

When the character length is less than 8 bits (as set by the NUMBITS field in the SPI Mode Register), the transmit character must be set as *left-justified* in the SPI Data Register. A received character of less than 8 bits is right justified (last bit received is in bit position 0). For example, if the SPI is configured for 4-bit characters, the transmit characters must be written to SPIDATA[7:4] and the received characters are read from SPIDATA[3:0].

Table 64. SPI Data Register (SPIDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	X							
R/W	R/W							
Address	F60H							

Bit	Description
[7:0] DATA	**SPI Data** Transmit and/or receive data.

SPI Control Register

The SPI Control Register, shown in Table 65, configures the SPI for transmit and receive operations.

Table 65. SPI Control Register (SPICTL)

Bit	7	6	5	4	3	2	1	0
Field	IRQE	STR	BIRQ	PHASE	CLKPOL	WOR	MMEN	SPIEN
RESET	0							
R/W	R/W							
Address	F61H							

Bit	Description
[7] IRQE	**Interrupt Request Enable** 0 = SPI interrupts are disabled. No interrupt requests are sent to the Interrupt Controller. 1 = SPI interrupts are enabled. Interrupt requests are sent to the Interrupt Controller.
[6] STR	**Start an SPI Interrupt Request** 0 = No effect. 1 = Setting this bit to 1 also sets the IRQ bit in the SPI Status Register to 1. Setting this bit forces the SPI to send an interrupt request to the Interrupt Control. This bit can be used by software for a function similar to transmit buffer empty in a UART. Writing a 1 to the IRQ bit in the SPI Status Register clears this bit to 0.
[5] BIRQ	**BRG Timer Interrupt Request** If the SPI is enabled, this bit has no effect. If the SPI is disabled: 0 = BRG timer function is disabled. 1 = BRG timer function and time-out interrupt are enabled.
[4] PHASE	**Phase Select** Sets the phase relationship of the data to the clock. For more information about operation of the PHASE bit, see the SPI Clock Phase and Polarity Control section on page 104.
[3] CLKPOL	**Clock Polarity** 0 = SCK idles Low (0). 1 = SCK idle High (1).
[2] WOR	**Wire-OR (Open-Drain) Mode Enabled** 0 = SPI signal pins not configured for open-drain. 1 = All four SPI signal pins (SCK, $\overline{SS}$, MISO, MOSI) configured for open-drain function. This setting is typically used for multimaster and/or multislave configurations.
[1] MMEN	**SPI MASTER Mode Enable** 0 = SPI configured in SLAVE Mode. 1 = SPI configured in MASTER Mode.
[0] SPIEN	**SPI Enable** 0 = SPI disabled. 1 = SPI enabled.

SPI Status Register

The SPI Status Register, shown in Table 66, indicates the current state of the SPI. All bits revert to their reset state if the SPIEN bit in the SPICTL Register equals 0.

Table 66. SPI Status Register (SPISTAT)

Bit	7	6	5	4	3	2	1	0
Field	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0							1
R/W	R/W*				R			
Address	F62H							
Note: *R/W = read access; write a 1 to clear the bit to 0.								

Bit	Description
[7] IRQ	**Interrupt Request** If SPIEN = 1, this bit is set if the STR bit in the SPICTL Register is set, or upon completion of an SPI Master or Slave transaction. This bit does not set if SPIEN = 0 and the SPI Baud Rate Generator is used as a timer to generate the SPI interrupt. 0 = No SPI interrupt request pending. 1 = SPI interrupt request is pending.
[6] OVR	**Overrun** 0 = An overrun error has not occurred. 1 = An overrun error has been detected.
[5] COL	**Collision** 0 = A multimaster collision (mode fault) has not occurred. 1 = A multimaster collision (mode fault) has been detected.
[4] ABT	**SLAVE Mode Transaction Abort** This bit is set if the SPI is configured in SLAVE Mode, a transaction is occurring and $\overline{SS}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the SPIMODE Register. The IRQ bit also sets, indicating the transaction has completed. 0 = A SLAVE Mode transaction abort has not occurred. 1 = A SLAVE Mode transaction abort has been detected.
[3:2]	**Reserved** These bits are reserved and must be programmed to 00.
[1] TXST	**Transmit Status** 0 = No data transmission currently in progress. 1 = Data transmission currently in progress.
[0] SLAS	**Slave Select** If SPI is enabled as a Slave, then the following bit settings are true: 0 = $\overline{SS}$ input pin is asserted (Low) 1 = $\overline{SS}$ input is not asserted (High). If SPI is enabled as a Master, this bit is not applicable.

SPI Mode Register

The SPI Mode Register, shown in Table 67, configures the character bit width and the direction and value of the $\overline{SS}$ pin.

Table 67. SPI Mode Register (SPIMODE)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		DIAG	NUMBITS[2:0]			SSIO	SSV
RESET	0							
R/W	R		R/W					
Address	F63H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5] DIAG	**Diagnostic Mode Control Bit** This bit is for SPI diagnostics. Setting this bit allows the BRG value to be read using the SPIBRH and SPIBRL Register locations. 0 = Reading SPIBRH, SPIBRL returns the value in the SPIBRH and SPIBRL registers 1 = Reading SPIBRH returns bits [15:8] of the SPI Baud Rate Generator; and reading SPIBRL returns bits [7:0] of the SPI Baud Rate Counter. The Baud Rate Counter High and Low byte values are not buffered. **Caution:** Be careful when reading these values while the BRG is counting, because the read may interfere with the operation of the BRG counter.
[4:2] NUMBITS[2:0]	**Number of Data Bits Per Character to Transfer** This field contains the number of bits to shift for each character transfer. See the the SPI Data Register section on page 109 for information about valid bit positions when the character length is less than 8 bits. 000 = 8 bits. 001 = 1 bit. 010 = 2 bits. 011 = 3 bits. 100 = 4 bits. 101 = 5 bits. 110 = 6 bits. 111 = 7 bits.

Bit	Description (Continued)
[1] SSIO	**Slave Select I/O** 0 = $\overline{SS}$ pin configured as an input. 1 = $\overline{SS}$ pin configured as an output (MASTER Mode only).
[0] SSV	**Slave Select Value** If SSIO = 1 and SPI is configured as a Master: 0 = $\overline{SS}$ pin driven Low (0). 1 = $\overline{SS}$ pin driven High (1). This bit has no effect if SSIO = 0 or SPI is configured as a Slave.

SPI Diagnostic State Register

The SPI Diagnostic State Register, shown in Table 68, provides observability of internal state. This register is a read-only register that is used for SPI diagnostics.

Table 68. SPI Diagnostic State Register (SPIDST)

Bit	7	6	5	4	3	2	1	0
Field	SCKEN	TCKEN	SPISTATE					
RESET	0							
R/W	R							
Address	F64H							

Bit	Description
[7] SCKEN	**Shift Clock Enable** 0 = The internal Shift Clock Enable signal is deasserted. 1 = The internal Shift Clock Enable signal is asserted (shift register is updated upon the next system clock).
[6] TCKEN	**Transmit Clock Enable** 0 = The internal Transmit Clock Enable signal is deasserted. 1 = The internal Transmit Clock Enable signal is asserted. When this signal is asserted, the serial data output is updated upon the next system clock (MOSI or MISO).
[5:0] SPISTATE	**SPI State Machine** Defines the current state of the internal SPI State Machine.

SPI Baud Rate High and Low Byte Registers

The SPI Baud Rate High and Low Byte registers, shown in Tables 69 and 70, combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator. When configured as a general purpose timer, the interrupt interval is calculated using the following equation:

Interrupt Interval (s) = System Clock Period (s) × BRG[15:0]

Table 69. SPI Baud Rate High Byte Register (SPIBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	1							
R/W	R/W							
Address	F66H							

Bit	Description
[7:0] BRH	**SPI Baud Rate High Byte** Most significant byte, BRG[15:8], of the SPI Baud Rate Generator's reload value.

Table 70. SPI Baud Rate Low Byte Register (SPIBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	1							
R/W	R/W							
Address	F67H							

Bit	Description
[7:0] BRL	**SPI Baud Rate Low Byte** Least significant byte, BRG[7:0], of the SPI Baud Rate Generator's reload value.

I²C Controller

The I²C Controller makes the F0822 Series products bus-compatible with the I²C protocol. The I²C Controller consists of two bidirectional bus lines: a serial data signal (SDA) and a serial clock signal (SCL). Features of the I²C Controller include:

- Transmit and Receive Operation in MASTER Mode
- Maximum data rate of 400 kbit/s
- 7-bit and 10-bit addressing modes for Slaves
- Unrestricted number of data bytes transmitted per transfer

The I²C Controller in the F0822 Series products does not operate in SLAVE Mode.

Architecture

Figure 25 displays the architecture of the I²C Controller.

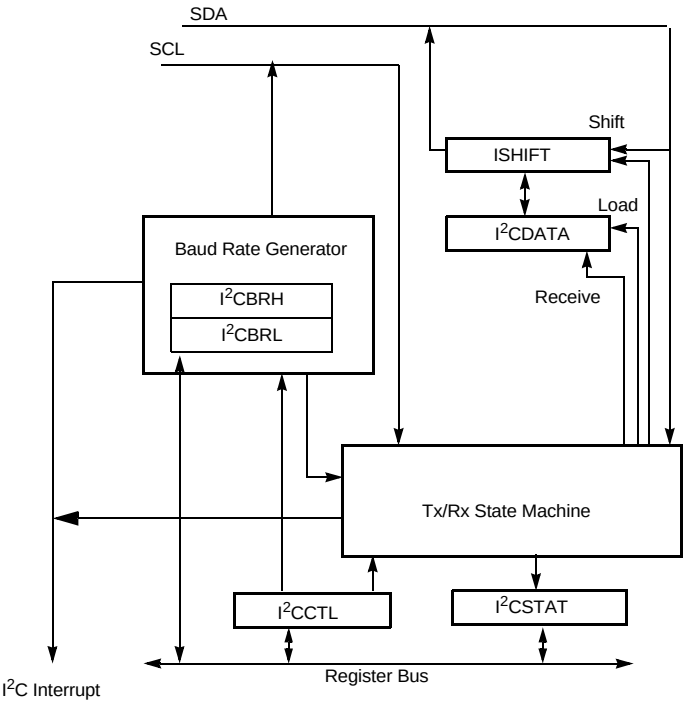


Figure 25. I²C Controller Block Diagram

Operation

The I²C Controller operates in MASTER Mode to transmit and receive data. Only a single master is supported. Arbitration between two masters must be accomplished in software. I²C supports the following operations:

- Master transmits to a 7-bit slave
- Master transmits to a 10-bit slave
- Master receives from a 7-bit slave
- Master receives from a 10-bit slave

SDA and SCL Signals

I²C sends all addresses, data and acknowledge signals over the SDA line, most-significant bit first. SCL is the common clock for the I²C Controller. When the SDA and SCL pin alternate functions are selected for their respective GPIO ports, the pins are automatically configured for open-drain operation.

The master (I²C) is responsible for driving the SCL clock signal, although the clock signal becomes skewed by a slow slave device. During the Low period of the clock, the slave pulls the SCL signal Low to suspend the transaction. The master releases the clock at the end of the Low period and notices that the clock remains Low instead of returning to a High level. When the slave releases the clock, the I²C Controller continues the transaction. All data is transferred in bytes and there is no limit to the amount of data transferred in one operation. When transmitting data or acknowledging read data from the slave, the SDA signal changes in the middle of the Low period of SCL and is sampled in the middle of the High period of SCL.

I²C Interrupts

The I²C Controller contains four sources of interrupts: Transmit, Receive, Not Acknowledge and Baud Rate Generator. These four interrupt sources are combined into a single interrupt request signal to the interrupt controller. The transmit interrupt is enabled by the IEN and TXI bits of the control register. The Receive and Not Acknowledge interrupts are enabled by the IEN bit of the control register. BRG interrupt is enabled by the BIRQ and IEN bits of the control register.

Not Acknowledge interrupts occur when a Not Acknowledge condition is received from the slave or sent by the I²C Controller and neither the start or stop bit is set. The Not Acknowledge event sets the NCKI bit of the I²C Status Register and can only be cleared by setting the start or stop bit in the I²C Control Register. When this interrupt occurs, the I²C Controller waits until either the stop or start bit is set before performing any action. In an ISR, the NCKI bit should always be checked prior to servicing transmit or receive interrupt conditions because it indicates the transaction is being terminated.

Receive interrupts occur when a byte of data has been received by the I²C Controller (Master reading data from Slave). This procedure sets the RDRF bit of the I²C Status Register. The RDRF bit is cleared by reading the I²C Data Register. The RDRF bit is set during the acknowledge phase. The I²C Controller pauses after the acknowledge phase until the receive interrupt is cleared before performing any other action.

Transmit interrupts occur when the TDRE bit of the I²C Status Register sets and the TXI bit in the I²C Control Register is set. Transmit interrupts occur under the following conditions when the Transmit Data Register is empty:

- The I²C Controller is enabled

- The first bit of the byte of an address is shifting out and the RD bit of the I2C Status Register is deasserted
- The first bit of a 10-bit address shifts out
- The first bit of write data shifts out

Note: Writing to the I2C Data Register always clears the TRDE bit to 0. When TDRE is asserted, the I2C Controller pauses at the beginning of the Acknowledge cycle of the byte currently shifting out until the data register is written with the next value to send or the stop or start bits are set indicating the current byte is the last one to send.

The fourth interrupt source is the BRG. If the I2C Controller is disabled (IEN bit in the I2CCTL Register = 0) and the BIRQ bit in the I2CCTL Register = 1, an interrupt is generated when the BRG counts down to 1. This allows the I2C Baud Rate Generator to be used by software as a general purpose timer when IEN = 0.

Software Control of I2C Transactions

Software controls I2C transactions by using the I2C Controller interrupt, by polling the I2C Status Register or by DMA. Note that not all products include a DMA Controller.

To use interrupts, the I2C interrupt must be enabled in the Interrupt Controller. The TXI bit in the I2C Control Register must be set to enable transmit interrupts.

To control transactions by polling, the interrupt bits (TDRE, RDRF and NCKI) in the I2C Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

Either or both transmit and receive data movement can be controlled by the DMA Controller. The DMA Controller channel(s) must be initialized to select the I2C transmit and receive requests. Transmit DMA requests require that the TXI bit in the I2C Control Register be set.

Caution: A transmit (write) DMA operation hangs if the slave responds with a Not Acknowledge before the last byte has been sent. After receiving the Not Acknowledge, the I2C Controller sets the NCKI bit in the Status Register and pauses until either the stop or start bits in the Control Register are set.

For a receive (read) DMA transaction to send a Not Acknowledge on the last byte, the receive DMA must be set up to receive *n–1* bytes, then software must set the NAK bit and receive the last (nth) byte directly.

Start and Stop Conditions

The Master (I^2C) drives all Start and Stop signals and initiates all transactions. To start a transaction, the I^2C Controller generates a start condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the I^2C Controller generates a Stop condition by creating a Low-to-High transition of the SDA signal while the SCL signal is High. The start and stop bits in the I^2C Control Register control the sending of start and stop conditions. A Master is also allowed to end one transaction and begin a new one by issuing a restart. This restart issuance is accomplished by setting the start bit at the end of a transaction rather than setting the stop bit.

Note: The start condition is not sent until the start bit is set and data has been written to the I^2C Data Register.

Master Write and Read Transactions

The following sections provide Zilog's recommended procedure for performing I^2C write and read transactions from the I^2C Controller (Master) to slave I^2C devices. In general, software should rely on the TDRE, RDRF and NCKI bits of the status register (these bits generate interrupts) to initiate software actions. When using interrupts or DMA, the TXI bit is set to start each transaction and cleared at the end of each transaction to eliminate a *trailing* transmit interrupt.

Caution: Caution should be used in using the ACK status bit within a transaction because it is difficult for software to tell when it is updated by hardware.

When writing data to a slave, the I^2C pauses at the beginning of the Acknowledge cycle if the data register has not been written with the next value to be sent (TDRE bit in the I^2C Status Register equal to 1). In this scenario where software is not keeping up with the I^2C bus (TDRE asserted longer than one byte time), the Acknowledge clock cycle for byte *n* is delayed until the data register is written with byte *n+1*, and appears to be grouped with the data clock cycles for byte *n+1*. If either the start or stop bit is set, the I^2C does not pause prior to the Acknowledge cycle because no additional data is sent.

When a Not Acknowledge condition is received during a write (either during the address or data phases), the I^2C Controller generates the Not Acknowledge interrupt (NCKI = 1) and pause until either the stop or start bit is set. Unless the Not Acknowledge was received on the last byte, the data register will already have been written with the next address or data byte to send. In this case the FLUSH bit of the control register should be set at the same time the stop or start bit is set to remove the stale transmit data and enable subsequent transmit interrupts.

When reading data from the slave, the I2C pauses after the data Acknowledge cycle until the receive interrupt is serviced and the RDRF bit of the status register is cleared by reading the I2C Data Register. After the I2C Data Register has been read, the I2C reads the next data byte.

Address Only Transaction with a 7-Bit Address

In situations in which software determines whether a slave with a 7-bit address is responding without sending or receiving data, a transaction can be performed that only consists of an address phase. Figure 26 displays this *address only* transaction to determine if a slave with a 7-bit address will acknowledge.

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

Figure 26 illustrates the format of a 7-bit address-only transaction.

S	Slave Address	W = 0	A/$\overline{A}$	P

Figure 26. 7-Bit Address Only Transaction Format

Observe the following procedure for an address-only transaction to a 7-bit addressed slave:

1. Software asserts the IEN bit in the I2C Control Register.
2. Software asserts the TXI bit of the I2C Control Register to enable Transmit interrupts.
3. The I2C interrupt asserts, because the I2C Data Register is empty (TDRE = 1).
4. Software responds to the TDRE bit by writing a 7-bit Slave address plus write bit (=0) to the I2C Data Register. As an alternative this could be a read operation instead of a write operation.
5. Software sets the start and stop bits of the I2C Control Register and clears the TXI bit.
6. The I2C Controller sends the start condition to the I2C Slave.
7. The I2C Controller loads the I2C Shift Register with the contents of the I2C Data Register.
8. Software polls the stop bit of the I2C Control Register. Hardware deasserts the stop bit when the address only transaction is completed.
9. Software checks the ACK bit of the I2C Status Register. If the slave acknowledged, the ACK bit is equal to 1. If the slave does not acknowledge, the ACK bit is equal to 0.

The NCKI interrupt does not occur in the not acknowledge case because the stop bit was set.

Write Transaction with a 7-Bit Address

Figure 27 displays the data transfer format for a 7-bit addressed slave. Shaded regions indicate data transferred from the I²C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I²C Controller.

S	Slave Address	W = 0	A	Data	A	Data	A	Data	A/A	P/S

Figure 27. 7-Bit Addressed Slave Data Transfer Format

Observe the following procedure for a transmit operation to a 7-bit addressed slave:

1. Software asserts the IEN bit in the I²C Control Register.
2. Software asserts the TXI bit of the I²C Control Register to enable transmit interrupts.
3. The I²C interrupt asserts, because the I²C Data Register is empty.
4. Software responds to the TDRE bit by writing a 7-bit Slave address plus write bit (=0) to the I²C Data Register.
5. Software asserts the start bit of the I²C Control Register.
6. The I²C Controller sends the start condition to the I²C Slave.
7. The I²C Controller loads the I²C Shift Register with the contents of the I²C Data Register.
8. After one bit of address has been shifted out by the SDA signal, the transmit interrupt is asserted (TDRE = 1).
9. Software responds by writing the transmit data into the I²C Data Register.
10. The I²C Controller shifts the rest of the address and write bit out by the SDA signal.
11. If the I²C Slave sends an acknowledge (by pulling the SDA signal Low) during the next High period of SCL the I²C Controller sets the ACK bit in the I²C Status Register. Continue to Step 12.

    If the slave does not acknowledge, the Not Acknowledge interrupt occurs (NCKI bit is set in the Status Register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I²C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remaining steps in this sequence.

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

Address-Only Transaction with a 10-Bit Address

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

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

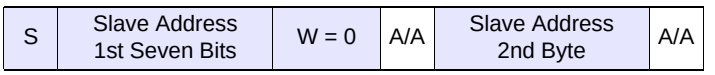


Figure 28. 10-Bit Address Only Transaction Format

Observe the following procedure for an address-only transaction to a 10-bit addressed slave:

1. Software asserts the IEN bit in the I2C Control Register.
2. Software asserts the TXI bit of the I2C Control Register to enable transmit interrupts.
3. The I2C interrupt asserts, because the I2C Data Register is empty (TDRE = 1).
4. Software responds to the TDRE interrupt by writing the first slave address byte. The least-significant bit must be 0 for the write operation.
5. Software asserts the start bit of the I2C Control Register.

6. The I2C Controller sends the start condition to the I2C Slave.
7. The I2C Controller loads the I2C Shift Register with the contents of the I2C Data Register.
8. After one bit of an address is shifted out by the SDA signal, the transmit interrupt is asserted.
9. Software responds by writing the second byte of the address into the contents of the I2C Data Register.
10. The I2C Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. If the I2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL and the I2C Controller sets the ACK bit in the I2C Status Register, continue to Step 12.

    If the slave does not acknowledge the first address byte, the I2C Controller sets the NCKI bit and clears the ACK bit in the I2C Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I2C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remaining steps in this sequence.
12. The I2C Controller loads the I2C Shift Register with the contents of the I2C Data Register (2nd byte of address).
13. The I2C Controller shifts the second address byte out the SDA signal. After the first bit has been sent, the transmit interrupt is asserted.
14. Software responds by setting the stop bit in the I2C Control Register. The TXI bit can be cleared at the same time.
15. Software polls the stop bit of the I2C Control Register. Hardware deasserts the stop bit when the transaction is completed (stop condition has been sent).
16. Software checks the ACK bit of the I2C Status Register. If the slave acknowledged, the ACK bit is equal to 1. If the slave does not acknowledge, the ACK bit is equal to 0. The NCKI interrupt do not occur because the stop bit was set.

Write Transaction with a 10-Bit Address

Figure 29 displays the data transfer format for a 10-bit addressed slave. Shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

S	Slave Address 1st Seven Bits	W = 0	A	Slave Address 2nd Byte	A	Data	A	Data	A/A	P/S

Figure 29. 10-Bit Addressed Slave Data Transfer Format

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

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

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

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

16. If the I2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL, the I2C Controller sets the ACK bit in the I2C Status Register. Continue to Step 17.

    If the slave does not acknowledge the second address byte or one of the data bytes, the I2C Controller sets the NCKI bit and clears the ACK bit in the I2C Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I2C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remainder of this sequence.
17. The I2C Controller shifts the data out by the SDA signal. After the first bit is sent, the transmit interrupt is asserted.
18. If more bytes remain to be sent, return to Step 14.
19. If the last byte is currently being sent, software sets the stop bit of the I2C Control Register (or start bit to initiate a new transaction). In the stop case, software simultaneously clears the TXI bit of the I2C Control Register.
20. The I2C Controller completes transmission of the last data byte on the SDA signal.
21. The slave can either Acknowledge or Not Acknowledge the last byte. Because either the stop or start bit is already set, the NCKI interrupt does not occur.
22. The I2C Controller sends the stop (or restart) condition to the I2C bus and clears the stop (or start) bit.

Read Transaction with a 7-Bit Address

Figure 30 displays the data transfer format for a read operation to a 7-bit addressed slave. The shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

S	Slave Address	R = 1	A	Data	A	Data	A	P/S

Figure 30. Receive Data Transfer Format for a 7-Bit Addressed Slave

Observe the following procedure for a read operation to a 7-bit addressed slave:

1. Software writes the I2C Data Register with a 7-bit Slave address plus the read bit (=1).
2. Software asserts the start bit of the I2C Control Register.
3. If this transfer is a single byte transfer, Software asserts the NAK bit of the I2C Control Register so that after the first byte of data has been read by the I2C Controller, a Not Acknowledge is sent to the I2C Slave.
4. The I2C Controller sends the start condition.

5. The I2C Controller shifts the address and read bit out the SDA signal.
6. If the I2C Slave acknowledges the address by pulling the SDA signal Low during the next High period of SCL, the I2C Controller sets the ACK bit in the I2C Status Register. Continue to Step 7.

   If the slave does not acknowledge, the Not Acknowledge interrupt occurs (NCKI bit is set in the Status Register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the stop bit and clearing the TXI bit. The I2C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remainder of this sequence.
7. The I2C Controller shifts in the byte of data from the I2C Slave on the SDA signal. The I2C Controller sends a Not Acknowledge to the I2C Slave if the NAK bit is set (last byte), else it sends an Acknowledge.
8. The I2C Controller asserts the Receive interrupt (RDRF bit set in the Status Register).
9. Software responds by reading the I2C Data Register which clears the RDRF bit. If there is only one more byte to receive, set the NAK bit of the I2C Control Register.
10. If there are more bytes to transfer, return to Step 7.
11. After the last byte is shifted in, a Not Acknowledge interrupt is generated by the I2C Controller.
12. Software responds by setting the stop bit of the I2C Control Register.
13. A stop condition is sent to the I2C Slave, the stop and NCKI bits are cleared.

Read Transaction with a 10-Bit Address

Figure 31 displays the read transaction format for a 10-bit addressed slave. The shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

S	Slave Address 1st 7 bits	W=0	A	Slave Address 2nd Byte	A	S	Slave Address 1st 7 bits	R=1	A	Data	A	Data	A	P

Figure 31. Receive Data Format for a 10-Bit Addressed Slave

The first seven bits transmitted in the first byte are `11110XX`. The two `XX` bits are the two most significant bits of the 10-bit address. The lowest bit of the first byte transferred is the write control bit.

Observe the following procedure for the data transfer procedure for a read operation to a 10-bit addressed slave:

1. Software writes `11110B` followed by the two address bits and a 0 (write) to the I²C Data Register.
2. Software asserts the start and TXI bits of the I²C Control Register.
3. The I²C Controller sends the start condition.
4. The I²C Controller loads the I²C Shift Register with the contents of the I²C Data Register.
5. After the first bit has been shifted out, a transmit interrupt is asserted.
6. Software responds by writing the lower eight bits of address to the I²C Data Register.
7. The I²C Controller completes shifting of the two address bits and a 0 (write).
8. If the I²C Slave acknowledges the first address byte by pulling the SDA signal Low during the next High period of SCL, the I²C Controller sets the ACK bit in the I²C Status Register. Continue to Step 9.

   If the slave does not acknowledge the first address byte, the I²C Controller sets the NCKI bit and clears the ACK bit in the I²C Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I²C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore following steps).
9. The I²C Controller loads the I²C Shift Register with the contents of the I²C Data Register (second address byte).
10. The I²C Controller shifts out the second address byte. After the first bit is shifted, the I²C Controller generates a transmit interrupt.
11. Software responds by setting the start bit of the I²C Control Register to generate a repeated start and by clearing the TXI bit.
12. Software responds by writing `11110B` followed by the 2-bit Slave address and a 1 (read) to the I²C Data Register.
13. If only one byte is to be read, software sets the NAK bit of the I²C Control Register.
14. After the I²C Controller shifts out the 2nd address byte, the I²C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL, the I²C Controller sets the ACK bit in the I²C Status Register. Continue to Step 15.

    If the slave does not acknowledge the second address byte, the I²C Controller sets the NCKI bit and clears the ACK bit in the I²C Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I²C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remainder of this sequence.
15. The I²C Controller sends the repeated start condition.

16. The I2C Controller loads the I2C Shift Register with the contents of the I2C Data Register (third address transfer).
17. The I2C Controller sends `11110B` followed by the two most significant bits of the slave read address and a 1 (read).
18. The I2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL.

    If the slave were to Not Acknowledge at this point (this should not happen because the slave did acknowledge the first two address bytes), software would respond by setting the stop and flush bits and clearing the TXI bit. The I2C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete; ignore the remainder of this sequence.
19. The I2C Controller shifts in a byte of data from the I2C Slave on the SDA signal. The I2C Controller sends a Not Acknowledge to the I2C Slave if the NAK bit is set (last byte), else it sends an Acknowledge.
20. The I2C Controller asserts the Receive interrupt (RDRF bit set in the Status Register).
21. Software responds by reading the I2C Data Register which clears the RDRF bit. If there is only one more byte to receive, set the NAK bit of the I2C Control Register.
22. If there are one or more bytes to transfer, return to Step 19.
23. After the last byte is shifted in, a Not Acknowledge interrupt is generated by the I2C Controller.
24. Software responds by setting the stop bit of the I2C Control Register.
25. A stop condition is sent to the I2C Slave and the stop and NCKI bits are cleared.

I2C Control Register Definitions

This section defines the features of the following I2C Control registers.

I2C Data Register: see page 129

I2C Status Register: see page 129

I2C Control Register: see page 131

I2C Baud Rate High and Low Byte Registers: see page 132

I2C Diagnostic State Register: see page 133

I2C Diagnostic Control Register: see page 135

I²C Data Register

The I²C Data Register, shown in Table 71, holds the data that is to be loaded into the I²C Shift Register during a write to a slave. This register also holds data that is loaded from the I²C Shift Register during a read from a slave. The I²C Shift Register is not accessible in the Register File address space, but is used only to buffer incoming and outgoing data.

Table 71. I²C Data Register (I2CDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	0							
R/W	R/W							
Address	F50H							

I²C Status Register

The read-only I²C Status Register, shown in Table 72, indicates the status of the I²C Controller.

Table 72. I²C Status Register (I2CSTAT)

Bit	7	6	5	4	3	2	1	0
Field	TDRE	RDRF	ACK	10B	RD	TAS	DSS	NCKI
RESET	1	0						
R/W	R							
Address	F51H							

Bit	Description
[7] TDRE	**Transmit Data Register Empty** When the I²C Controller is enabled, this bit is 1 when the I²C Data Register is empty. When this bit is set, an interrupt is generated if the TXI bit is set, except when the I²C Controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit is cleared by writing to the I2CDATA Register.
[6] RDRF	**Receive Data Register Full** This bit is set = 1 when the I²C Controller is enabled and the I²C Controller has received a byte of data. When asserted, this bit causes the I²C Controller to generate an interrupt. This bit is cleared by reading the I²C Data Register (unless the read is performed using execution of the OCD's Read Register command).

Bit	Description (Continued)
[5] ACK	**Acknowledge** This bit indicates the status of the Acknowledge for the last byte transmitted or received. When set, this bit indicates that an Acknowledge occurred for the last byte transmitted or received. This bit is cleared when IEN = 0 or when a Not Acknowledge occurred for the last byte transmitted or received. It is not reset at the beginning of each transaction and is not reset when this register is read. **Caution:** When making decisions based on this bit within a transaction, software cannot determine when the bit is updated by hardware. In the case of write transactions, the I^2C pauses at the beginning of the Acknowledge cycle if the next transmit data or address byte has not been written (TDRE = 1) and STOP and start = 0. In this case the ACK bit is not updated until the transmit interrupt is serviced and the Acknowledge cycle for the previous byte completes. For examples on usage of the ACK bit, see the Address Only Transaction with a 7-Bit Address section on page 120 and the Address-Only Transaction with a 10-Bit Address section on page 122.
[4] 10B	**10-Bit Address** This bit indicates whether a 10-bit or 7-bit address is being transmitted. After the start bit is set, if the five most-significant bits of the address are 11110B, this bit is set. When set, it is reset after the first byte of the address has been sent.
[3] RD	**Read** This bit indicates the direction of transfer of the data. It is active High during a read. The status of this bit is determined by the least-significant bit of the I^2C Shift Register after the start bit is set.
[2] TAS	**Transmit Address State** This bit is active High while the address is being shifted out of the I^2C Shift Register.
[1] DSS	**Data Shift State** This bit is active High while data is being shifted to or from the I^2C Shift Register.
[0] NCKI	**NACK Interrupt** This bit is set High when a Not Acknowledge condition is received or sent and neither the start nor the stop bit is active. When set, this bit generates an interrupt that can only be cleared by setting the start or stop bit, allowing you to specify whether you want to perform a stop or a repeated start.

I²C Control Register

The I²C Control Register, shown in Table 73, enables I²C operation.

Table 73. I²C Control Register (I2CCTL)

Bit	7	6	5	4	3	2	1	0
Field	IEN	START	STOP	BIRQ	TXI	NAK	FLUSH	FILTEN
RESET	0							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	W	R/W
Address	F52H							

Bit	Description
[7] IEN	**I²C Enable** 1 = The I²C transmitter and receiver are enabled. 0 = The I²C transmitter and receiver are disabled.
[6] START	**Send Start Condition** This bit sends the start condition. After it is asserted, it is cleared by the I²C Controller after it sends the START condition or if the IEN bit is deasserted. If this bit is 1, it cannot be cleared to 0 by writing to the register. After this bit is set, the Start condition is sent if there is data in the I²C Data or I²C Shift Register. If there is no data in one of these registers, the I²C Controller waits until the data register is written. If this bit is set while the I²C Controller is shifting out data, it generates a start condition after the byte shifts and the acknowledge phase completes. If the stop bit is also set, it also waits until the stop condition is sent before sending the start condition.
[5] STOP	**Send Stop Condition** This bit causes the I²C Controller to issue a stop condition after the byte in the I²C Shift Register has completed transmission or after a byte is received in a receive operation. After it is set, this bit is reset by the I²C Controller after a stop condition is sent or by deasserting the IEN bit. If this bit is 1, it cannot be cleared to 0 by writing to the register.
[4] BIRQ	**Baud Rate Generator Interrupt Request** This bit allows the I²C Controller to be used as an additional timer when the I²C Controller is disabled. This bit is ignored when the I²C Controller is enabled. 1 = An interrupt occurs every time the BRG counts down to 1. 0 = No BRG interrupt occurs.
[3] TXI	**Enable TDRE interrupts** This bit enables the transmit interrupt when the I²C Data Register is empty (TDRE = 1). 1 = transmit interrupt (and DMA transmit request) is enabled. 0 = transmit interrupt (and DMA transmit request) is disabled.
[2] NAK	**Send NAK** This bit sends a Not Acknowledge condition after the next byte of data is read from the I²C Slave. After it is asserted, it is deasserted after a Not Acknowledge is sent or the IEN bit is deasserted. If this bit is 1, it cannot be cleared to 0 by writing to the register.

Bit	Description (Continued)
[1] FLUSH	**Flush Data** Setting this bit to 1 clears the I²C Data Register and sets the TDRE bit to 1. This bit allows flushing of the I²C Data Register when a Not Acknowledge interrupt is received after the data has been sent to the I²C Data Register. Reading this bit always returns 0.
[0] FILTEN	**I²C Signal Filter Enable** This bit enables low-pass digital filters on the SDA and SCL input signals. These filters reject any input pulse with periods less than a full system clock cycle. The filters introduce a 3-system clock cycle latency on the inputs. 1 = Low-pass filters are enabled. 0 = Low-pass filters are disabled.

I²C Baud Rate High and Low Byte Registers

The I²C Baud Rate High and Low Byte registers, shown in Tables 74 and 75, combine to form a 16-bit reload value, BRG[15:0], for the I²C Baud Rate Generator. When configured as a general purpose timer, the interrupt interval is calculated using the following equation:

Interrupt Interval (s) = System Clock Period (s) ×BRG[15:0]

Table 74. I²C Baud Rate High Byte Register (I2CBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFH							
R/W	R/W							
Address	F53H							

Bit	Description
[7:0] BRH	**I²C Baud Rate High Byte** Most significant byte, BRG[15:8], of the I²C Baud Rate Generator's reload value. **Note:** If the DIAG bit in the I²C Diagnostic Control Register is set to 1, a read of the I2CBRH Register returns the current value of the I²C Baud Rate Counter[15:8].

Table 75. I²C Baud Rate Low Byte Register (I2CBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFH							
R/W	R/W							
Address	F54H							

Bit	Description
[7:0] BRL	**I²C Baud Rate Low Byte** Least significant byte, BRG[7:0], of the I²C Baud Rate Generator's reload value. **Note:** If the DIAG bit in the I²C Diagnostic Control Register is set to 1, a read of the I2CBRL Register returns the current value of the I²C Baud Rate Counter [7:0].

I²C Diagnostic State Register

The I²C Diagnostic State Register, shown in Table 76, provides observability into the internal state. This register is read-only; it is used for I²C diagnostics and manufacturing test purposes.

Table 76. I²C Diagnostic State Register (I2CDST)

Bit	7	6	5	4	3	2	1	0
Field	SCLIN	SDAIN	STPCNT	TXRXSTATE				
RESET	X		0					
R/W	R							
Address	F55H							

Bit	Description
[7] SCLIN	**Serial Clock Input** Value of the Serial Clock input signal.
[6] SDAIN	**Serial Data Input** Value of the Serial Data input signal.
[5] STPCNT	**Stop Count** Value of the internal Stop Count control signal.

Bit	Description (Continued)	
[4:0] TXRXSTATE	**Internal State** Value of the internal I^2C state machine.	
	TXRXSTATE	**State Description**
	0_0000	Idle State.
	0_0001	Start State.
	0_0010	Send/Receive data bit 7.
	0_0011	Send/Receive data bit 6.
	0_0100	Send/Receive data bit 5.
	0_0101	Send/Receive data bit 4.
	0_0110	Send/Receive data bit 3.
	0_0111	Send/Receive data bit 2.
	0_1000	Send/Receive data bit 1.
	0_1001	Send/Receive data bit 0.
	0_1010	Data Acknowledge State.
	0_1011	Second half of data Acknowledge State used only for not acknowledge.
	0_1100	First part of stop state.
	0_1101	Second part of stop state.
	0_1110	10-bit addressing: Acknowledge State for 2nd address byte; 7-bit addressing: Address Acknowledge State.
	0_1111	10-bit address: Bit 0 (Least significant bit) of 2nd address byte; 7-bit address: Bit 0 (Least significant bit) (R/W) of address byte.
	1_0000	10-bit addressing: Bit 7 (Most significant bit) of 1st address byte.
	1_0001	10-bit addressing: Bit 6 of 1st address byte.
	1_0010	10-bit addressing: Bit 5 of 1st address byte.
	1_0011	10-bit addressing: Bit 4 of 1st address byte.
	1_0100	10-bit addressing: Bit 3 of 1st address byte.
	1_0101	10-bit addressing: Bit 2 of 1st address byte.
	1_0110	10-bit addressing: Bit 1 of 1st address byte.
	1_0111	10-bit addressing: Bit 0 (R/W) of 1st address byte.
	1_1000	10-bit addressing: Acknowledge state for 1st address byte.
	1_1001	10-bit addressing: Bit 7 of 2nd address byte; 7-bit addressing: Bit 7 of address byte.
	1_1010	10-bit addressing: Bit 6 of 2nd address byte; 7-bit addressing: Bit 6 of address byte.
	1_1011	10-bit addressing: Bit 5 of 2nd address byte; 7-bit addressing: Bit 5 of address byte.
	1_1100	10-bit addressing: Bit 4 of 2nd address byte; 7-bit addressing: Bit 4 of address byte.
	1_1101	10-bit addressing: Bit 3 of 2nd address byte; 7-bit addressing: Bit 3 of address byte.
	1_1110	10-bit addressing: Bit 2 of 2nd address byte; 7-bit addressing: Bit 2 of address byte.
	1_1111	10-bit addressing: Bit 1 of 2nd address byte; 7-bit addressing: Bit 1 of address byte.

I²C Diagnostic Control Register

The I²C Diagnostic Register, shown in Table 77, provides control over diagnostic modes. This register is a read/write register used for I²C diagnostics.

Table 77. I²C Diagnostic Control Register (I2CDIAG)

Bit	7	6	5	4	3	2	1	0
Field	Reserved							DIAG
RESET	0							
R/W	R							R/W
Address	F56H							

Bit	Description
[7:1]	**Reserved** These bits are reserved and must be programmed to 0000000.
[0] DIAG	**Diagnostic Control Bit** Selects the read-back value of the Baud Rate Reload registers. 0 = Normal Mode. Reading the Baud Rate High and Low Byte registers returns the baud rate reload value. 1 = Diagnostic Mode. Reading the Baud Rate High and Low Byte registers returns the baud rate counter value.

Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) converts an analog input signal to a 10-bit binary number. The features of the sigma-delta ADC include:

- Five analog input sources are multiplexed with GPIO ports
- Interrupt upon conversion complete
- Internal voltage reference generator

The ADC is available only in the Z8F0822, Z8F0821, Z8F0422, Z8F0421, Z8R0822, Z8R0821, Z8R0422 and Z8R0421 devices.

Architecture

Figure 32 displays the three major functional blocks (converter, analog multiplexer and voltage reference generator) of the ADC. The ADC converts an analog input signal to its digital representation. The five-input analog multiplexer selects one of the five analog input sources. The ADC requires an input reference voltage for the conversion. The voltage reference for the conversion can be input through the external V_{REF} pin or generated internally by the voltage reference generator.

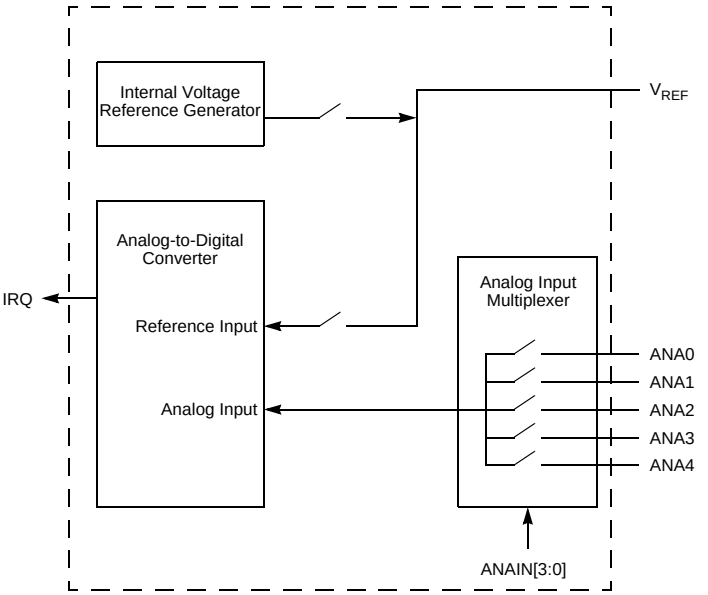


Figure 32. Analog-to-Digital Converter Block Diagram

Operation

This section describes the operational aspects of the ADC's power-down and conversion features.

Automatic Power-Down

If the ADC is idle (no conversions in progress) for 160 consecutive system clock cycles, portions of the ADC are automatically powered down. From this powered-down state, the ADC requires 40 system clock cycles to power up. The ADC powers up when a conversion is requested via the ADC Control Register.

Single-Shot Conversion

When configured for single-shot conversion, the ADC performs a single analog-to-digital conversion on the selected analog input channel. After completion of the conversion, the ADC shuts down. Observe the following procedure for setting up the ADC and initiating a single-shot conversion:

1. Enable the appropriate analog inputs by configuring the GPIO pins for alternate function. This configuration disables the digital input and output drivers.
2. Write to the ADC Control Register to configure the ADC and begin the conversion. The following bit fields in the ADC Control Register are written simultaneously:
   - Write to the ANAIN[3:0] field to select one of the 5 analog input sources
   - Clear CONT to 0 to select a single-shot conversion
   - Write to the $\overline{\text{VREF}}$ bit to enable or disable the internal voltage reference generator
   - Set CEN to 1 to start the conversion
3. CEN remains 1 while the conversion is in progress. A single-shot conversion requires 5129 system clock cycles to complete. If a single-shot conversion is requested from an ADC powered-down state, the ADC uses 40 additional clock cycles to power-up before beginning the 5129 cycle conversion.
4. When the conversion is complete, the ADC control logic performs the following operations:
   - 10-bit data result written to {ADCD_H[7:0], ADCD_L[7:6]}
   - CEN resets to 0 to indicate the conversion is complete
   - An interrupt request is sent to the Interrupt Controller
5. If the ADC remains idle for 160 consecutive system clock cycles, it is automatically powered-down.

Continuous Conversion

When configured for continuous conversion, the ADC continuously performs an analog-to-digital conversion on the selected analog input. Each new data value over-writes the previous value stored in the ADC Data registers. An interrupt is generated after each conversion.

Caution: In CONTINUOUS Mode, ensure that ADC updates are limited by the input signal bandwidth of the ADC and the latency of the ADC and its digital filter. Step changes at the input are not seen at the next output from the ADC. The response of the ADC (in all modes) is limited by the input signal bandwidth and the latency.

Observe the following procedure for setting up the ADC and initiating continuous conversion:

1. Enable the appropriate analog input by configuring the GPIO pins for alternate function. This disables the digital input and output driver.

2. Write to the ADC Control Register to configure the ADC for continuous conversion. The bit fields in the ADC Control Register can be written simultaneously:
   - Write to the ANAIN[3:0] field to select one of the 5 analog input sources.
   - Set CONT to 1 to select continuous conversion.
   - Write to the $\overline{\text{VREF}}$ bit to enable or disable the internal voltage reference generator.
   - Set CEN to 1 to start the conversions.
3. When the first conversion in continuous operation is complete (after 5129 system clock cycles, plus the 40 cycles for power-up, if necessary), the ADC control logic performs the following operations:
   - CEN resets to 0 to indicate the first conversion is complete. CEN remains 0 for all subsequent conversions in continuous operation.
   - An interrupt request is sent to the Interrupt Controller to indicate the conversion is complete.
4. Thereafter, the ADC writes a new 10-bit data result to {ADCD_H[7:0], ADCD_L[7:6]} every 256 system clock cycles. An interrupt request is sent to the Interrupt Controller when each conversion is complete.
5. To disable continuous conversion, clear the CONT bit in the ADC Control Register to 0.

ADC Control Register Definitions

This section defines the features of the following ADC Control registers.

ADC Control Register: see page 139

ADC Data High Byte Register: see page 141

ADC Data Low Bits Register: see page 142

ADC Control Register

The ADC Control Register, shown in Table 78, selects the analog input channel and initiates the analog-to-digital conversion.

Table 78. ADC Control Register (ADCCTL)

Bit	7	6	5	4	3	2	1	0
Field	CEN	Reserved	VREF	CONT	ANAIN[3:0]			
RESET	0		1	0				
R/W	R/W							
Address	F70H							

Bit	Description
[7] CEN	**Conversion Enable** 0 = Conversion is complete. Writing a 0 produces no effect. The ADC automatically clears this bit to 0 when a conversion has been completed. 1 = Begin conversion. Writing a 1 to this bit starts a conversion. If a conversion is already in progress, the conversion restarts. This bit remains 1 until the conversion is complete.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] VREF	**Voltage Reference** 0 = Internal reference generator enabled. The V_{REF} pin must remain unconnected or capacitively coupled to analog ground (AV_{SS}). 1 = Internal voltage reference generator disabled. An external voltage reference must be provided through the V_{REF} pin.
[4] CONT	**Conversion** 0 = SINGLE-SHOT conversion. ADC data is output one time at completion of the 5129 system clock cycles. 1 = Continuous conversion. ADC data updated every 256 system clock cycles.
[3] ANAIN[3:0]	**Analog Input Select** These bits select the analog input for conversion. Not all Port pins in this list are available in all packages for Z8 Encore! XP® F0822 Series. See the Signal and Pin Descriptions chapter on page 7 for information regarding the port pins available with each package style. Do not enable unavailable analog inputs. 0000 = ANA0. 0001 = ANA1. 0010 = ANA2. 0011 = ANA3. 0100 = ANA4. 0101 = Reserved. 011X = Reserved. 1XXX = Reserved.

ADC Data High Byte Register

The ADC Data High Byte Register, shown in Table 79, contains the upper eight bits of the 10-bit ADC output. During a SINGLE-SHOT conversion, this value is invalid. Access to the ADC Data High Byte Register is read-only. The full 10-bit ADC result is furnished by {ADCD_H[7:0], ADCD_L[7:6]}. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

Table 79. ADC Data High Byte Register (ADCD_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCD_H							
RESET	X							
R/W	R							
Address	F72H							

Bit	Description
[7:0] ADCD_H	**ADC Data High Byte** This byte contains the upper eight bits of the 10-bit ADC output. These bits are not valid during a single-shot conversion. During a continuous conversion, the last conversion output is held in this register. These bits are undefined after a Reset.

ADC Data Low Bits Register

The ADC Data Low Bits Register, shown in Table 80, contains the lower two bits of the conversion value. The data in the ADC Data Low Bits Register is latched each time the ADC Data High Byte Register is read. Reading this register always returns the lower two bits of the conversion last read into the ADC High Byte Register. Access to the ADC Data Low Bits Register is read-only. The full 10-bit ADC result is furnished by {ADCD_H[7:0], ADCD_L[7:6]}.

Table 80. ADC Data Low Bits Register (ADCD_L)

Bit	7	6	5	4	3	2	1	0
Field	ADCD_L		Reserved					
RESET	X							
R/W	R							
Address	F73H							

Bit	Description
[7:6] ADCD_L	**ADC Data Low Bits** These are the least significant two bits of the 10-bit ADC output. These bits are undefined after a Reset.
[5:0]	**Reserved** These bits are reserved and are always undefined.

Flash Memory

The products in the Z8 Encore! XP® F0822 Series feature either 8KB (8192) or 4KB (4096) bytes of Flash memory with Read/Write/Erase capability. Flash memory is programmed and erased in-circuit by either user code or through the OCD.

The Flash memory array is arranged in 512-byte per page. The 512-byte page is the minimum Flash block size that can be erased. Flash memory is divided into eight sectors which is protected from programming and erase operations on a per sector basis.

Table 81 describes the Flash memory configuration for each device in the Z8F082xfamily. Table 82 lists the sector address ranges. Figure 33 displays the Flash memory arrangement.

Table 81. Flash Memory Configurations

Part Number	Flash Size	Number of Pages	Flash Memory Addresses	Sector Size	Number of Sectors	Pages per Sector
Z8F08xx	8KB (8192)	16	0000H–1FFFH	1KB (1024)	8	2
Z8F04xx	4KB (4096)	8	0000H–0FFFH	0.5KB (512)	8	1

Table 82. Flash Memory Sector Addresses

	Flash Sector Address Ranges	
Sector Number	Z8F04xx	Z8F08xx
0	0000H–01FFH	0000H–03FFH
1	0200H–03FFH	0400H–07FFH
2	0400H–05FFH	0800H–0BFFH
3	0600H–07FFH	0C00H–0FFFH
4	0800H–09FFH	1000H–13FFH
5	0A00H–0BFFH	1400H–17FFH
6	0C00H–0DFFH	1800H–1BFFH
7	0E00H–0FFFH	1C00H–1FFFH

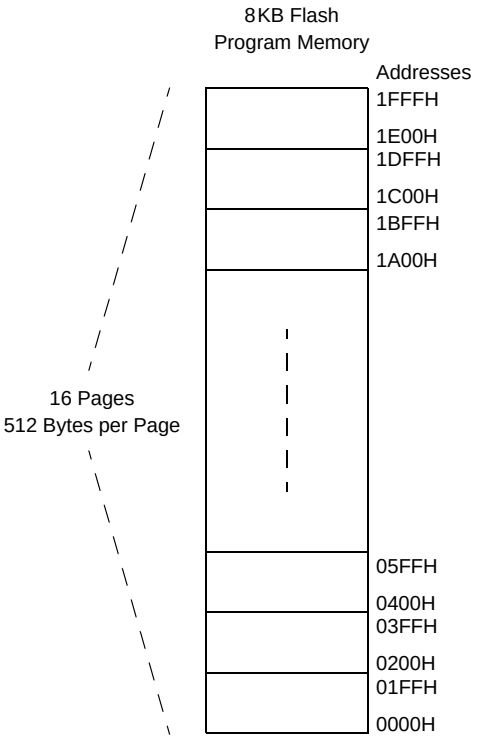


Figure 33. Flash Memory Arrangement

Information Area

Table 83 describes the Z8 Encore! XP® F0822 Series Information Area. This 512-byte Information Area is accessed by setting bit 7 of the Page Select Register to 1. When access is enabled, the Information Area is mapped into Flash memory and overlays the 512 bytes at addresses FE00H to FFFFH. When the Information Area access is enabled, LDC instructions return data from the Information Area. CPU instruction fetches always comes from Flash memory regardless of the Information Area access bit. Access to the Information Area is read-only.

Table 83. Z8 Encore! XP® F0822 Series Information Area Map

Flash Memory Address (Hex)	Function
FE00H–FE3FH	Reserved
FE40H–FE53H	**Part Number** 20-character ASCII alphanumeric code Left-justified and filled with zeros
FE54H–FFFFH	Reserved

Operation

The Flash Controller provides the proper signals and timing for the Byte Programming, Page Erase, and Mass Erase functions within Flash memory. The Flash Controller contains a protection mechanism, using the Flash Control Register (FCTL), to prevent accidental programming or erasure. The following subsections provide details about the various operations (Lock, Unlock, Sector Protect, Byte Programming, Page Erase and Mass Erase).

Timing Using the Flash Frequency Registers

Before performing a program or erase operation in Flash memory, you must first configure the Flash Frequency High and Low Byte registers. The Flash Frequency registers allow programming and erasure of Flash memory with system clock frequencies ranging from 20kHz through 20MHz (the valid range is limited to the device operating frequencies).

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 Flash Frequency value must contain the system clock frequency in kHz. This value is calculated using the following equation:

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

Caution: Flash programming and erasure are not supported for system clock frequencies below 20kHz, above 20MHz, or outside of the device operating frequency range. The Flash Frequency High and Low Byte registers must be loaded with the correct value to ensure proper Flash programming and erase operations.

Flash Read Protection

The user code contained within Flash memory can be protected from external access. Programming the Flash Read Protect option bit prevents reading of user code by the OCD or by using the Flash Controller Bypass mode. For more information, see the Option Bits chapter on page 155 and the On-Chip Debugger chapter on page 158.

Flash Write/Erase Protection

Z8 Encore! XP® F0822 Series provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by the Flash Controller unlock mechanism, the Flash Sector Protect Register, and the Flash Write Protect option bit.

Flash Controller Unlock Mechanism

At Reset, the Flash Controller locks to prevent accidental program or erasure of Flash memory. To program or erase Flash memory, the Flash Controller must be unlocked. After unlocking the Flash Controller, the Flash can be programmed or erased. Any value written by user code to the Flash Control Register or Page Select Register out of sequence locks the Flash Controller.

Observe the following procedure to unlock the Flash Controller from user code:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write the page to be programmed or erased to the Page Select Register.
3. Write the first unlock command 73H to the Flash Control Register.
4. Write the second unlock command 8CH to the Flash Control Register.
5. Rewrite the page written in Step 2 to the Page Select Register.

Flash Sector Protection

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

The Flash Sector Protect Register shares its Register File address with the Page Select Register. The Flash Sector Protect Register is accessed by writing the Flash Control Register with 5EH. After the Flash Sector Protect Register is selected, it can be accessed at the Page Select Register address. When the user code writes the Flash Sector Protect Register, bits can only be set to 1. Sectors can be protected, but not unprotected, using register write operations. Writing a value other than 5EH to the Flash Control Register deselects the Flash Sector Protect Register and reenables access to the Page Select Register.

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

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

Flash Write Protection Option Bit

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

Byte Programming

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

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

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

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

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

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

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

4. Write the second unlock command 8CH to the Flash Control Register.
5. Rewrite the page written in Step 2 to the Page Select Register.
6. Write Flash memory using LDC or LDCI instructions to program Flash memory.
7. Repeat Step 6 to program additional memory locations on the same page.
8. Write 00H to the Flash Control Register to lock the Flash Controller.

Page Erase

Flash memory can be erased one page (512 bytes) at a time. Page Erasing the Flash memory sets all bytes in that page to the value FFH. The Page Select Register identifies the page to be erased. While the Flash Controller executes the Page Erase operation, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. The eZ8 CPU resumes operation after the Page Erase operation completes. Interrupts that occur when the Page Erase operation is in progress are serviced after the Page Erase operation is complete. When the Page Erase operation is complete, the Flash Controller returns to its locked state. Only pages located in unprotected sectors can be erased.

Observe the following procedure to perform a Page Erase operation:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write the page to be erased to the Page Select Register.
3. Write the first unlock command 73H to the Flash Control Register.
4. Write the second unlock command 8CH to the Flash Control Register.
5. Rewrite the page written in Step 2 to the Page Select Register.
6. Write the Page Erase command 95H to the Flash Control Register.

Mass Erase

Flash memory cannot be mass-erased by user code.

Flash Controller Bypass

The Flash Controller can be bypassed and the control signals for Flash memory brought out to the GPIO pins. Bypassing the Flash Controller allows faster programming algorithms by controlling the Flash programming signals directly.

Zilog recommends Flash Controller Bypass functionality for gang programming applications and for large-volume customers who do not require in-circuit programming of Flash memory.

For more information about bypassing the Flash Controller, refer to the Third Party Flash Programming Support for Z8 Encore! MCU Application Note (AN0117), available for download at www.zilog.com.

Flash Controller Behavior in Debug Mode

The following changes in behavior of the Flash Controller occur when the Flash Controller is accessed using the OCD:

- The Flash Write Protect option bit is ignored
- The Flash Sector Protect Register is ignored for programming and erase operations
- Programming operations are not limited to the page selected in the Page Select Register
- Bits in the Flash Sector Protect Register can be written to 1 or 0
- The second write of the Page Select Register to unlock the Flash Controller is not necessary
- The Page Select Register is written when the Flash Controller is unlocked
- The Mass Erase command is enabled

Flash Control Register Definitions

This section defines the features of the following Flash Control registers.

Flash Control Register: see page 150

Flash Status Register: see page 151

Page Select Register: see page 152

Flash Sector Protect Register: see page 152

Flash Frequency High and Low Byte Registers: see page 153

Flash Control Register

The Flash Control Register, shown in Table 84, is used to unlock the Flash Controller for programming and erase operations, or to select the Flash Sector Protect Register. The write-only Flash Control Register shares its Register File address with the read-only Flash Status Register.

Table 84. Flash Control Register (FCTL)

Bit	7	6	5	4	3	2	1	0
Field	FCMD							
RESET	0							
R/W	W							
Address	FF8H							

Bit	Description
[7:0] FCMD	**Flash Command*** 73H = First unlock command. 8CH = Second unlock command. 95H = Page erase command. 63H = Mass erase command. 5EH = Flash Sector Protect Register select.

Note: *All other commands, or any command out of sequence, lock the Flash Controller.

Flash Status Register

The Flash Status Register, shown in Table 85, indicates the current state of the Flash Controller. This register can be read at any time. The read-only Flash Status Register shares its Register File address with the write-only Flash Control Register.

Table 85. Flash Status Register (FSTAT)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		FSTAT					
RESET	0							
R/W	R							
Address	FF8H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:0] FSTAT	**Flash Controller Status** 00_0000 = Flash Controller locked. 00_0001 = First unlock command received. 00_0010 = Second unlock command received. 00_0011 = Flash Controller unlocked. 00_0100 = Flash Sector Protect Register selected. 00_1xxx = Program operation in progress. 01_0xxx = Page erase operation in progress. 10_0xxx = Mass erase operation in progress.

Page Select Register

The Page Select (FPS) Register, shown in Table 86, selects the Flash memory page to be erased or programmed. Each Flash page contains 512 bytes of Flash memory. During a Page Erase operation, all Flash memory locations with the 7 most significant bits of the address provided by the PAGE field are erased to FFH.

The Page Select Register shares its Register File address with the Flash Sector Protect Register. The Page Select Register cannot be accessed when the Flash Sector Protect Register is enabled.

Table 86. Page Select Register (FPS)

Bit	7	6	5	4	3	2	1	0
Field	INFO_EN	PAGE						
RESET	0							
R/W	R/W							
Address	FF9H							

Bit	Description
[7] INFO_EN	**Information Area Enable** 0 = Information Area is not selected. 1 = Information Area is selected. The Information area is mapped into the Flash memory address space at addresses FE00H through FFFFH.
[6:0] PAGE	**Page Select** This 7-bit field selects the Flash memory page for Programming and Page Erase operations. Flash memory address[15:9] = PAGE[6:0].

Flash Sector Protect Register

The Flash Sector Protect Register, shown in Table 87, protects Flash memory sectors from being programmed or erased from user code. The Flash Sector Protect Register shares its Register File address with the Page Select Register. The Flash Sector Protect Register can be accessed only after writing the Flash Control Register with 5EH.

User code can only write bits in this register to 1 (bits cannot be cleared to 0 by user code). To determine the appropriate Flash memory sector address range and sector number for your F0822 Series product, please refer to Table 82 on page 143.

Table 87. Flash Sector Protect Register (FPROT)

Bit	7	6	5	4	3	2	1	0
Field	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
RESET	0							
R/W	R/W*							
Address	FF9H							

Note: *R/W = this register is accessible for read operations, but can only be written to 1 (via user code).

Bit	Description
[7:0] SECT*n*	**Sector Protect** 0 = Sector *n* can be programmed or erased from user code. 1 = Sector *n* is protected and cannot be programmed or erased from user code. User code can only write bits from 0 to 1.

Note: *n* indicates bits in the range [7:0].

Flash Frequency High and Low Byte Registers

The Flash Frequency High and Low Byte registers, shown in Tables 88 and 89, combine to form a 16-bit value, FFREQ, to control timing for Flash program and erase operations. The 16-bit Flash Frequency registers must be written with the system clock frequency in kHz for Program and Erase operations. The Flash Frequency value is calculated using the following equation:

$$FFREQ[15:0] = \{FFREQH[7:0],FFREQL[7:0]\} = \frac{\text{System Clock Frequency}}{1000}$$

Caution: Flash programming and erasure is not supported for system clock frequencies below 20 kHz, above 20 MHz, or outside of the valid operating frequency range for the device. The Flash Frequency High and Low Byte registers must be loaded with the correct value to ensure proper program and erase times.

Table 88. Flash Frequency High Byte Register (FFREQH)

Bit	7	6	5	4	3	2	1	0
Field	FFREQH							
RESET	0							
R/W	R/W							
Address	FFAH							

Table 89. Flash Frequency Low Byte Register (FFREQL)

Bit	7	6	5	4	3	2	1	0
Field	FFREQL							
RESET	0							
R/W	R/W							
Address	FFBH							

Bit	Description
[7:0] FFREQH, FFREQL	**Flash Frequency High and Low Bytes** These 2 bytes, {FFREQH[7:0], FFREQL[7:0]}, contain the 16-bit Flash Frequency value.

Option Bits

Option bits allow user configuration of certain aspects of Z8 Encore! XP® F0822 Series operation. The feature configuration data is stored in Flash memory and read during Reset. Features available for control through the option bits are:

- Watchdog Timer time-out response selection–interrupt or Reset
- Watchdog Timer enabled at Reset
- The ability to prevent unwanted read access to user code in Flash memory
- The ability to prevent accidental programming and erasure of all or a portion of the user code in Flash memory
- Voltage Brown-Out configuration is always enabled or disabled during STOP Mode to reduce STOP Mode power consumption
- Oscillator Mode selection for high-, medium- and low-power crystal oscillators, or external RC oscillator

Operation

This section describes the type and configuration of the programmable Flash option bits.

Option Bit Configuration By Reset

During any reset operation (System Reset, Reset or Stop Mode Recovery), the option bits are automatically read from Flash memory and written to Option Configuration registers. The Option Configuration registers control operation of the devices within the Z8 Encore! XP® F0822 Series. Option bit control is established before the device exits Reset and the eZ8 CPU begins code execution. The Option Configuration registers are not part of the Register File and are not accessible for read or write access. Each time the option bits are programmed or erased, the device must be Reset for the change to take place (Flash version only).

Option Bit Address Space

The first two bytes of Flash memory at addresses 0000H (shown in Table 90) and 0001H (shown in Table 91) are reserved for the user-programmable option bits. The byte at Program memory address 0000H configures user options. The byte at Flash memory address 0001H is reserved for future use and must remain in its unprogrammed state.

Flash Memory Address 0000H

Table 90. Option Bits at Flash Memory Address 0000H for 8K Series Flash Devices

Bit	7	6	5	4	3	2	1	0
Field	WDT_RES	WDT_AO	OSC_SEL[1:0]		VBO_AO	RP	Reserved	FWP
RESET	U							
R/W	R/W							
Address	Program Memory 0000H							

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

Bit	Description
[7] WDT_RES	**Watchdog Timer Reset** 0 = Watchdog Timer time-out generates an interrupt request. Interrupts must be globally enabled for the eZ8 CPU to acknowledge the interrupt request. 1 = Watchdog Timer time-out causes a Reset. This setting is the default for unprogrammed (erased) Flash.
[6] WDT_AO	**Watchdog Timer Always On** 0 = Watchdog Timer is automatically enabled upon application of system power. Watchdog Timer can not be disabled. 1 = Watchdog Timer is enabled upon execution of the WDT instruction. After it is enabled, the Watchdog Timer can only be disabled by a Reset or Stop Mode Recovery. This setting is the default for unprogrammed (erased) Flash.
[5:4] OSC_SEL[1:0]	**OSCILLATOR Mode Selection** 00 = On-chip oscillator configured for use with external RC networks (<4MHz). 01 = Minimum power for use with very-low-frequency crystals (32kHz to 1.0MHz). 10 = Medium power for use with medium frequency crystals or ceramic resonators (0.5MHz to 10.0MHz). 11 = Maximum power for use with high-frequency crystals (8.0MHz to 20.0MHz). This setting is the default for unprogrammed (erased) Flash.
[3] VBO_AO	**Voltage Brown-Out Protection Always On** 0 = Voltage Brown-Out Protection is disabled in STOP Mode to reduce total power consumption. 1 = Voltage Brown-Out Protection is always enabled including during STOP Mode. This setting is the default for unprogrammed (erased) Flash.
[2] RP	**Read Protect** 0 = User program code is inaccessible. Limited control features are available through the OCD. 1 = User program code is accessible. All OCD commands are enabled. This setting is the default for unprogrammed (erased) Flash.

Note: *Applies only to the Flash versions of the F0822 Series of devices.

Bit	Description (Continued)
[1]	**Reserved** This bit is reserved and must always be 1.
[0] FWP	**Flash Write Protect*** These two option bits combine to provide three levels of Program memory protection. 0 = Programming, Page Erase, and Mass Erase using User Code is disabled. Mass Erase is available through the OCD. 1 = Programming and Page Erase are enabled for all of Flash program memory.

Note: *Applies only to the Flash versions of the F0822 Series of devices.

Flash Memory Address 0001H

Table 91. Options Bits at Flash Memory Address 0001H

Bit	7	6	5	4	3	2	1	0
Field	Reserved							
RESET	U							
R/W	R/W							
Address	Program Memory 0001H							

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

Bit	Description
[7:0]	**Reserved** These option bits are reserved and must always be 1. This setting is the default for unprogrammed (erased) Flash.

On-Chip Debugger

Z8 Encore! XP® F0822 Series products have an integrated On-Chip Debugger (OCD) that provides advanced debugging features, including:

- Reading and writing of the Register File
- Reading and (Flash version only) writing of Program and Data Memory
- Setting of breakpoints
- Executing eZ8 CPU instructions

Architecture

The OCD consists of four primary functional blocks: transmitter, receiver, autobaud generator, and debug controller. Figure 34 displays the architecture of the OCD.

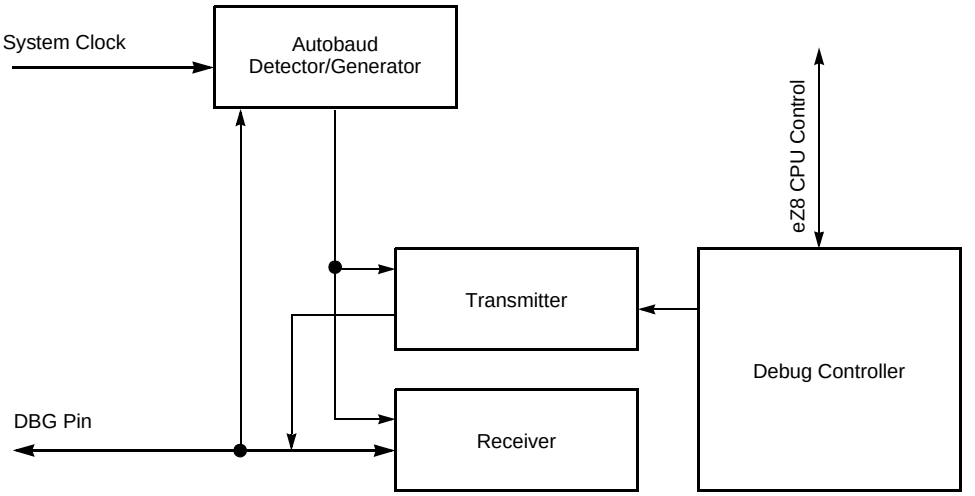


Figure 34. On-Chip Debugger Block Diagram

Operation

The following section describes the operation of the OCD.

OCD Interface

The OCD uses the DBG pin for communication with an external host. This one-pin interface is a bidirectional open-drain interface that transmits and receives data. Data transmission is half-duplex, in that transmit and receive cannot occur simultaneously. The serial data on the DBG pin is sent using the standard asynchronous data format defined in RS-232. This pin can interface the Z8 Encore! XP® F0822 Series products to the serial port of a host PC using minimal external hardware.Two different methods for connecting the DBG pin to an RS-232 interface are shown in Figures 35 and 36.

Caution: For operation of the OCD, all power pins (V_{DD} and AV_{DD}) must be supplied with power, and all ground pins (V_{SS} and AV_{SS}) must be properly grounded. The DBG pin is open-drain and must always be connected to V_{DD} through an external pull-up resistor to ensure proper operation.

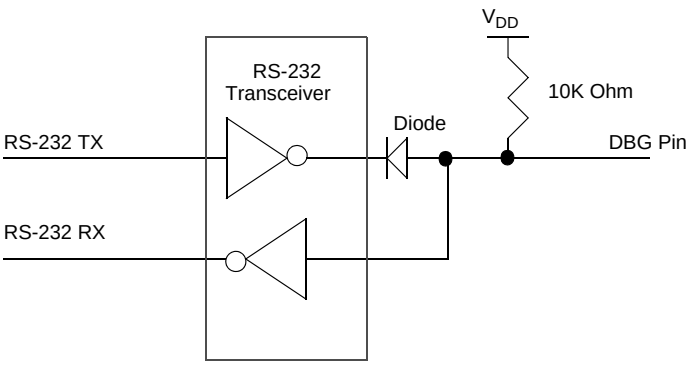


Figure 35. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #1 of 2

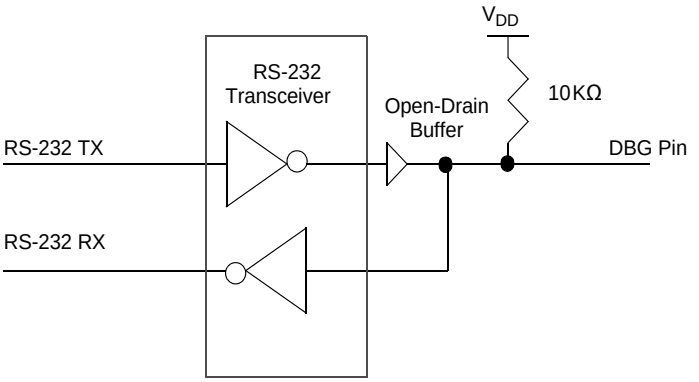


Figure 36. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #2 of 2

Debug Mode

The operating characteristics of the Z8 Encore! XP® F0822 Series devices in DEBUG Mode are:

- The eZ8 CPU fetch unit stops, idling the eZ8 CPU, unless directed by the OCD to execute specific instructions
- The system clock operates unless in STOP Mode
- All enabled on-chip peripherals operate unless in STOP Mode
- Automatically exits HALT Mode
- Constantly refreshes the Watchdog Timer, if enabled

Entering Debug Mode

The device enters DEBUG Mode following any of the following operations:

- Writing the DBGMODE bit in the OCD Control Register to 1 using the OCD interface
- eZ8 CPU execution of a breakpoint (BRK) instruction
- Matching of the PC to the OCDCNTR Register (when enabled)
- The OCDCNTR Register decrements to 0000H (when enabled)
- If the DBG pin is Low when the device exits Reset, the OCD automatically places the device into DEBUG Mode

Exiting Debug Mode

The device exits DEBUG Mode following any of the following operations:

- Clearing the DBGMODE bit in the OCD Control Register to 0
- Power-On Reset
- Voltage Brown-Out reset
- Asserting the RESET pin Low to initiate a Reset
- Driving the DBG pin Low while the device is in STOP Mode initiates a System Reset

OCD Data Format

The OCD interface uses the asynchronous data format defined for RS-232. Each character is transmitted as 1 start bit, 8 data bits (least-significant bit first), and 1 stop bit; see Figure 37.

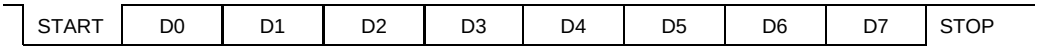


Figure 37. OCD Data Format

OCD Autobaud Detector/Generator

To run over a range of baud rates (bits per second) with various system clock frequencies, the OCD contains an Autobaud Detector/Generator. After a reset, the OCD is idle until it receives data. The OCD requires that the first character sent from the host is the character 80H. The character 80H has eight continuous bits Low (one start bit plus 7 data bits). The Autobaud Detector measures this period and sets the OCD Baud Rate Generator accordingly.

The Autobaud Detector/Generator is clocked by the system clock. The minimum baud rate is the system clock frequency divided by 512. For optimal operation, the maximum recommended baud rate is the system clock frequency divided by 8. The theoretical maximum baud rate is the system clock frequency divided by 4. This theoretical maximum is possible for low-noise designs with clean signals. Table 92 lists minimum and recommended maximum baud rates for sample crystal frequencies.

Table 92. OCD Baud-Rate Limits

System Clock Frequency (MHz)	Recommended Maximum Baud Rate (Kbps)	Minimum Baud Rate (Kbps)
20.0	2500	39.1
1.0	125.0	1.96
0.032768 (32kHz)	4.096	0.064

If the OCD receives a serial break (nine or more continuous bits Low) the Autobaud Detector/Generator resets. The Autobaud Detector/Generator can then be reconfigured by sending 80H.

OCD Serial Errors

The OCD can detect any of the following error conditions on the DBG pin:

- Serial break (a minimum of nine continuous bits Low)
- Framing error (received stop bit is Low)
- Transmit collision (OCD and host simultaneous transmission detected by the OCD)

When the OCD detects one of these errors, it aborts any command currently in progress, transmits a serial break that is 4096 system clock cycles in duration to the host, and resets the Autobaud Detector/Generator. A framing error or transmit collision can be caused by the host sending a serial break to the OCD. Because of the open-drain nature of the interface, returning a serial break back to the host only extends the length of the serial break if the host releases the serial break early.

The host transmits a serial break on the DBG pin when first connecting to the Z8 Encore! XP® F0822 Series device or when recovering from an error. A serial break from the host resets the Autobaud Generator/Detector but does not reset the OCD Control Register. A serial break leaves the device in DEBUG Mode if that is the current mode. The OCD is held in Reset until the end of the serial break when the DBG pin returns High. Because of the open-drain nature of the DBG pin, the host can send a serial break to the OCD even if the OCD is transmitting a character.

Breakpoints

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

If breakpoints are enabled, the OCD can be configured to automatically enter DEBUG Mode, or to loop on the break instruction. If the OCD is configured to loop on the BRK instruction, then the CPU is still enabled to service DMA and interrupt requests.

The loop on a BRK instruction can be used to service interrupts in the background. For interrupts to be serviced in the background, there cannot be any breakpoints in the ISR. Otherwise, the CPU stops on the breakpoint in the interrupt routine. For interrupts to be serviced in the background, interrupts must also be enabled. Debugging software should not automatically enable interrupts when using this feature, because interrupts are typically disabled during critical sections of code where interrupts should not occur (such as adjusting the stack pointer or modifying shared data).

Software can poll the IDLE bit of the OCDSTAT Register to determine if the OCD is looping on a BRK instruction. When software wants to stop the CPU on the BRK instruction it is looping on, software should not set the DBGMODE bit of the OCDCTL Register. The CPU can have vectored to and be in the middle of an ISR when this bit gets set. Instead, software must clear the BRKLP bit. This allows the CPU to finish the ISR it is in and return the BRK instruction. When the CPU returns to the BRK instruction it was previously looping on, it automatically sets the DBGMODE bit and enter DEBUG Mode.

Software should also note that the majority of the OCD commands are still disabled when the eZ8 CPU is looping on a BRK instruction. The eZ8 CPU must be stopped and the part must be in DEBUG Mode before these commands can be issued.

Breakpoints in Flash Memory

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

OCDCNTR Register

The OCD contains a multipurpose 16-bit counter register. It can be used for the following:

- Count system clock cycles between breakpoints
- Generate a BRK when it counts down to zero
- Generate a BRK when its value matches the Program Counter

When configured as a counter, the OCDCNTR Register 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`. The OCDCNTR Register automatically resets itself to `0000H` when the OCD exits DEBUG Mode if it is configured to count clock cycles between breakpoints.

Caution: The OCDCNTR Register is used by many of the OCD commands. It counts the number of bytes for the register and memory read/write commands. It holds the residual value when generating the CRC. Therefore, if the OCDCNTR is being used to generate a BRK, its value should be written as a last step before leaving DEBUG Mode.

Because this register is overwritten by various OCD commands, it should only be used to generate temporary breakpoints, such as stepping over CALL instructions or running to a specific instruction and stopping.

On-Chip Debugger Commands

The host communicates to the OCD 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 Read Protect option bit (RP). This Read Protect option bit prevents the code in memory from being read out of the Z8 Encore! XP® F0822 Series products. When this option is enabled, several of the OCD commands are disabled. Table 93 contains a summary of the OCD commands. Each OCD command is described further in the bulleted list. It also lists the commands that operate when the device is not in DEBUG Mode (normal operation) and those commands that are disabled by programming the Read Protect option bit.

Table 93. On-Chip Debugger Commands

Debug Command	Command Byte	Enabled When Not in DEBUG Mode?	Disabled by Read Protect Option Bit
Read OCD Revision	00H	Yes	–
Write OCD Counter Register	01H	–	–
Read OCD Status Register	02H	Yes	–
Read OCD Counter Register	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 Register	08H	–	Only writes of the peripheral control registers at address F00H–FFH are allowed. Additionally, only the Mass Erase command is allowed to be written to the Flash Control Register.

Table 93. On-Chip Debugger Commands (Continued)

Debug Command	Command Byte	Enabled When Not in DEBUG Mode?	Disabled by Read Protect Option Bit
Read Register	09H	–	Only reads of the peripheral control registers at address F00H–FFH are allowed.
Write Program Memory	0AH	–	Disabled
Read Program Memory	0BH	–	Disabled
Write Data Memory	0CH	–	Disabled
Read Data Memory	0DH	–	Disabled
Read Program Memory CRC	0EH	–	–
Reserved	0FH	–	–
Step Instruction	10H	–	Disabled
Stuff Instruction	11H	–	Disabled
Execute Instruction	12H	–	Disabled
Reserved	13H–FFH	–	–

In the following bulleted list of OCD Commands, data and commands sent from the host to the OCD are identified by DBG ← Command/Data. Data sent from the OCD back to the host is identified by DBG → Data.

Read OCD Revision (00H). The Read OCD Revision command determines the version of the OCD. If OCD commands are added, removed, or changed, this revision number changes.

```
DBG ← 00H
DBG → OCDREV[15:8] (Major revision number)
DBG → OCDREV[7:0] (Minor revision number)
```

Write OCD Counter Register (01H). The Write OCD Counter Register command writes the data that follows to the OCDCNTR Register. If the device is not in DEBUG Mode, the data is discarded.

```
DBG ← 01H
DBG ← OCDCNTR[15:8]
DBG ← OCDCNTR[7:0]
```

Read OCD Status Register (02H). The Read OCD Status Register command reads the OCDSTAT Register.

```
DBG ← 02H
DBG → OCDSTAT[7:0]
```

Read OCD Counter Register (03H). The OCD Counter Register can be used to count system clock cycles in between breakpoints, generate a BRK when it counts down to zero,

or generate a BRK when its value matches the Program Counter. Because this register is really a down counter, the returned value is inverted when this register is read so the returned result appears to be an up counter. If the device is not in DEBUG Mode, this command returns FFFFH.

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

Write OCD Control Register (04H). The Write OCD Control Register command writes the data that follows to the OCDCTL Register. When the 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 putting the device back into 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. If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the Program Counter 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. If the device is not in DEBUG Mode or if the 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 zero). If the device is not in DEBUG Mode, the address and data values are discarded. If the Read Protect option bit is enabled, then 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
```

Read Register (09H). The Read Register command reads data from the Register File. Data can be read 1-256 bytes at a time (256 bytes can be read by setting size to zero). Reading peripheral control registers through the OCD does not effect peripheral operation. For example, register bits that are normally cleared upon a read operation will not be affected (the WDTSTAT Register is affected by the OCD Read Register operation). If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, this command returns FFH for all of the data values.

```
DBG ← 09H
DBG ← {4'h0,Register Address[11:8]
DBG ← Register Address[7:0]
DBG ← Size[7:0]
DBG ⟶ 1-256 data bytes
```

Write Program Memory (0AH). The Write Program Memory command writes data to Program memory. This command is equivalent to the LDC and LDCI instructions. Data can be written 1-65536 bytes at a time (65536 bytes can be written by setting size to zero). The on-chip Flash Controller must be written to and unlocked for the programming operation to occur. If the Flash Controller is not unlocked, the data is discarded. If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the data is discarded.

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

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

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

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

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

```
DBG ← Size[7:0]
DBG ← 1-65536 data bytes
```

Read Data Memory (0DH). The Read Data Memory command reads from Data Memory. This command is equivalent to the LDE and LDEI instructions. Data can be read 1-65536 bytes at a time (65536 bytes can be read by setting size to 0). If the device is not in DEBUG Mode, this command returns FFH for the data.

```
DBG ← 0DH
DBG ← Data Memory Address[15:8]
DBG ← Data Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG → 1-65536 data bytes
```

Read Program Memory CRC (0EH). The Read Program Memory CRC command computes and returns the CRC (cyclic redundancy check) of Program memory using the 16-bit CRC-CCITT polynomial. If the device is not in DEBUG Mode, this command returns FFFFH for the CRC value. Unlike most other OCD Read commands, there is a delay from issuing of the command until the OCD returns the data. The OCD reads the Program memory, calculates the CRC value, and returns the result. The delay is a function of the Program memory size and is approximately equal to the system clock period multiplied by the number of bytes in the Program memory.

```
DBG ← 0EH
DBG → CRC[15:8]
DBG → CRC[7:0]
```

Step Instruction (10H). The Step Instruction command steps one assembly instruction at the current Program Counter location. If the device is not in DEBUG Mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 10H
```

Stuff Instruction (11H). The Stuff Instruction command steps one assembly instruction and allows specification of the first byte of the instruction. The remaining 0-4 bytes of the instruction are read from Program memory. This command is useful for stepping over instructions where the first byte of the instruction has been overwritten by a Breakpoint. If the device is not in DEBUG Mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 11H
DBG ← opcode[7:0]
```

Execute Instruction (12H). The Execute Instruction command allows sending an entire instruction to be executed to the eZ8 CPU. This command can also step over breakpoints. The number of bytes to send for the instruction depends on the Op Code. If the device is not in DEBUG Mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 12H
DBG ← 1-5 byte opcode
```

On-Chip Debugger Control Register Definitions

This section describes the features of the On-Chip Debugger Control and Status registers.

OCD Control Register

The OCD Control Register, shown in Table 94, controls the state of the OCD. This register enters or exits DEBUG Mode and enables the BRK instruction. It can also reset the Z8 Encore! XP® F0822 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 94. OCD Control Register (OCDCTL)

Bit	7	6	5	4	3	2	1	0
Field	DBGMODE	BRKEN	DBGACK	BRKLOOP	BRKPC	BRKZRO	Reserved	RST
RESET	0							
R/W	R/W			R				R/W

Bit	Description
[7] DBGMODE	**Debug Mode** Setting this bit to 1 causes the device to enter DEBUG Mode. When in DEBUG Mode, the eZ8 CPU stops fetching new instructions. Clearing this bit causes the eZ8 CPU to start running again. This bit is automatically set when a BRK instruction is decoded and breakpoints are enabled. If the 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® F0822 Series device is operating in NORMAL Mode. 1 = The Z8 Encore! XP® F0822 Series device is in DEBUG Mode.
[6] BRKEN	**Breakpoint Enable** This bit controls the behavior of the BRK instruction (Op Code 00H). By default, breakpoints are disabled and the BRK instruction behaves like an NOP instruction. If this bit is set to 1 and a BRK instruction is decoded, the OCD takes action dependent upon the BRKLOOP bit. 0 = BRK instruction is disabled. 1 = BRK instruction is enabled.
[5] DBGACK	**Debug Acknowledge** This bit enables the debug acknowledge feature. If this bit is set to 1, then the OCD sends an Debug Acknowledge character (FFH) to the host when a Breakpoint occurs. 0 = Debug Acknowledge is disabled. 1 = Debug Acknowledge is enabled.

Bit	Description (Continued)
[4] BRKLOOP	**Breakpoint Loop** This bit determines what action the OCD takes when a BRK instruction is decoded if breakpoints are enabled (BRKEN is 1). If this bit is 0, then the DBGMODE bit is automatically set to 1 and the OCD enter DEBUG Mode. If BRKLOOP is set to 1, then the eZ8 CPU loops on the BRK instruction. 0 = BRK instruction sets DBGMODE to 1. 1 = eZ8 CPU loops on BRK instruction.
[3] BRKPC	**Break When PC == OCDCNTR** If this bit is set to 1, then the OCDCNTR Register is used as a hardware breakpoint. When the program counter matches the value in the OCDCNTR Register, DBGMODE is automatically set to 1. If this bit is set, the OCDCNTR Register does not count when the CPU is running. 0 = OCDCNTR is setup as counter. 1 = OCDCNTR generates hardware break when PC == OCDCNTR.
[2] BRKZRO	**Break When OCDCNTR == 0000H** If this bit is set, then the OCD automatically sets the DBGMODE bit when the OCDCNTR Register counts down to 0000H. If this bit is set, the OCDCNTR Register is not reset when the part leaves DEBUG Mode. 0 = OCD does not generate BRK when OCDCNTR decrements to 0000H. 1 = OCD sets DBGMODE to 1 when OCDCNTR decrements to 0000H.
[1]	**Reserved** This bit is reserved and must be programmed to 0.
[0] RST	**Reset** Setting this bit to 1 resets the Z8 Encore! XP® F0822 Series device. The device goes through a normal POR sequence with the exception that the OCD is not reset. This bit is automatically cleared to 0 when the reset finishes. 0 = No effect. 1 = Reset the Z8 Encore! XP® F0822 Series device.

OCD Status Register

The OCD Status Register, shown in Table 95, reports status information about the current state of the debugger and the system.

Table 95. OCD Status Register (OCDSTAT)

Bit	7	6	5	4	3	2	1	0
Field	IDLE	HALT	RPEN	Reserved				
RESET	0							
R/W	R							

Bit	Description
[7] IDLE	**CPU Idling** This bit is set if the part is in DEBUG Mode (DBGMODE is 1), or if a BRK instruction occurred since the last time OCDCTL was written. This can be used to determine if the CPU is running or if it is idling. 0 = The eZ8 CPU is running. 1 = The eZ8 CPU is either stopped or looping on a BRK instruction.
[6] HALT	**HALT Mode** 0 = The device is not in HALT Mode. 1 = The device is in HALT Mode.
[5] RPEN	**Read Protect Option Bit Enabled** 0 = The Read Protect option bit is disabled (1). 1 = The Read Protect option bit is enabled (0), disabling many OCD commands.
[4:0]	**Reserved** These bits are reserved and must be programmed to 00000.

On-Chip Oscillator

Z8 Encore! XP® F0822 Series products feature an on-chip oscillator for use with external crystals with frequencies from 32kHz to 20MHz. In addition, the oscillator can support external RC networks with oscillation frequencies up to 4MHz or ceramic resonators with oscillation frequencies up to 20MHz. This oscillator generates the primary system clock for the internal eZ8 CPU and the majority of the on-chip peripherals. Alternatively, the X_{IN} input pin can also accept a CMOS-level clock input signal (32kHz–20MHz). If an external clock generator is used, the X_{OUT} pin must remain unconnected.

When configured for use with crystal oscillators or external clock drivers, the frequency of the signal on the X_{IN} input pin determines the frequency of the system clock (that is, no internal clock divider). In RC operation, the system clock is driven by a clock divider (divide by 2) to ensure 50% duty cycle.

Operating Modes

Z8 Encore! XP® F0822 Series products support 4 different oscillator modes:

- On-chip oscillator configured for use with external RC networks (<4MHz)
- Minimum power for use with very-low-frequency crystals (32kHz to 1.0MHz)
- Medium power for use with medium frequency crystals or ceramic resonators (0.5MHz to 10.0MHz)
- Maximum power for use with high-frequency crystals or ceramic resonators (8.0MHz to 20.0MHz)

The oscillator mode is selected through user-programmable option bits. For more information, see the Option Bits chapter on page 155.

Crystal Oscillator Operation

Figure 38 displays a recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 20MHz. Recommended 20MHz crystal specifications are provided in Table 96. Resistor R1 is optional and limits total power dissipation by the crystal. The printed circuit board layout must add no more than 4pF of stray capacitance to either the X_{IN} or X_{OUT} pins. If oscillation does not occur, reduce the values of capacitors C1 and C2 to decrease loading.

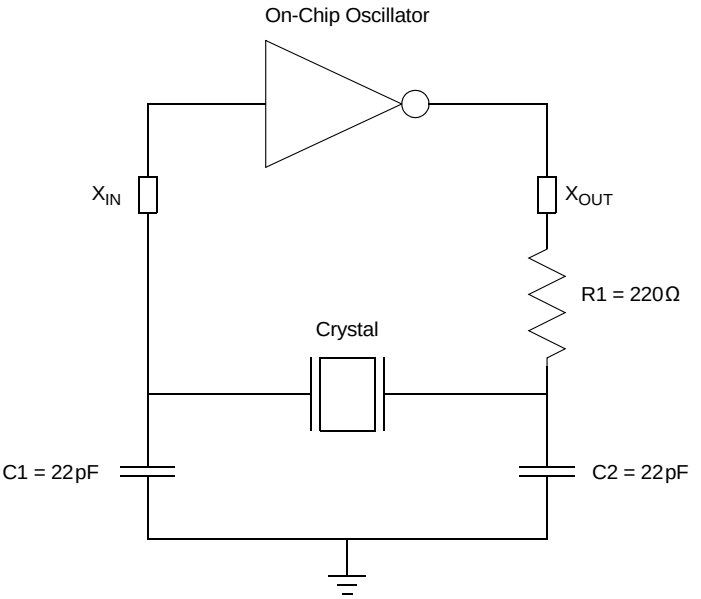


Figure 38. Recommended 20MHz Crystal Oscillator Configuration

Table 96. Recommended Crystal Oscillator Specifications (20MHz Operation)

Parameter	Value	Units	Comments
Frequency	20	MHz	
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	25	Ω	Maximum
Load Capacitance (C_L)	20	pF	Maximum
Shunt Capacitance (C_0)	7	pF	Maximum
Drive Level	1	mW	Maximum

Oscillator Operation with an External RC Network

The External RC oscillator mode is applicable to timing insensitive applications. Figure 39 displays a recommended configuration for connection with an external resistor-capacitor (RC) network.

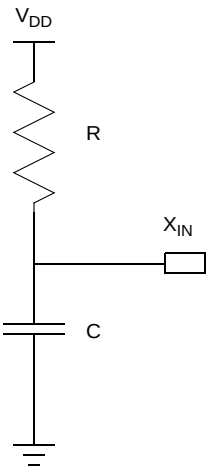


Figure 39. Connecting the On-Chip Oscillator to an External RC Network

An external resistance value of 45 kΩ 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 below equation:

$$\text{Oscillator Frequency (kHz)} = \frac{1\times10^{6}}{(0.4 \times R \times C) + (4 \times C)}$$

Figure 40 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 45 kΩ external resistor. For very small values of C, the parasitic capacitance of the oscillator X_{IN} pin and the printed circuit board should 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 parasites, external capacitance values in excess of 20pF are recommended.

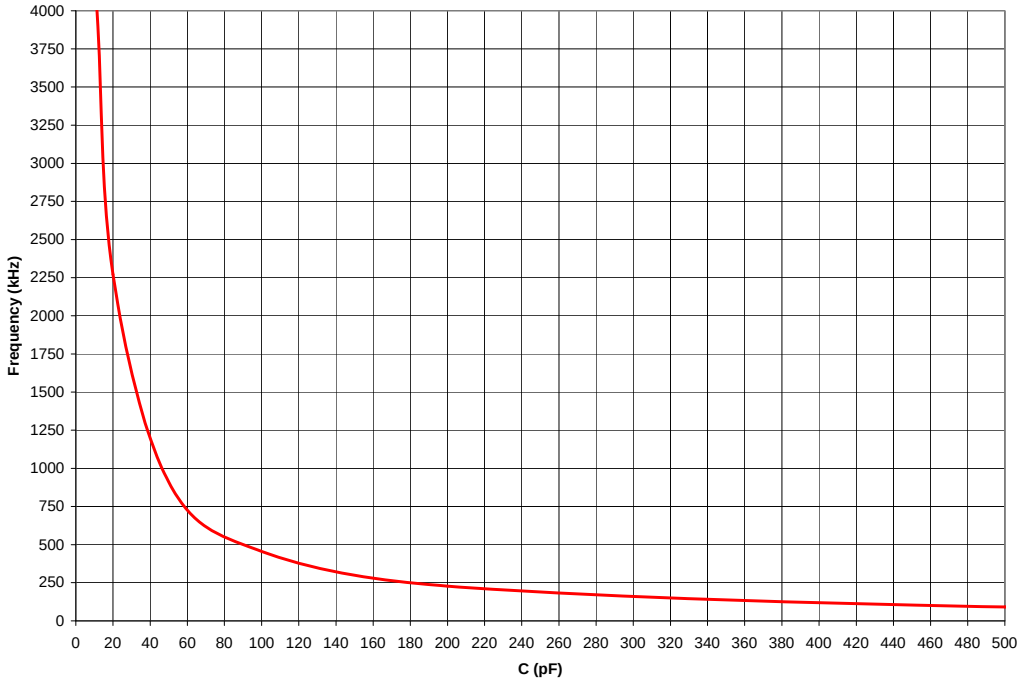


Figure 40. Typical RC Oscillator Frequency as a Function of External Capacitance with a 45kΩ Resistor

Caution: When using the external RC oscillator mode, the oscillator can stop oscillating if the power supply drops below 2.7V, but before the power supply drops to the Voltage Brown-Out threshold. The oscillator resumes oscillation when the supply voltage exceeds 2.7V.

Electrical Characteristics

The data in this chapter represents all known data prior to qualification and characterization of the Z8 Encore! XP® F0822 Series of products, and is therefore subject to change. Additional electrical characteristics may be found in the individual chapters of this document.

Absolute Maximum Ratings

These ratings are stress ratings only. Operation of the device at any condition outside those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods can affect device reliability. For improved reliability, unused inputs must be tied to one of the supply voltages (V_{DD} or V_{SS}).

Caution: Stresses greater than those listed in Table 97 can cause permanent damage to the device.

Table 97. Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units	Notes
Ambient temperature under bias	–40	+105	°C	1
Storage temperature	–65	+150	°C	
Voltage on any pin with respect to V_{SS}	–0.3	+5.5	V	2
Voltage on AV_{SS} pin with respect to V_{SS}	–0.3	+0.3	V	2
Voltage on V_{DD} pin with respect to V_{SS}	–0.3	+3.6	V	
Voltage on AV_{DD} pin with respect to V_{DD}	–0.3	+0.3	V	
Maximum current on input and/or inactive output pin	–5	+5	µA	
Maximum output current from active output pin	–25	+25	mA	
20-pin SSOP Package Maximum Ratings at –40°C to 70°C				
Total power dissipation		430	mW	
Maximum current into V_{DD} or out of V_{SS}		120	mA	

Note: This voltage applies to all pins except the following: V_{DD}, AV_{DD}, V_{REF}, pins that support analog input (Port B), and where otherwise noted.

Table 97. Absolute Maximum Ratings (Continued)

Parameter	Minimum	Maximum	Units	Notes
20-pin SSOP Package Maximum Ratings at 70°C to 105°C				
Total power dissipation		250	mW	
Maximum current into V_{DD} or out of V_{SS}		69	mA	
20-pin PDIP Package Maximum Ratings at –40°C to 70°C				
Total power dissipation		775	mW	
Maximum current into V_{DD} or out of V_{SS}		215	mA	
20-pin PDIP Package Maximum Ratings at 70°C to 105°C				
Total power dissipation		285	mW	
Maximum current into V_{DD} or out of V_{SS}		79	mA	
28-pin SOIC Package Maximum Ratings at –40°C to 70°C				
Total power dissipation		450	mW	
Maximum current into V_{DD} or out of V_{SS}		125	mA	
28-pin SOIC Package Maximum Ratings at 70°C to 105°C				
Total power dissipation		260	mW	
Maximum current into V_{DD} or out of V_{SS}		73	mA	
28-pin PDIP Package Maximum Ratings at –40°C to 70°C				
Total power dissipation		1100	mW	
Maximum current into V_{DD} or out of V_{SS}		305	mA	
28-pin PDIP Package Maximum Ratings at 70°C to 105°C				
Total power dissipation		400	mW	
Maximum current into V_{DD} or out of V_{SS}		110	mA	

Note: This voltage applies to all pins except the following: V_{DD}, AV_{DD}, V_{REF}, pins that support analog input (Port B), and where otherwise noted.

DC Characteristics

Table 98 lists the DC characteristics of the Z8 Encore! XP® F0822 Series products. All voltages are referenced to V_{SS}, the primary system ground.

Table 98. DC Characteristics

		T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
V_{DD}	Supply Voltage	2.7	–	3.6	V	
V_{IL1}	Low Level Input Voltage	-0.3	–	0.3*V_{DD}	V	For all input pins except $\overline{\text{RESET}}$, DBG, and X_{IN}.
V_{IL2}	Low Level Input Voltage	–0.3	–	0.2*V_{DD}	V	For $\overline{\text{RESET}}$, DBG, and X_{IN}.
V_{IH1}	High Level Input Voltage	0.7*V_{DD}	–	5.5	V	Ports A and C pins when their programmable pull-ups are disabled.
V_{IH2}	High Level Input Voltage	0.7*V_{DD}	–	V_{DD}+0.3	V	Port B pins. Ports A and C pins when their programmable pull-ups are enabled.
V_{IH3}	High Level Input Voltage	0.8*V_{DD}	–	V_{DD}+0.3	V	$\overline{\text{RESET}}$, DBG, and X_{IN} pins.
V_{OL1}	Low Level Output Voltage	–	–	0.4	V	I_{OL} = 2 mA; V_{DD} = 3.0V High Output Drive disabled.
V_{OH1}	High Level Output Voltage	2.4	–	–	V	I_{OH} = –2 mA; V_{DD} = 3.0V High Output Drive disabled.
V_{OL2}	Low Level Output Voltage High Drive	–	–	0.6	V	I_{OL} = 20 mA; V_{DD} = 3.3V High Output Drive enabled T_A = –40°C to +70°C
V_{OH2}	High Level Output Voltage High Drive	2.4	–	–	V	I_{OH} = –20 mA; V_{DD} = 3.3V High Output Drive enabled; T_A = –40°C to +70°C
V_{OL3}	Low Level Output Voltage High Drive	–	–	0.6	V	I_{OL} = 15mA; V_{DD} = 3.3V High Output Drive enabled; T_A = +70°C to +105°C
V_{OH3}	High Level Output Voltage High Drive	2.4	–	–	V	I_{OH} = 15 mA; V_{DD} = 3.3V High Output Drive enabled; T_A = +70°C to +105°C
V_{RAM}	RAM Data Retention	0.7	–	–	V	
I_{IL}	Input Leakage Current	–5	–	+5	µA	V_{DD} = 3.6V; V_{IN} = V_{DD} or V_{SS}[1]

Table 98. DC Characteristics (Continued)

		T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
I_{TL}	Tri-State Leakage Current	–5	–	+5	µA	V_{DD} = 3.6 V
C_{PAD}	GPIO Port Pad Capacitance	–	8.0[2]	–	pF	
C_{XIN}	X_{IN} Pad Capacitance	–	8.0[2]	–	pF	
C_{XOUT}	X_{OUT} Pad Capacitance	–	9.5[2]	–	pF	
I_{PU1}	Weak Pull-up Current	9	20	50	µA	V_{DD} = 2.7–3.6V T_A = 0°C to +70°C
I_{PU2}	Weak Pull-up Current	7	20	75	µA	V_{DD} = 2.7–3.6V T_A = –40°C to +105°C

Note: [1] This condition excludes all pins that have on-chip pull-ups, when driven Low.
Note: [2] These values are provided for design guidance only and are not tested in production.

Figure 41 displays the typical active mode current consumption while operating at 25ºC, 3.3V, plotted opposite the system clock frequency. All GPIO pins are configured as outputs and driven High.

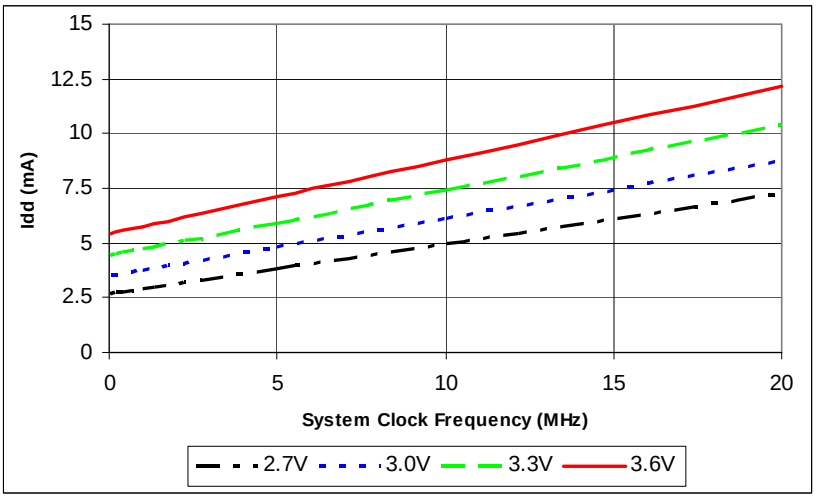


Figure 41. Typical Active Mode I_{DD} vs. System Clock Frequency

Figure 42 displays the maximum active mode current consumption across the full operating temperature range of the device and plotted opposite the system clock frequency. All GPIO pins are configured as outputs and driven High.

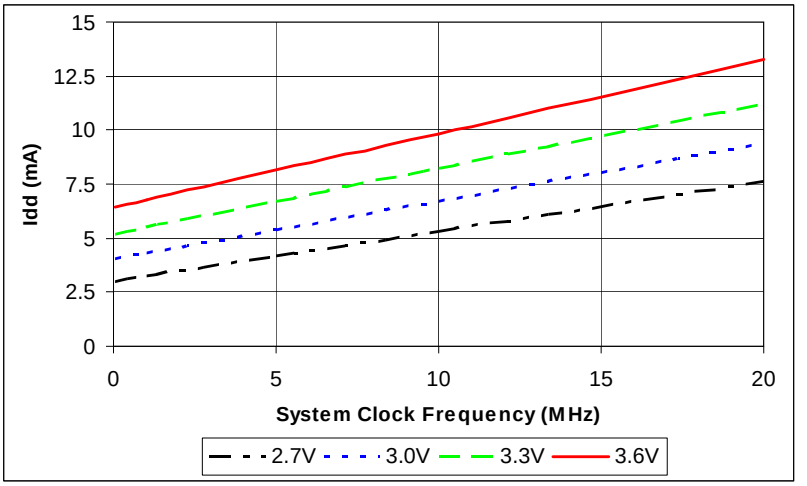


Figure 42. Maximum Active Mode I_{DD} vs. System Clock Frequency

Figure 43 displays the typical current consumption in HALT Mode while operating at 25ºC plotted opposite the system clock frequency. All GPIO pins are configured as outputs and driven High.

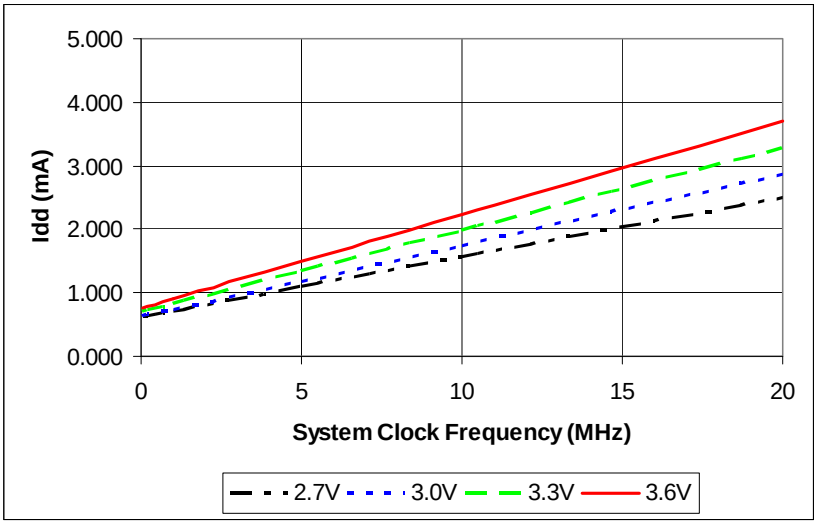


Figure 43. Typical HALT Mode I_{DD} vs. System Clock Frequency

Figure 44 displays the maximum HALT Mode current consumption across the entire operating temperature range of the device and plotted opposite the system clock frequency. All GPIO pins are configured as outputs and driven High.

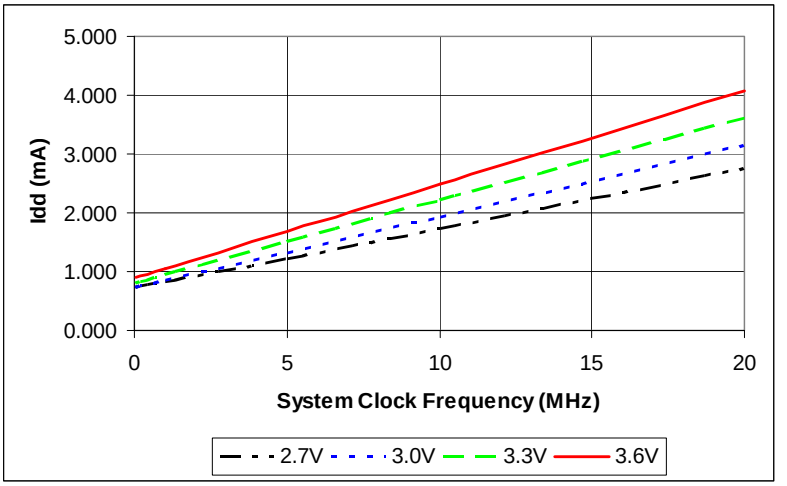


Figure 44. Maximum HALT Mode I_{CC} vs. System Clock Frequency

Figure 45 displays the maximum current consumption in STOP Mode with the VBO and Watchdog Timer enabled plotted opposite the power supply voltage. All GPIO pins are configured as outputs and driven High.

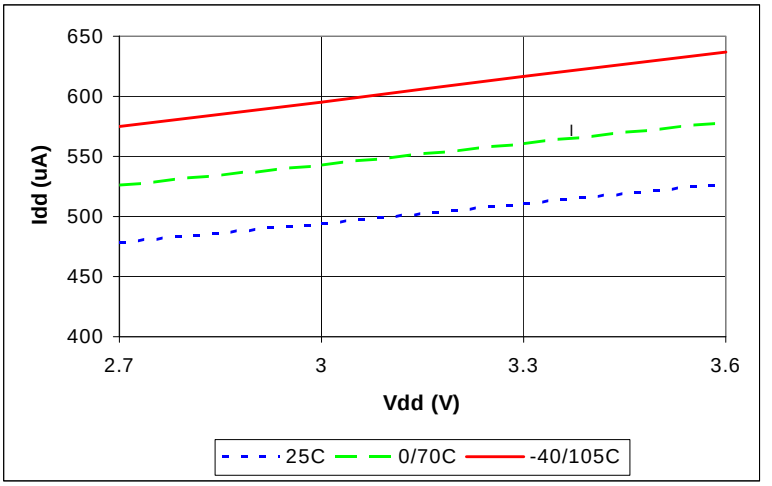


Figure 45. Maximum STOP Mode I_{DD} with VBO Enabled vs. Power Supply Voltage

Figure 46 displays the maximum current consumption in STOP Mode with the VBO disabled and Watchdog Timer enabled plotted opposite the power supply voltage. All GPIO pins are configured as outputs and driven High. Disabling the Watchdog Timer and its internal RC oscillator in STOP Mode will provide some additional reduction in STOP Mode current consumption. This small current reduction is indistinquishable on the scale of Figure 46.

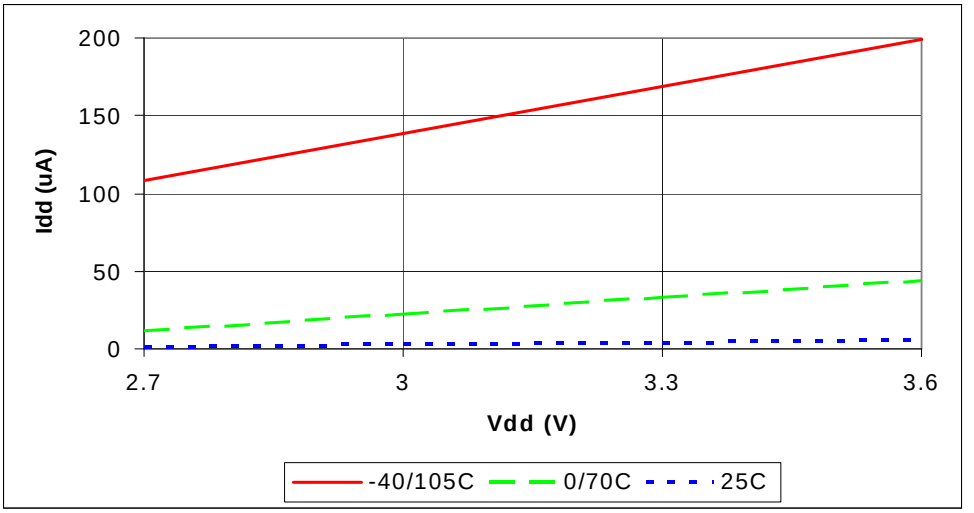


Figure 46. Maximum STOP Mode I_{DD} with VBO Disabled vs. Power Supply Voltage

AC Characteristics

Table 99 provides information about the AC characteristics and timing. All AC timing information assumes a standard load of 50pF on all outputs.

Table 99. AC Characteristics

		V_{DD} = 2.7–3.6V T_A = –40°C to 105°C			
Symbol	**Parameter**	**Minimum**	**Maximum**	**Units**	**Conditions**
F_{SYSCLK}	System Clock Frequency (ROM)	–	20.0	MHz	
F_{SYSCLK}	System Clock Frequency (Flash)	–	20.0	MHz	Read-only from Flash memory.
		0.032768	20.0	MHz	Program or erasure of Flash memory.
F_{XTAL}	Crystal Oscillator Frequency	0.032768	20.0	MHz	System clock frequencies below the crystal oscillator minimum require an external clock driver.
T_{XIN}	System Clock Period	50	–	ns	$T_{CLK} = 1/F_{SYSCLK}$
T_{XINH}	System Clock High Time	20	30	ns	T_{CLK} = 50ns
T_{XINL}	System Clock Low Time	20	30	ns	T_{CLK} = 50ns

On-Chip Peripheral AC and DC Electrical Characteristics

Table 100 provides information about the Power-On Reset and Voltage Brown-Out electrical characteristics.

Table 100. Power-On Reset and Voltage Brown-Out Electrical Characteristics and Timing

		T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical***	**Maximum**	**Units**	**Conditions**
V_{POR}	Power-On Reset Voltage Threshold	2.15	2.40	2.60	V	$V_{DD} = V_{POR}$
V_{VBO}	Voltage Brown-Out Reset Voltage Threshold	2.05	2.30	2.55	V	$V_{DD} = V_{VBO}$
	V_{POR} to V_{VBO} hysteresis	50	100	–	mV	
	Starting V_{DD} voltage to ensure valid POR	–	V_{SS}	–	V	
T_{ANA}	POR Analog Delay	–	50	–	µs	$V_{DD} > V_{POR}$; T_{POR} Digital Reset delay follows T_{ANA}
T_{POR}	POR Digital Delay	–	5.0	–	ms	50 WDT Oscillator cycles (10kHz) + 16 System Clock cycles (20MHz)
T_{VBO}	Voltage Brown-Out Pulse Rejection Period	–	10	–	µs	$V_{DD} < V_{VBO}$ to generate a Reset.
T_{RAMP}	Time for VDD to transition from V_{SS} to V_{POR} to ensure valid Reset	0.10	–	100	ms	

Note: *Data in the typical column is from characterization at 3.3 V and 25°C. These values are provided for design guidance only and are not tested in production.

Table 101 provides information about the external RC oscillator electrical characteristics and timing, and Table 102 provides information about the Flash memory electrical characteristics and timing.

Table 101. External RC Oscillator Electrical Characteristics and Timing

		T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical***	**Maximum**	**Units**	**Conditions**
V_{DD}	Operating Voltage Range	2.70	–	–	V	
R_{EXT}	External Resistance from X_{IN} to V_{DD}	40	45	200	kΩ	
C_{EXT}	External Capacitance from X_{IN} to V_{SS}	0	20	1000	pF	
F_{OSC}	External RC Oscillation Frequency	–	–	4	MHz	

Note: *When using the external RC oscillator mode, the oscillator can stop oscillating if the power supply drops below 2.7V, but before the power supply drops to the voltage brown-out threshold. The oscillator will resume oscillation as soon as the supply voltage exceeds 2.7V.

Table 102. Flash Memory Electrical Characteristics and Timing

	V_{DD} = 2.7–3.6V T_A = –40°C to 105°C				
Parameter	**Minimum**	**Typical**	**Maximum**	**Units**	**Notes**
Flash Byte Read Time	50	–	–	µs	
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 103 lists Reset and Stop Mode Recovery pin timing data; Table 104 lists Watchdog Timer Electrical Characteristics and Timing data.

Table 103. Reset and Stop Mode Recovery Pin Timing

		T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical***	**Maximum**	**Units**	**Conditions**
T_{RESET}	$\overline{\text{Reset}}$ pin assertion to initiate a System Reset	4	–	–	T_{CLK}	Not in STOP Mode. T_{CLK} = System Clock period.
T_{SMR}	Stop Mode Recovery pin Pulse Rejection Period	10	20	40	ns	$\overline{\text{RESET}}$, DBG and GPIO pins configured as SMR sources.

Note: *When using the external RC oscillator mode, the oscillator can stop oscillating if the power supply drops below 2.7V, but before the power supply drops to the voltage brown-out threshold. The oscillator will resume oscillation as soon as the supply voltage exceeds 2.7V.

Table 104. Watchdog Timer Electrical Characteristics and Timing

		V_{DD} = 2.7–3.6 V T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
F_{WDT}	WDT Oscillator Frequency	5	10	20	kHz	
I_{WDT}	WDT Oscillator Current including internal RC oscillator	–	< 1	5	µA	

Figure 47 displays the input frequency response of the ADC.

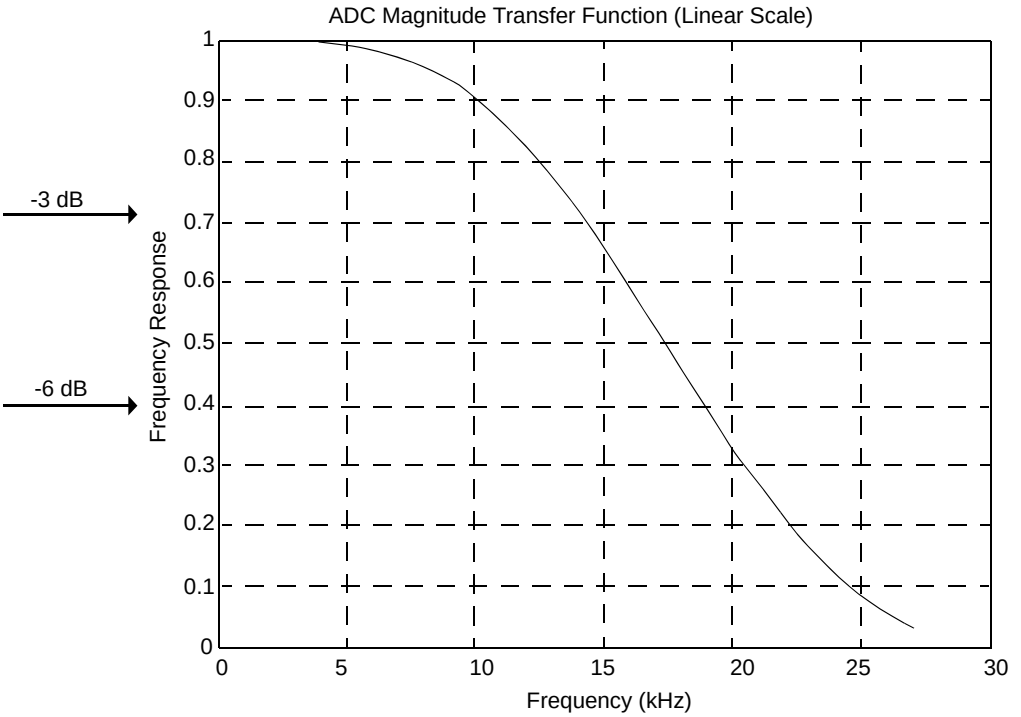


Figure 47. Analog-to-Digital Converter Frequency Response

Table 105 lists ADC electrical characteristics and timing data.

Table 105. Analog-to-Digital Converter Electrical Characteristics and Timing

		V_{DD} = 3.0–3.6 V T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
	Resolution	10	–	–	bits	External V_{REF} = 3.0V
	Differential Nonlinearity (DNL)	–0.25	–	0.25	lsb	Guaranteed by design
	Integral Nonlinearity (INL)	–2.0	–	2.0	lsb	External V_{REF} = 3.0V
	DC Offset Error	–35	–	25	mV	80-pin QFP and 64-pin LQFP packages.
V_{REF}	Internal Reference Voltage	1.9	2.0	2.4	V	V_{DD} = 3.0–3.6V T_A = –40°C to 105°C
VC_{REF}	Voltage Coefficient of Internal Reference Voltage	–	78	–	mV/V	V_{REF} variation as a function of AV_{DD}.
TC_{REF}	Temperature Coefficient of Internal Reference Voltage	–	1	–	mV/°C	
	Single-Shot Conversion Period		5129		cycles	System clock cycles
	Continuous Conversion Period		256		cycles	System clock cycles
R_S	Analog Source Impedance	–	–	150	Ω	Recommended
Zin	Input Impedance		150		KΩ	20MHz system clock. Input impedance increases with lower system clock frequency.
V_{REF}	External Reference Voltage			AV_{DD}	V	AV_{DD} <= V_{DD}. When using an external reference voltage, decoupling capacitance should be placed from V_{REF} to AV_{SS}.
I_{REF}	Current draw into V_{REF} pin when driving with external source.		25.0	40.0	µA	

General Purpose I/O Port Input Data Sample Timing

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

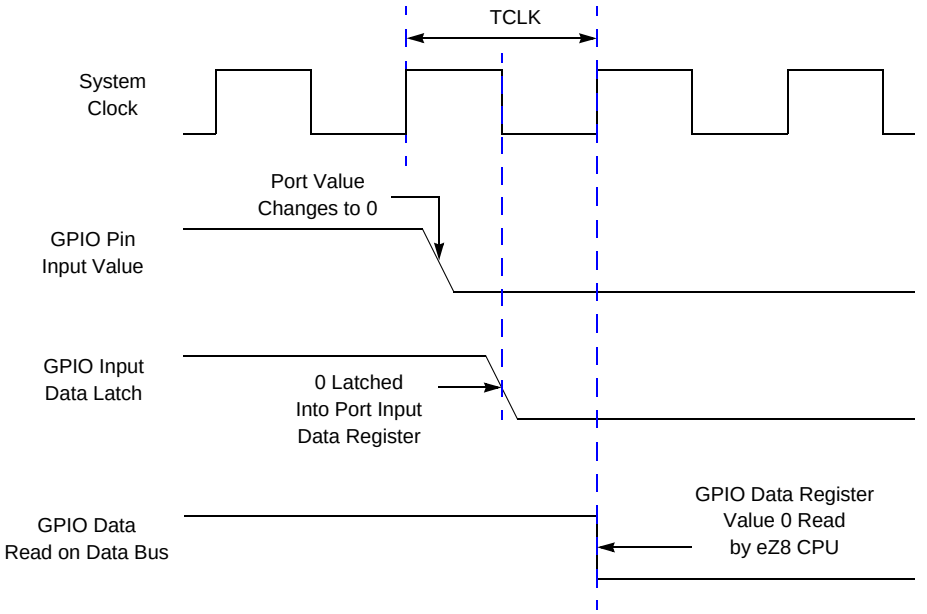


Figure 48. Port Input Sample Timing

Table 106. GPIO Port Input Timing

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

General Purpose I/O Port Output Timing

Figure 49 and Table 107 provide timing information for GPIO port pins.

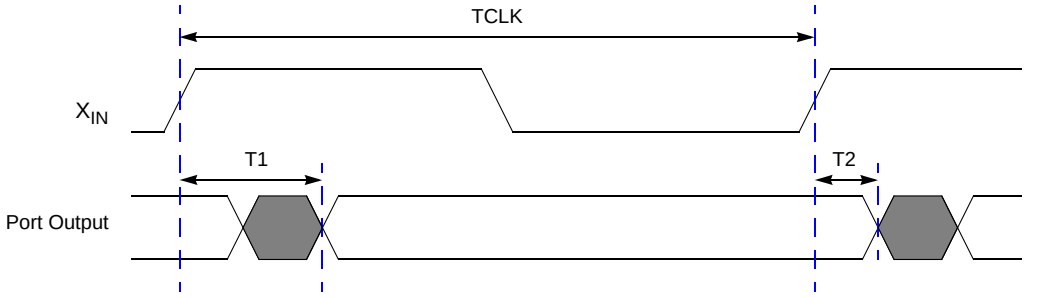


Figure 49. GPIO Port Output Timing

Table 107. GPIO Port Output Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
GPIO Port Pins			
T_1	X_{IN} Rise to Port Output Valid Delay	–	15
T_2	X_{IN} Rise to Port Output Hold Time	2	–

On-Chip Debugger Timing

Figure 50 and Table 108 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4µs maximum rise and fall time.

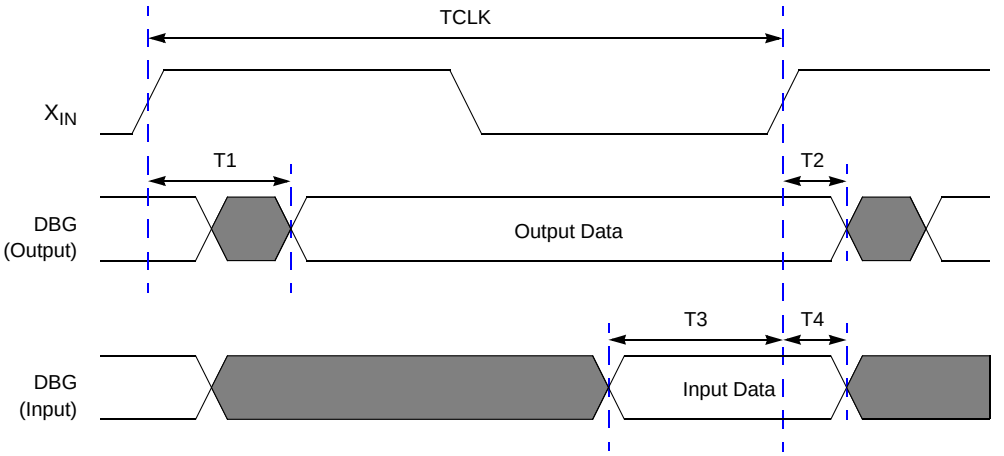


Figure 50. On-Chip Debugger Timing

Table 108. On-Chip Debugger Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
DBG			
T_1	X_{IN} Rise to DBG Valid Delay	–	15
T_2	X_{IN} Rise to DBG Output Hold Time	2	–
T_3	DBG to X_{IN} Rise Input Setup Time	10	–
T_4	DBG to X_{IN} Rise Input Hold Time	5	–
	DBG frequency		System Clock/4

SPI MASTER Mode Timing

Figure 51 and Table 109 provide timing information for SPI MASTER Mode pins. Timing is shown with SCK rising edge used to source MOSI output data, SCK falling edge used to sample MISO input data. Timing on the SS output pin(s) is controlled by software.

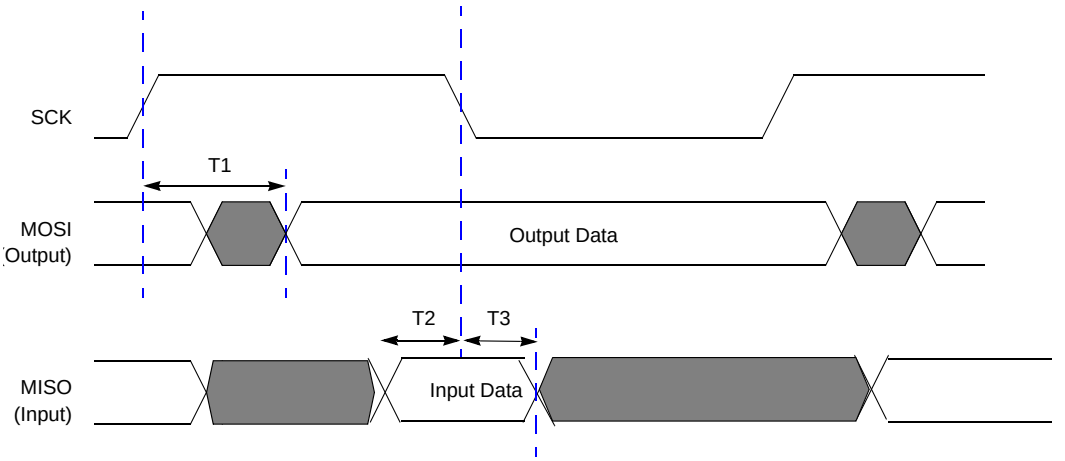


Figure 51. SPI MASTER Mode Timing

Table 109. SPI MASTER Mode Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
SPI MASTER			
T_1	SCK Rise to MOSI output Valid Delay	–5	+5
T_2	MISO input to SCK (receive edge) Setup Time	20	
T_3	MISO input to SCK (receive edge) Hold Time	0	

SPI SLAVE Mode Timing

Figure 52 and Table 110 provide timing information for the SPI SLAVE Mode pins. Timing is shown with SCK rising edge used to source MISO output data, SCK falling edge used to sample MOSI input data.

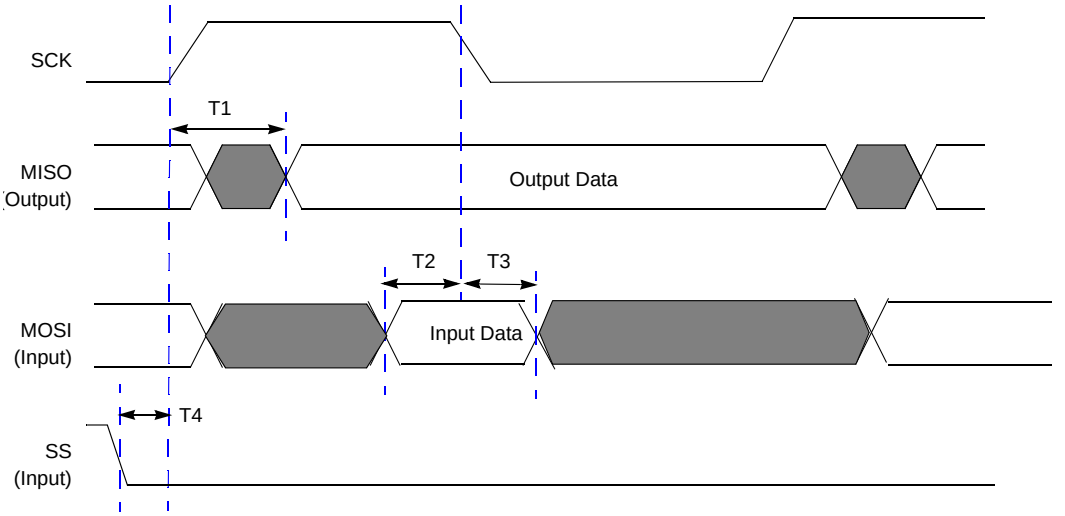


Figure 52. SPI SLAVE Mode Timing

Table 110. SPI SLAVE Mode Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
SPI SLAVE			
T_1	SCK (transmit edge) to MISO output Valid Delay	2 * X_{IN} period	3 * X_{IN} period + 20ns
T_2	MOSI input to SCK (receive edge) Setup Time	0	
T_3	MOSI input to SCK (receive edge) Hold Time	3 * X_{IN} period	
T_4	SS input assertion to SCK setup	1 * X_{IN} period	

I²C Timing

Figure 53 and Table 111 provide timing information for I²C pins.

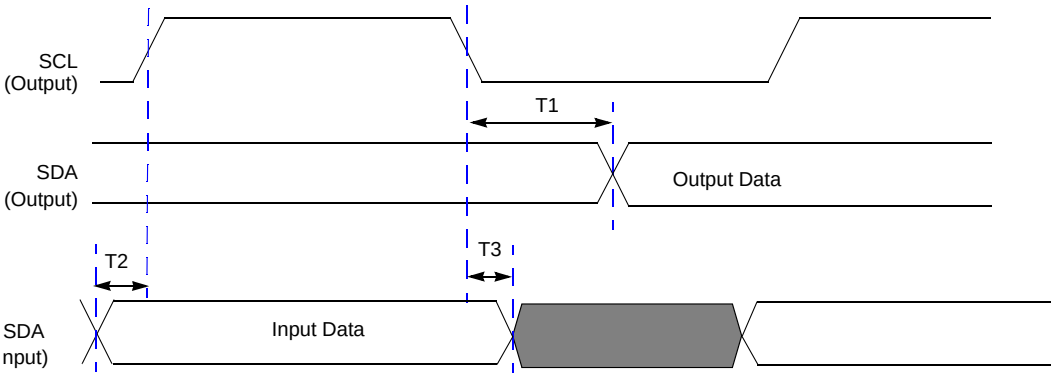


Figure 53. I²C Timing

Table 111. I²C Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
I²C			
T_1	SCL Fall to SDA output delay	SCL period/4	
T_2	SDA Input to SCL rising edge Setup Time	0	
T_3	SDA Input to SCL falling edge Hold Time	0	

UART Timing

Figure 54 and Table 112 provide timing information for UART pins for the case where the Clear To Send input pin ($\overline{CTS}$) is used for flow control. In this example, it is assumed that the Driver Enable polarity has been configured to be Active Low and is represented here by $\overline{DE}$. The $\overline{CTS}$ to $\overline{DE}$ assertion delay (T1) assumes the UART Transmit Data Register has been loaded with data prior to $\overline{CTS}$ assertion.

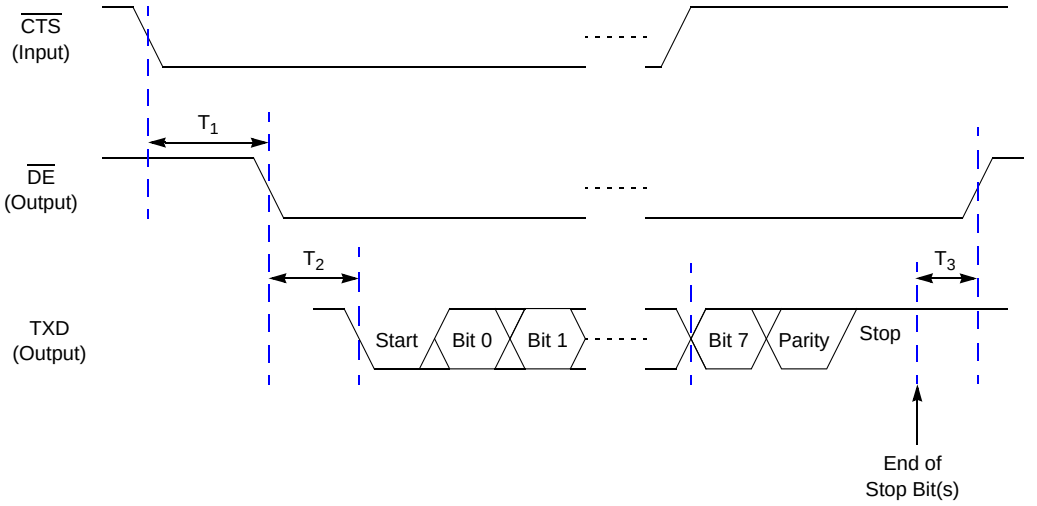


Figure 54. UART Timing with $\overline{CTS}$

Table 112. UART Timing with $\overline{CTS}$

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	$\overline{CTS}$ Fall to $\overline{DE}$ Assertion Delay	2 * X_{IN} period	2 * X_{IN} period + 1 Bit period
T_2	$\overline{DE}$ Assertion to TXD Falling Edge (Start) Delay	1 Bit period	1 Bit period + 1 * X_{IN} period
T_3	End of Stop Bit(s) to $\overline{DE}$ Deassertion Delay	1 * X_{IN} period	2 * X_{IN} period

Figure 55 and Table 113 provide timing information for UART pins for the case where the Clear To Send input signal ($\overline{CTS}$) is not used for flow control. In this example, it is assumed that the Driver Enable polarity has been configured to be Active Low and is rep-

resented here by $\overline{DE}$. $\overline{DE}$ asserts after the UART Transmit Data Register has been written. $\overline{DE}$ remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

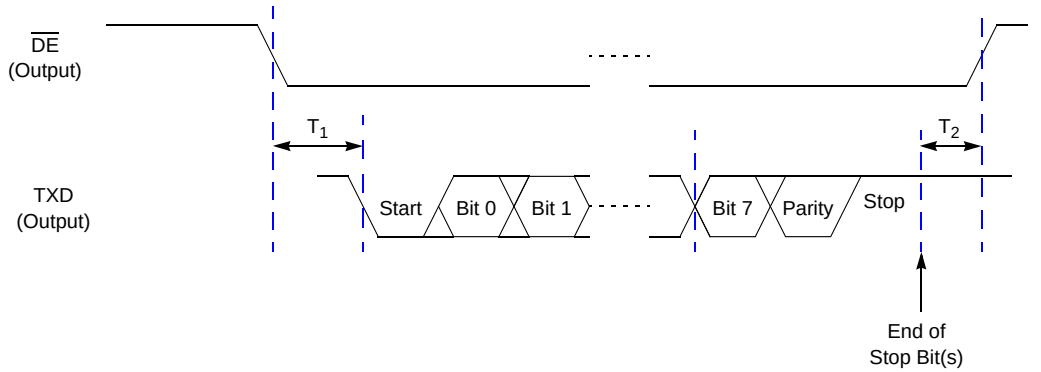


Figure 55. UART Timing without $\overline{CTS}$

Table 113. UART Timing without $\overline{CTS}$

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	$\overline{DE}$ Assertion to TXD Falling Edge (Start) Delay	1 Bit period	1 Bit period + 1 * X_{IN} period
T_2	End of Stop Bit(s) to $\overline{DE}$ Deassertion Delay	1 * X_{IN} period	2 * X_{IN} period

eZ8 CPU Instruction Set

This chapter describes the following features of the eZ8 CPU instruction set:

Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without having to be concerned with actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (op codes and operands) to represent the instructions themselves. The op codes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement can contain labels, operations, operands and comments.

Labels can be assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or pseudo-ops, are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the following example.

Assembly Language Source Program Example

```
JP START      ; Everything after the semicolon is a comment.
START:        ; A label called “START”. The first instruction (JP
              ; START) in this example causes program execution to
              ; jump to the point within the program where the
              ; START label occurs.
LD R4, R7     ; A Load (LD) instruction with two operands. The
              ; first operand, Working Register R4, is the
              ; destination. The second operand, Working Register
              ; R7, is the source. The contents of R7 is written
              ; into R4.
LD 234H, 01H  ; Another Load (LD) instruction with two operands.
              ; The first operand, Extended Mode Register Address
              ; 234H, is the destination. The second operand,
              ; Immediate Data value 01H, is the source. The value
              ; 01H is written into the Register at address 234H.
```

Assembly Language Syntax

For proper instruction execution, eZ8 CPU assembly language syntax requires that the operands be written as ‘destination, source’. After assembly, the object code usually has the operands in the order ’source, destination’, but ordering is op code-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. This binary format must be followed by users that prefer manual program coding or intend to implement their own assembler.

Example 1. If the contents of registers 43H and 08H are added and the result is stored in 43H, the assembly syntax and resulting object code result is shown in Table 114.

Table 114. Assembly Language Syntax Example 1

Assembly Language Code	ADD	43H	08H	(ADD dst, src)
Object Code	04	08	43	(OPC src, dst)

Example 2. In general, when an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or, using Escaped Mode Addressing, a Working Register R0–R15. If the contents of Register 43H and Working Register R8 are added and the result is stored in 43H, the assembly syntax and resulting object code result is shown in Table 115.

Table 115. Assembly Language Syntax Example 2

Assembly Language Code	ADD	43H,	R8	(ADD dst, src)
Object Code	04	E8	43	(OPC src, dst)

The register file size varies, depending on device type. See the device-specific Z8 Encore! XP Product Specification to determine the exact register file range available.

eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary and Description sections, the operands, condition codes, status flags, and address modes are represented by a notational shorthand that is described in Table 116.

Table 116. Notational Shorthand

Notation	Description	Operand	Range
b	Bit	b	b represents a value from 0 to 7 (000B to 111B).
cc	Condition Code	—	See the Condition Codes overview in the eZ8 CPU User Manual.
DA	Direct Address	Addrs	Addrs. represents a number in the range of 0000H to FFFFH
ER	Extended Addressing Register	Reg	Reg. represents a number in the range of 000H to FFFH
IM	Immediate Data	#Data	Data is a number between 00H to FFH
Ir	Indirect Working Register	@Rn	n = 0–15
IR	Indirect Register	@Reg	Reg. represents a number in the range of 00H to FFH
Irr	Indirect Working Register Pair	@RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
IRR	Indirect Register Pair	@Reg	*Reg* represents an even number in the range 00H to FEH
p	Polarity	p	Polarity is a single bit binary value of either 0B or 1B.
r	Working Register	Rn	n = 0 – 15
R	Register	Reg	*Reg* represents a number in the range of 00H to FFH

Table 116. Notational Shorthand (Continued)

Notation	Description	Operand	Range
RA	Relative Address	X	X represents an index in the range of +127 to –128, which is an offset relative to the address of the next instruction
rr	Working Register Pair	RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
RR	Register Pair	Reg	*Reg* represents an even number in the range of 00H to FEH
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 117 contains additional symbols that are used throughout the Instruction Summary and Instruction Set Description sections.

Table 117. 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.

Condition Codes

The C, Z, S, and V flags control the operation of the conditional jump (JP cc and JR cc) instructions. Sixteen frequently useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms Bits 7:4 of the conditional jump instructions. The condition codes are summarized in Table 118. Some binary condition codes can be created using more than one assembly code mnemonic. The result of the flag test operation decides if the conditional jump is executed.

Table 118. Condition Codes

Binary	Hex	Assembly Mnemonic	Definition	Flag Test Operation
0000	0	F	Always False	–
0001	1	LT	Less Than	(S XOR V) = 1
0010	2	LE	Less Than or Equal	(Z OR (S XOR V)) = 1
0011	3	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0100	4	OV	Overflow	V = 1
0101	5	MI	Minus	S = 1
0110	6	Z	Zero	Z = 1
0110	6	EQ	Equal	Z = 1
0111	7	C	Carry	C = 1
0111	7	ULT	Unsigned Less Than	C = 1
1000	8	T (or blank)	Always True	–
1001	9	GE	Greater Than or Equal	(S XOR V) = 0
1010	A	GT	Greater Than	(Z OR (S XOR V)) = 0
1011	B	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
1100	C	NOV	No Overflow	V = 0
1101	D	PL	Plus	S = 0
1110	E	NZ	Non-Zero	Z = 0
1110	E	NE	Not Equal	Z = 0
1111	F	NC	No Carry	C = 0
1111	F	UGE	Unsigned Greater Than or Equal	C = 0

eZ8 CPU Instruction Classes

eZ8 CPU instructions are divided functionally into the following groups:

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

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

Table 119. Arithmetic Instructions

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

Table 119. Arithmetic Instructions (Continued)

Mnemonic	Operands	Instruction
MULT	dst	Multiply
SBC	dst, src	Subtract with Carry
SBCX	dst, src	Subtract with Carry using Extended Addressing
SUB	dst, src	Subtract
SUBX	dst, src	Subtract using Extended Addressing

Table 120. Bit Manipulation Instructions

Mnemonic	Operands	Instruction
BCLR	bit, dst	Bit Clear
BIT	p, bit, dst	Bit Set or Clear
BSET	bit, dst	Bit Set
BSWAP	dst	Bit Swap
CCF	—	Complement Carry Flag
RCF	—	Reset Carry Flag
SCF	—	Set Carry Flag
TCM	dst, src	Test Complement Under Mask
TCMX	dst, src	Test Complement Under Mask using Extended Addressing
TM	dst, src	Test Under Mask
TMX	dst, src	Test Under Mask using Extended Addressing

Table 121. Block Transfer Instructions

Mnemonic	Operands	Instruction
LDCI	dst, src	Load Constant to/from Program memory and Auto-Increment Addresses
LDEI	dst, src	Load External Data to/from Data Memory and Auto-Increment Addresses

Table 122. CPU Control Instructions

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

Table 123. 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 124. 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 125. 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 126. Rotate and Shift Instructions

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

eZ8 CPU Instruction Summary

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

Table 127. eZ8 CPU Instruction Summary

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

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

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
ADD dst, src	dst ← dst + src	r	r	02	*	*	*	*	0	*	2	3
		r	Ir	03							2	4
		R	R	04							3	3
		R	IR	05							3	4
		R	IM	06							3	3
		IR	IM	07							3	4
ADDX dst, src	dst ← dst + src	ER	ER	08	*	*	*	*	0	*	4	3
		ER	IM	09							4	3
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
BCLR bit, dst	dst[bit] ← 0	r		E2	-	-	-	-	-	-	2	2
BIT p, bit, dst	dst[bit] ← p	r		E2	-	-	-	-	-	-	2	2
BRK	Debugger Break			00	-	-	-	-	-	-	1	1
BSET bit, dst	dst[bit] ← 1	r		E2	-	-	-	-	-	-	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 PC ← PC + X		r	F6	-	-	-	-	-	-	3	3
			Ir	F7							3	4
BTJNZ bit, src, dst	if src[bit] = 1 PC ← PC + X		r	F6	-	-	-	-	-	-	3	3
			Ir	F7							3	4
BTJZ bit, src, dst	if src[bit] = 0 PC ← PC + X		r	F6	-	-	-	-	-	-	3	3
			Ir	F7							3	4

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

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
CALL dst	SP ← SP -2 @SP ← PC 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
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

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

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
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 if dst ≠ 0 PC ← PC + X	r		0A-FA	-	-	-	-	-	-	2	3
EI	IRQCTL[7] ← 1			9F	-	-	-	-	-	-	1	2
HALT	HALT Mode			7F	-	-	-	-	-	-	1	2
INC dst	dst ← dst + 1	R		20	-	*	*	*	-	-	2	2
		IR		21							2	3
		r		0E-FE							1	2
INCW dst	dst ← dst + 1	RR		A0	-	*	*	*	-	-	2	5
		IRR		A1							2	6
IRET	FLAGS ← @SP SP ← SP + 1 PC ← @SP SP ← SP + 2 IRQCTL[7] ← 1			BF	*	*	*	*	*	*	1	5
JP dst	PC ← dst	DA		8D	-	-	-	-	-	-	3	2
		IRR		C4							2	3
JP cc, dst	if cc is true PC ← dst	DA		0D-FD	-	-	-	-	-	-	3	2
JR dst	PC ← PC + X	DA		8B	-	-	-	-	-	-	2	2
JR cc, dst	if cc is true PC ← PC + X	DA		0B-FB	-	-	-	-	-	-	2	2

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

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
LD dst, rc	dst ← src	r	IM	0C-FC	-	-	-	-	-	-	2	2
		r	X(r)	C7							3	3
		X(r)	r	D7							3	4
		r	Ir	E3							2	3
		R	R	E4							3	2
		R	IR	E5							3	4
		R	IM	E6							3	2
		IR	IM	E7							3	3
		Ir	r	F3							2	3
		IR	R	F5							3	3
LDC dst, src	dst ← src	r	Irr	C2	-	-	-	-	-	-	2	5
		Ir	Irr	C5							2	9
		Irr	r	D2							2	5
LDCI dst, src	dst ← src r ← r + 1 rr ← rr + 1	Ir	Irr	C3	-	-	-	-	-	-	2	9
		Irr	Ir	D3							2	9
LDE dst, src	dst ← src	r	Irr	82	-	-	-	-	-	-	2	5
		Irr	r	92							2	5
LDEI dst, src	dst ← src r ← r + 1 rr ← rr + 1	Ir	Irr	83	-	-	-	-	-	-	2	9
		Irr	Ir	93							2	9
LDWX dst, src	dst ← src	ER	ER	1F E8	-	-	-	-	-	-	5	4

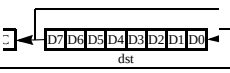
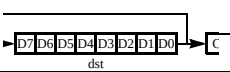
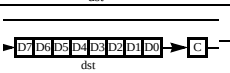
Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
LDX dst, src	dst ← src	r	ER	84	-	-	-	-	-	-	3	2
		Ir	ER	85							3	3
		R	IRR	86							3	4
		IR	IRR	87							3	5
		r	X(rr)	88							3	4
		X(rr)	r	89							3	4
		ER	r	94							3	2
		ER	Ir	95							3	3
		IRR	R	96							3	4
		IRR	IR	97							3	5
		ER	ER	E8							4	2
		ER	IM	E9							4	2
LEA dst, X(src)	dst ← src + X	r	X(r)	98	-	-	-	-	-	-	3	3
		rr	X(rr)	99							3	5
MULT dst	dst[15:0] ← dst[15:8] * dst[7:0]	RR		F4	-	-	-	-	-	-	2	8
NOP	No operation			0F	-	-	-	-	-	-	1	2
OR dst, src	dst ← dst OR src	r	r	42	-	*	*	0	-	-	2	3
		r	Ir	43							2	4
		R	R	44							3	3
		R	IR	45							3	4
		R	IM	46							3	3
		IR	IM	47							3	4
ORX dst, src	dst ← dst OR src	ER	ER	48	-	*	*	0	-	-	4	3
		ER	IM	49							4	3
POP dst	dst ← @SP SP ← SP + 1	R		50	-	-	-	-	-	-	2	2
		IR		51							2	3

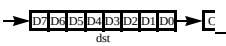
Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
POPX dst	dst ← @SP SP ← SP + 1	ER		D8	-	-	-	-	-	-	3	2
PUSH src	SP ← SP – 1 @SP ← src	R		70	-	-	-	-	-	-	2	2
		IR		71							2	3
PUSHX src	SP ← SP – 1 @SP ← src	ER		C8	-	-	-	-	-	-	3	2
RCF	C ← 0			CF	0	-	-	-	-	-	1	2
RET	PC ← @SP SP ← SP + 2			AF	-	-	-	-	-	-	1	4
RL dst		R		90	*	*	*	*	-	-	2	2
		IR		91							2	3
RLC dst		R		10	*	*	*	*	-	-	2	2
		IR		11							2	3
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

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

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
SRL dst		R		1F C0	*	*	0	*	-	-	3	2
		IR		1F C1							3	3
SRP src	RP ← src		IM	01	-	-	-	-	-	-	2	2
STOP	STOP Mode			6F	-	-	-	-	-	-	1	2
SUB dst, src	dst ← dst – src	r	r	22	*	*	*	*	1	*	2	3
		r	Ir	23							2	4
		R	R	24							3	3
		R	IR	25							3	4
		R	IM	26							3	3
		IR	IM	27							3	4
SUBX dst, src	dst ← dst – src	ER	ER	28	*	*	*	*	1	*	4	3
		ER	IM	29							4	3
SWAP dst	dst[7:4] ↔ dst[3:0]	R		F0	X	*	*	X	-	-	2	2
		IR		F1							2	3
TCM dst, src	(NOT dst) AND src	r	r	62	-	*	*	0	-	-	2	3
		r	Ir	63							2	4
		R	R	64							3	3
		R	IR	65							3	4
		R	IM	66							3	3
		IR	IM	67							3	4
TCMX dst, src	(NOT dst) AND src	ER	ER	68	-	*	*	0	-	-	4	3
		ER	IM	69							4	3

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

Table 127. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
TM dst, src	dst AND src	r	r	72	-	*	*	0	-	-	2	3
		r	Ir	73							2	4
		R	R	74							3	3
		R	IR	75							3	4
		R	IM	76							3	3
		IR	IM	77							3	4
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.

Flags Register

The Flags Register contains the status information regarding the most recent arithmetic, logical, bit manipulation or rotate and shift operation. The Flags Register contains six bits of status information that are set or cleared by CPU operations. Four of the bits (C, V, Z and S) can be tested with conditional jump instructions. Two flags (H and D) cannot be tested and are used for Binary-Coded Decimal (BCD) arithmetic.

The two remaining bits, User Flags (F1 and F2), are available as general-purpose status bits. User Flags are unaffected by arithmetic operations and must be set or cleared by instructions. The User Flags cannot be used with conditional Jumps. They are undefined at initial power-up and are unaffected by Reset. Figure 56 illustrates the flags and their bit positions in the Flags Register.

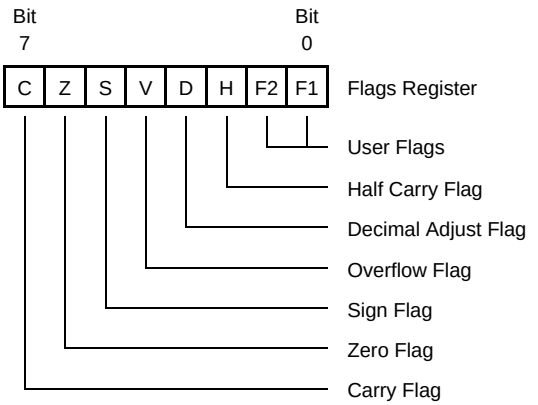


Figure 56. Flags Register

Interrupts, the Software Trap (TRAP) instruction, and Illegal Instruction Traps all write the value of the Flags Register to the stack. Executing an Interrupt Return (IRET) instruction restores the value saved on the stack into the Flags Register.

Op Code Maps

A description of the Op Code map data and the abbreviations are provided in Figure 57 and Table 128. Figures 58 and 59 provide information about each of the eZ8 CPU instructions.

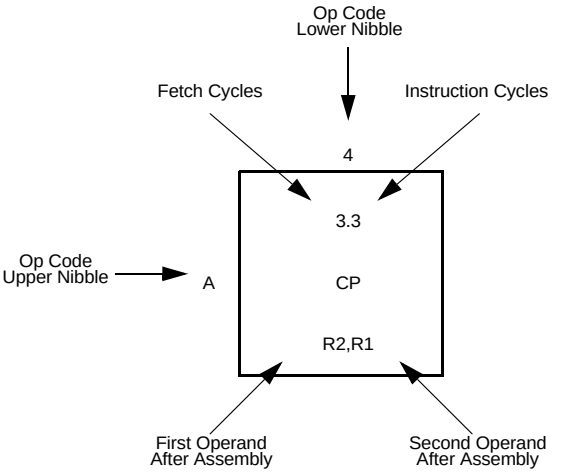
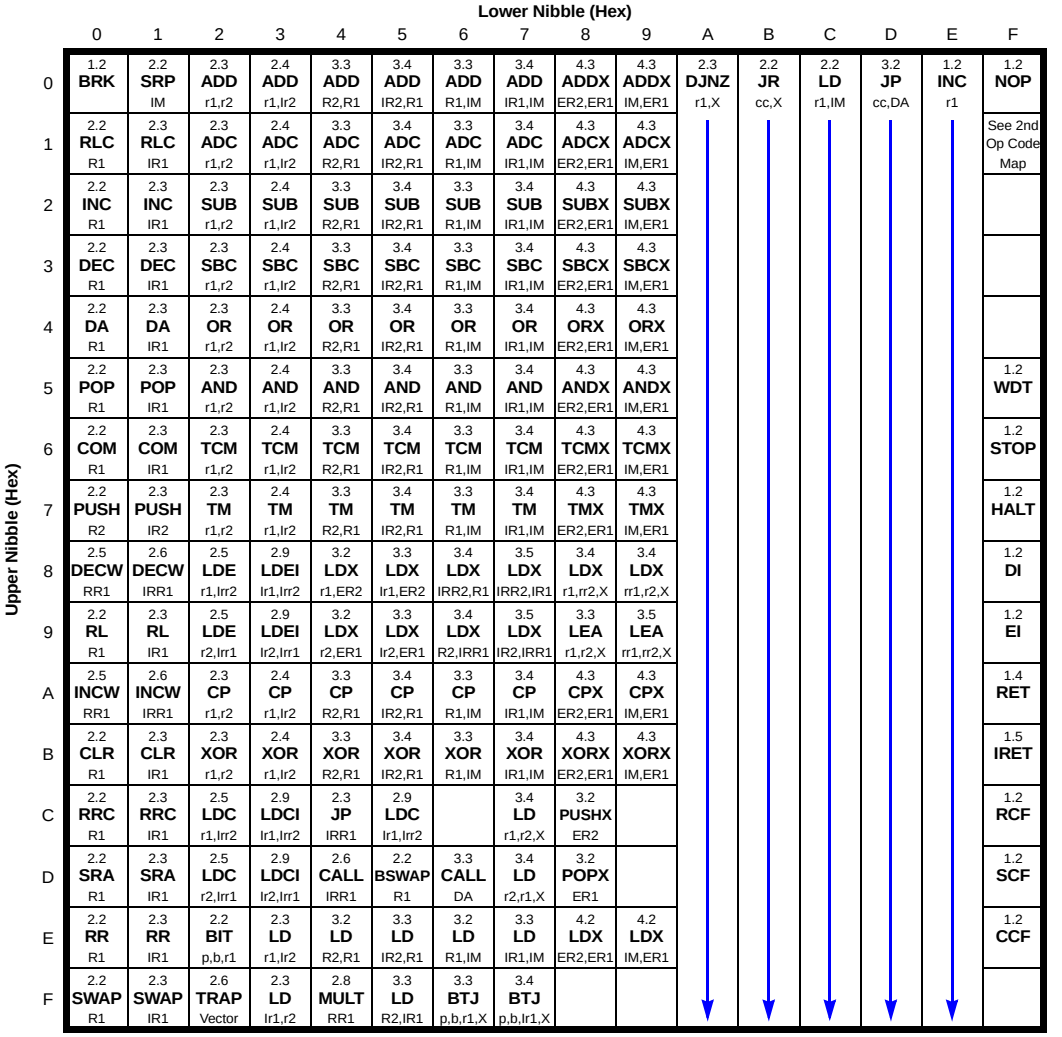


Figure 57. Op Code Map Cell Description

Table 128. Op Code Map Abbreviations

Abbreviation	Description	Abbreviation	Description
b	Bit position	IRR	Indirect register pair
cc	Condition code	p	Polarity (0 or 1)
X	8-bit signed index or displacement	r	4-bit working register
DA	Destination address	R	8-bit register
ER	Extended addressing register	r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1	Destination address
IM	Immediate data value	r2, R2, Ir2, Irr2, IR2, rr2, RR2, IRR2, ER2	Source address
Ir	Indirect working register	RA	Relative
IR	Indirect register	rr	Working register pair
Irr	Indirect working register pair	RR	Register pair



Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1.2 **BRK**	2.2 **SRP** IM	2.3 **ADD** r1,r2	2.4 **ADD** r1,Ir2	3.3 **ADD** R2,R1	3.4 **ADD** IR2,R1	3.3 **ADD** R1,IM	3.4 **ADD** IR1,IM	4.3 **ADDX** ER2,ER1	4.3 **ADDX** IM,ER1	2.3 **DJNZ** r1,X	2.2 **JR** cc,X	2.2 **LD** r1,IM	3.2 **JP** cc,DA	1.2 **INC** r1	1.2 **NOP**
1	2.2 **RLC** R1	2.3 **RLC** IR1	2.3 **ADC** r1,r2	2.4 **ADC** r1,Ir2	3.3 **ADC** R2,R1	3.4 **ADC** IR2,R1	3.3 **ADC** R1,IM	3.4 **ADC** IR1,IM	4.3 **ADCX** ER2,ER1	4.3 **ADCX** IM,ER1						See 2nd Op Code Map
2	2.2 **INC** R1	2.3 **INC** IR1	2.3 **SUB** r1,r2	2.4 **SUB** r1,Ir2	3.3 **SUB** R2,R1	3.4 **SUB** IR2,R1	3.3 **SUB** R1,IM	3.4 **SUB** IR1,IM	4.3 **SUBX** ER2,ER1	4.3 **SUBX** IM,ER1						
3	2.2 **DEC** R1	2.3 **DEC** IR1	2.3 **SBC** r1,r2	2.4 **SBC** r1,Ir2	3.3 **SBC** R2,R1	3.4 **SBC** IR2,R1	3.3 **SBC** R1,IM	3.4 **SBC** IR1,IM	4.3 **SBCX** ER2,ER1	4.3 **SBCX** IM,ER1						
4	2.2 **DA** R1	2.3 **DA** IR1	2.3 **OR** r1,r2	2.4 **OR** r1,Ir2	3.3 **OR** R2,R1	3.4 **OR** IR2,R1	3.3 **OR** R1,IM	3.4 **OR** IR1,IM	4.3 **ORX** ER2,ER1	4.3 **ORX** IM,ER1						
5	2.2 **POP** R1	2.3 **POP** IR1	2.3 **AND** r1,r2	2.4 **AND** r1,Ir2	3.3 **AND** R2,R1	3.4 **AND** IR2,R1	3.3 **AND** R1,IM	3.4 **AND** IR1,IM	4.3 **ANDX** ER2,ER1	4.3 **ANDX** IM,ER1						1.2 **WDT**
6	2.2 **COM** R1	2.3 **COM** IR1	2.3 **TCM** r1,r2	2.4 **TCM** r1,Ir2	3.3 **TCM** R2,R1	3.4 **TCM** IR2,R1	3.3 **TCM** R1,IM	3.4 **TCM** IR1,IM	4.3 **TCMX** ER2,ER1	4.3 **TCMX** IM,ER1						1.2 **STOP**
7	2.2 **PUSH** R2	2.3 **PUSH** IR2	2.3 **TM** r1,r2	2.4 **TM** r1,Ir2	3.3 **TM** R2,R1	3.4 **TM** IR2,R1	3.3 **TM** R1,IM	3.4 **TM** IR1,IM	4.3 **TMX** ER2,ER1	4.3 **TMX** IM,ER1						1.2 **HALT**
8	2.5 **DECW** RR1	2.6 **DECW** IRR1	2.5 **LDE** r1,Irr2	2.9 **LDEI** Ir1,Irr2	3.2 **LDX** r1,ER2	3.3 **LDX** Ir1,ER2	3.4 **LDX** IRR2,R1	3.5 **LDX** IRR2,IR1	3.4 **LDX** r1,rr2,X	3.4 **LDX** rr1,r2,X						1.2 **DI**
9	2.2 **RL** R1	2.3 **RL** IR1	2.5 **LDE** r2,Irr1	2.9 **LDEI** Ir2,Irr1	3.2 **LDX** r2,ER1	3.3 **LDX** Ir2,ER1	3.4 **LDX** R2,IRR1	3.5 **LDX** IR2,IRR1	3.3 **LEA** r1,r2,X	3.5 **LEA** rr1,rr2,X						1.2 **EI**
A	2.5 **INCW** RR1	2.6 **INCW** IRR1	2.3 **CP** r1,r2	2.4 **CP** r1,Ir2	3.3 **CP** R2,R1	3.4 **CP** IR2,R1	3.3 **CP** R1,IM	3.4 **CP** IR1,IM	4.3 **CPX** ER2,ER1	4.3 **CPX** IM,ER1						1.4 **RET**
B	2.2 **CLR** R1	2.3 **CLR** IR1	2.3 **XOR** r1,r2	2.4 **XOR** r1,Ir2	3.3 **XOR** R2,R1	3.4 **XOR** IR2,R1	3.3 **XOR** R1,IM	3.4 **XOR** IR1,IM	4.3 **XORX** ER2,ER1	4.3 **XORX** IM,ER1						1.5 **IRET**
C	2.2 **RRC** R1	2.3 **RRC** IR1	2.5 **LDC** r1,Irr2	2.9 **LDCI** Ir1,Irr2	2.3 **JP** IRR1	2.9 **LDC** Ir1,Irr2		3.4 **LD** r1,r2,X	3.2 **PUSHX** ER2							1.2 **RCF**
D	2.2 **SRA** R1	2.3 **SRA** IR1	2.5 **LDC** r2,Irr1	2.9 **LDCI** Ir2,Irr1	2.6 **CALL** IRR1	2.2 **BSWAP** R1	3.3 **CALL** DA	3.4 **LD** r2,r1,X	3.2 **POPX** ER1							1.2 **SCF**
E	2.2 **RR** R1	2.3 **RR** IR1	2.2 **BIT** p,b,r1	2.3 **LD** r1,Ir2	3.2 **LD** R2,R1	3.3 **LD** IR2,R1	3.2 **LD** R1,IM	3.3 **LD** IR1,IM	4.2 **LDX** ER2,ER1	4.2 **LDX** IM,ER1						1.2 **CCF**
F	2.2 **SWAP** R1	2.3 **SWAP** IR1	2.6 **TRAP** Vector	2.3 **LD** Ir1,r2	2.8 **MULT** RR1	3.3 **LD** R2,IR1	3.3 **BTJ** p,b,r1,X	3.4 **BTJ** p,b,Ir1,X								

Figure 58. First Op Code Map

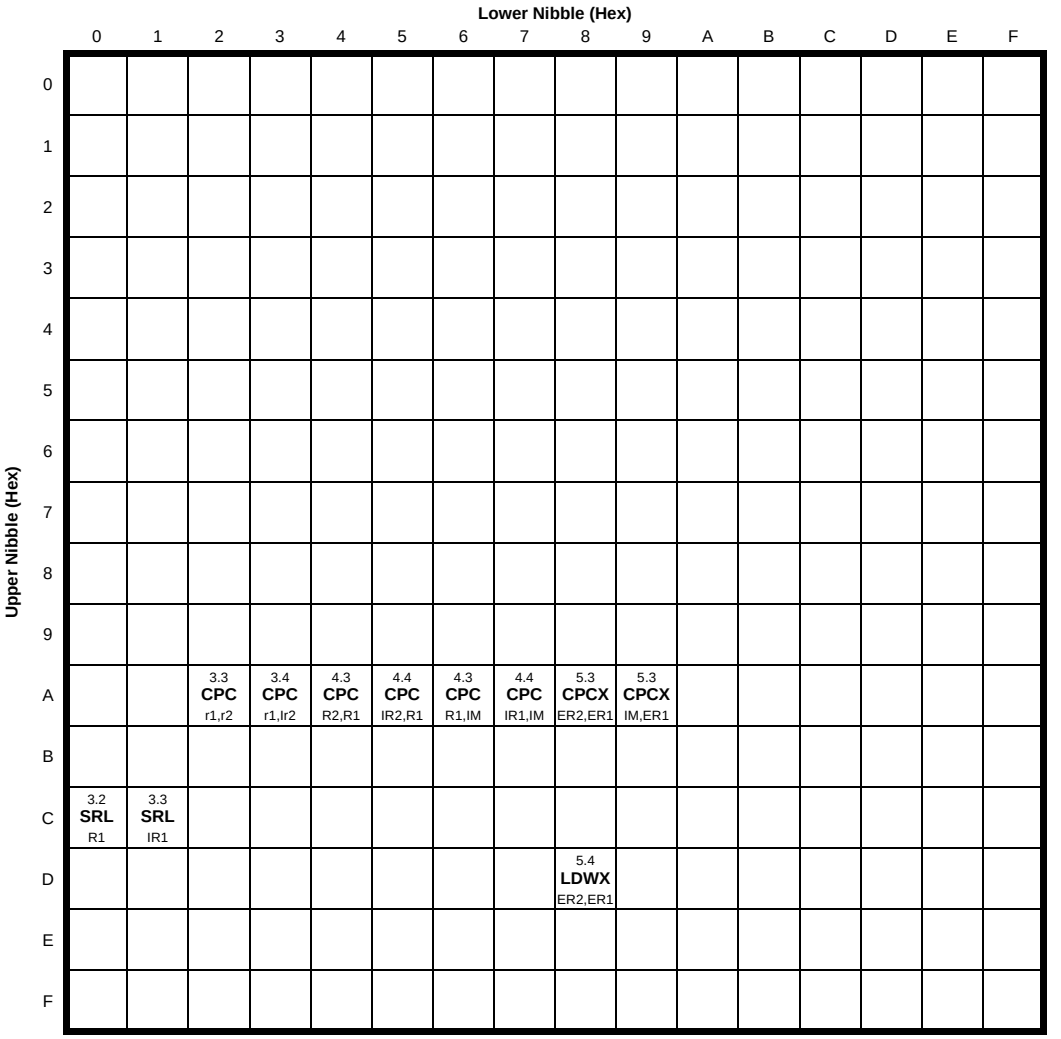


Figure 59. Second Op Code Map after 1FH

Packaging

Zilog's Z8 Encore! XP® F0822 Series of MCUs includes the Z8F0411, Z8F0421, Z8F0811 and Z8F0821 devices, which are available in the following packages:

- 20-pin Small Shrink Outline Package (SSOP)
- 20-pin Plastic Dual-Inline Package (PDIP)

Zilog's Z8 Encore! XP® F0822 Series of MCUs also includes the Z8F0412, Z8F0422, Z8F0812 and Z8F0822 devices, which are available in the following packages:

- 28-pin Small Outline Integrated Circuit Package (SOIC)
- 28-pin Plastic Dual-Inline Package (PDIP)

Current diagrams for each of these packages are published in Zilog's Packaging Product Specification (PS0072), which is available free for download from the Zilog website.

Ordering Information

Order your Z8 Encore! XP® F0822 Series products from Zilog using the part numbers shown in Table 129. For more information about ordering, please consult your local Zilog sales office. The Sales Location page on the Zilog website lists all regional offices.

Table 129. Z8 Encore! XP F0830 Series Ordering Matrix

Part Number	Flash	RAM	I/O Lines	Interrupts	16-Bit Timers w/PWM	10-Bit A/D Channels	I^2C	SPI	UARTs with IrDA	Description
Z8F08xx with 8KB Flash, 10-Bit Analog-to-Digital Converter										
Standard Temperature: 0°C to 70°C										
Z8F0821HH020SG	8KB	1KB	11	16	2	2	1	0	1	SSOP 20-pin package
Z8F0821PH020SG	8KB	1KB	11	16	2	2	1	0	1	PDIP 20-pin package
Z8F0822SJ020SG	8KB	1KB	19	19	2	5	1	1	1	SOIC 28-pin package
Z8F0822PJ020SG	8KB	1KB	19	19	2	5	1	1	1	PDIP 28-pin package
Extended Temperature: –40° to +105°C										
Z8F0821HH020EG	8KB	1KB	11	16	2	2	1	0	1	SSOP 20-pin package
Z8F0821PH020EG	8KB	1KB	11	16	2	2	1	0	1	PDIP 20-pin package
Z8F0822SJ020EG	8KB	1KB	19	19	2	5	1	1	1	SOIC 28-pin package
Z8F0822PJ020EG	8KB	1KB	19	19	2	5	1	1	1	PDIP 28-pin package
Z8F08xx with 8KB Flash										
Standard Temperature: 0°C to 70°C										
Z8F0811HH020SG	8KB	1KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0811PH020SG	8KB	1KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0812SJ020SG	8KB	1KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0812PJ020SG	8KB	1KB	19	19	2	0	1	1	1	PDIP 28-pin package
Extended Temperature: –40°C to +105°C										
Z8F0811HH020EG	8KB	1KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0811PH020EG	8KB	1KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0812SJ020EG	8KB	1KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0812PJ020EG	8KB	1KB	19	19	2	0	1	1	1	PDIP 28-pin package

Table 129. Z8 Encore! XP F0830 Series Ordering Matrix

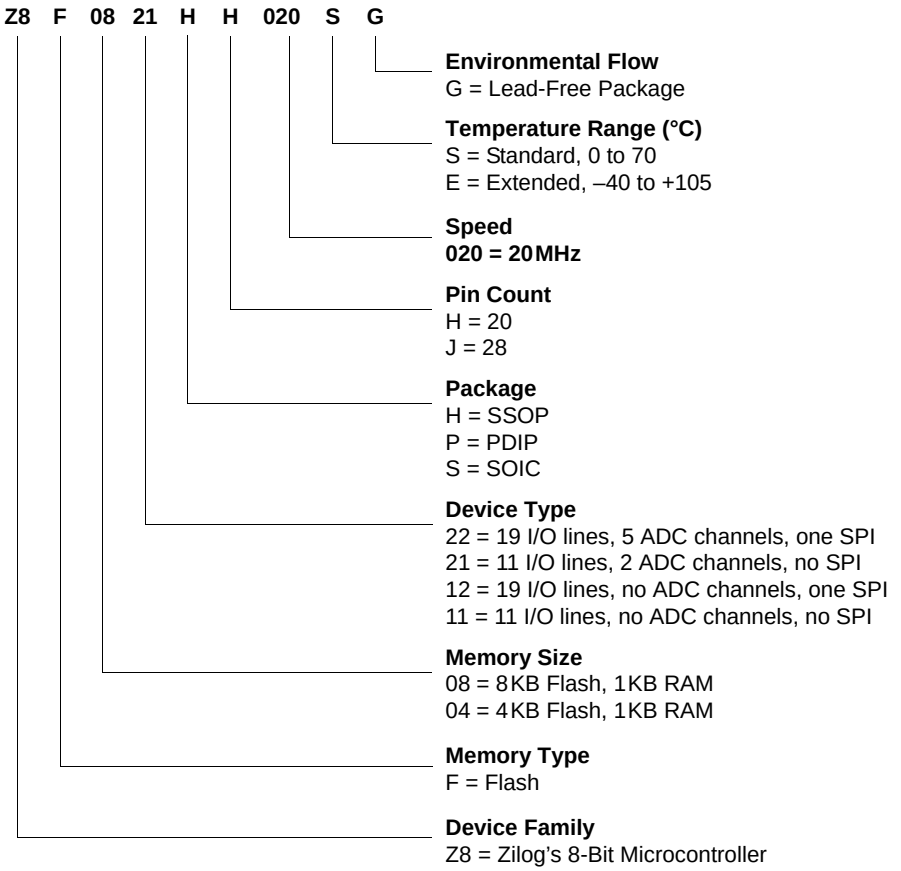
Part Number	Flash	RAM	I/O Lines	Interrupts	16-Bit Timers w/PWM	10-Bit A/D Channels	I^2C	SPI	UARTs with IrDA	Description
Z8F04xx with 4KB Flash, 10-Bit Analog-to-Digital Converter										
Standard Temperature: 0°C to 70°C										
Z8F0421HH020SG	4KB	1KB	11	16	2	2	1	0	1	SSOP 20-pin package
Z8F0421PH020SG	4KB	1KB	11	16	2	2	1	0	1	PDIP 20-pin package
Z8F0422SJ020SG	4KB	1KB	19	19	2	5	1	1	1	SOIC 28-pin package
Z8F0422PJ020SG	4KB	1KB	19	19	2	5	1	1	1	PDIP 28-pin package
Extended Temperature: –40°C to 105°C										
Z8F0421HH020EG	4KB	1KB	11	16	2	2	1	0	1	SSOP 20-pin package
Z8F0421PH020EG	4KB	1KB	11	16	2	2	1	0	1	PDIP 20-pin package
Z8F0422SJ020EG	4KB	1KB	19	19	2	5	1	1	1	SOIC 28-pin package
Z8F0422PJ020EG	4KB	1KB	19	19	2	5	1	1	1	PDIP 28-pin package
Z8F04xx with 4KB Flash										
Standard Temperature: 0°C to 70°C										
Z8F0411HH020SG	4KB	1KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0411PH020SG	4KB	1KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0412SJ020SG	4KB	1KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0412PJ020SG	4KB	1KB	19	19	2	0	1	1	1	PDIP 28-pin package
Extended Temperature: –40°C to 105°C										
Z8F0411HH020EG	4KB	1KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0411PH020EG	4KB	1KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0412SJ020EG	4KB	1KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0412PJ020EG	4KB	1KB	19	19	2	0	1	1	1	PDIP 28-pin package
Z8F08200100KITG	Development Kit (20- and 28-pin)									
ZUSBSC00100ZACG	USB Smart Cable Accessory Kit									
ZUSBOPTSC01ZACG	Opto-Isolated USB Smart Cable Accessory Kit									

Visit the Zilog website at http://www.zilog.com for ordering information about Z8 Encore! XP® F0822 Series development tools and accessories.

Part Number Suffix Designations

Zilog part numbers consist of a number of components, as indicated in the following example.

Example. Part number Z8F0821HH020SG is an 8-bit Flash Motor Controller with 8KB of Program Memory, equipped with 11 I/O lines and 2 ADC channels in a 20-pin SSOP package, operating within a 0ºC to +70ºC temperature range and built using lead-free solder.



Appendix A. Register Summary

For the reader's convenience, this appendix lists all Z8 Encore! XP® F0822 Series registers numerically by hexadecimal address.

General Purpose RAM

In the Z8 Encore! XP® F0822 Series, the 000–EFF hexadecimal address range is partitioned for general-purpose random access memory, as follows.

Hex Addresses: 000–1FF

This address range is reserved for general-purpose register file RAM. For more details, see the Register File Address Map chapter on page 17.

Hex Addresses: 200–EFF

This address range is reserved.

Timer 0 Control Registers

For more information about these Timer Control registers, see the Timer Control Register Definitions section on page 64.

Hex Address: F00

Table 130. Timer 0–1 High Byte Register (TxH)

Bit	7	6	5	4	3	2	1	0
Field	TH							
RESET	0							
R/W	R/W							
Address	F00H, F08H							

Hex Address: F01

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

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

Hex Address: F02

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

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

Hex Address: F03

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

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

Hex Address: F04

Table 134. Timer 0–1 PWM High Byte Register (TxPWMH)

Bit	7	6	5	4	3	2	1	0
Field	PWMH							
RESET	0							
R/W	R/W							
Address	F04H, F0CH							

Hex Address: F05

Table 135. Timer 0–1 PWM Low Byte Register (TxPWML)

Bit	7	6	5	4	3	2	1	0
Field	PWML							
RESET	0							
R/W	R/W							
Address	F05H, F0DH							

Hex Address: F06

Table 136. Timer 0–3 Control 0 Registers (TxCTL0)

Bit	7	6	5	4	3	2	1	0
Field	Reserved			CSC	Reserved			
RESET	0							
R/W	R/W							
Address	F06H, F0EH, F16H, F1EH							

Hex Address: F07

Table 137. Timer 0–1 Control Registers (TxCTL)

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

Hex Address: F08

Table 138. Timer 0–1 High Byte Register (TxH)

Bit	7	6	5	4	3	2	1	0
Field	TH							
RESET	0							
R/W	R/W							
Address	F00H, F08H							

Hex Address: F09

Table 139. Timer 0–1 Low Byte Register (TxL)

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

Hex Address: F0A

Table 140. Timer 0–1 Reload High Byte Register (TxRH)

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

Hex Address: F0B

Table 141. Timer 0–1 Reload Low Byte Register (TxRL)

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

Hex Address: F0C

Table 142. Timer 0–1 PWM High Byte Register (TxPWMH)

Bit	7	6	5	4	3	2	1	0
Field	PWMH							
RESET	0							
R/W	R/W							
Address	F04H, F0CH							

Hex Address: F0D

Table 143. Timer 0–1 PWM Low Byte Register (TxPWML)

Bit	7	6	5	4	3	2	1	0
Field	PWML							
RESET	0							
R/W	R/W							
Address	F05H, F0DH							

Hex Address: F0E

Table 144. Timer 0–3 Control 0 Registers (TxCTL0)

Bit	7	6	5	4	3	2	1	0
Field	Reserved			CSC	Reserved			
RESET	0							
R/W	R/W							
Address	F06H, F0EH, F16H, F1EH							

Hex Address: F0F

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

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

Hex Addresses: F10–F3F

This address range is reserved.

UART Control Registers

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

Hex Address: F40

Table 146. UART Transmit Data Register (U0TXD)

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

Table 147. UART Receive Data Register (U0RXD)

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

Hex Address: F41

Table 148. UART Status 0 Register (U0STAT0)

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

Hex Address: F42

Table 149. UART Control 0 Register (U0CTL0)

Bit	7	6	5	4	3	2	1	0
Field	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
RESET	0							
R/W	R/W							
Address	F42H							

Hex Address: F43

Table 150. UART Control 1 Register (U0CTL1)

Bit	7	6	5	4	3	2	1	0
Field	MPMD[1]	MPEN	MPMD[0]	MPBT	DEPOL	BRGCTL	RDAIRQ	IREN
RESET	0							
R/W	R/W							
Address	F43H							

Hex Address: F44

Table 151. UART Status 1 Register (U0STAT1)

Bit	7	6	5	4	3	2	1	0
Field	Reserved						NEWFRM	MPRX
RESET	0							
R/W	R				R/W		R	
Address	F44H							

Hex Address: F45

Table 152. UART Address Compare Register (U0ADDR)

Bit	7	6	5	4	3	2	1	0
Field	COMP_ADDR							
RESET	0							
R/W	R/W							
Address	F45H							

Hex Address: F46

Table 153. UART Baud Rate High Byte Register (U0BRH)

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

Hex Address: F47

Table 154. UART Baud Rate Low Byte Register (U0BRL)

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

Hex Addresses: F48–F4F

This address range is reserved.

I²C Control Registers

For more information about the I²C registers, see the I2C Control Register Definitions section on page 128.

Hex Address: F50

Table 155. I²C Data Register (I2CDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	0							
R/W	R/W							
Address	F50H							

Hex Address: F51

Table 156. I²C Status Register (I2CSTAT)

Bit	7	6	5	4	3	2	1	0
Field	TDRE	RDRF	ACK	10B	RD	TAS	DSS	NCKI
RESET	1	0						
R/W	R							
Address	F51H							

Hex Address: F52

Table 157. I²C Control Register (I2CCTL)

Bit	7	6	5	4	3	2	1	0
Field	IEN	START	STOP	BIRQ	TXI	NAK	FLUSH	FILTEN
RESET	0							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	W	R/W
Address	F52H							

Hex Address: F53

Table 158. I²C Baud Rate High Byte Register (I2CBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFH							
R/W	R/W							
Address	F53H							

Hex Address: F54

Table 159. I²C Baud Rate Low Byte Register (I2CBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFH							
R/W	R/W							
Address	F54H							

Hex Address: F55

Table 160. I²C Diagnostic State Register (I2CDST)

Bit	7	6	5	4	3	2	1	0
Field	SCLIN	SDAIN	STPCNT	TXRXSTATE				
RESET	X		0					
R/W	R							
Address	F55H							

Hex Address: F56

Table 161. I²C Diagnostic Control Register (I2CDIAG)

Bit	7	6	5	4	3	2	1	0
Field	Reserved							DIAG
RESET	0							
R/W	R							R/W
Address	F56H							

Hex Addresses: F57–F5F

This address range is reserved.

SPI Control Registers

For more information about the SPI registers, see the SPI Control Register Definitions section on page 109.

Hex Address: F60

Table 162. SPI Data Register (SPIDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	X							
R/W	R/W							
Address	F60H							

Hex Address: F61

Table 163. SPI Control Register (SPICTL)

Bit	7	6	5	4	3	2	1	0
Field	IRQE	STR	BIRQ	PHASE	CLKPOL	WOR	MMEN	SPIEN
RESET	0							
R/W	R/W							
Address	F61H							

Hex Address: F62

Table 164. SPI Status Register (SPISTAT)

Bit	7	6	5	4	3	2	1	0
Field	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0							1
R/W	R/W*				R			
Address	F62H							

Note: *R/W = read access; write a 1 to clear the bit to 0.

Hex Address: F63

Table 165. SPI Mode Register (SPIMODE)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		DIAG	NUMBITS[2:0]			SSIO	SSV
RESET	0							
R/W	R		R/W					
Address	F63H							

Hex Address: F64

Table 166. SPI Diagnostic State Register (SPIDST)

Bit	7	6	5	4	3	2	1	0
Field	SCKEN	TCKEN	SPISTATE					
RESET	0							
R/W	R							
Address	F64H							

Hex Address: F65

This address is reserved.

Hex Address: F66

Table 167. SPI Baud Rate High Byte Register (SPIBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	1							
R/W	R/W							
Address	F66H							

Hex Address: F67

Table 168. SPI Baud Rate Low Byte Register (SPIBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	1							
R/W	R/W							
Address	F67H							

Hex Addresses: F68–F6F

This address range is reserved.

Analog-to-Digital Converter Control Registers

For more information about these ADC registers, see the ADC Control Register Definitions section on page 139.

Hex Address: F70

Table 169. ADC Control Register (ADCCTL)

Bit	7	6	5	4	3	2	1	0
Field	CEN	Reserved	VREF	CONT	ANAIN[3:0]			
RESET	0		1	0				
R/W	R/W							
Address	F70H							

Hex Address: F71

This address is reserved.

Hex Address: F72

Table 170. ADC Data High Byte Register (ADCD_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCD_H							
RESET	X							
R/W	R							
Address	F72H							

Hex Address: F73

Table 171. ADC Data Low Bits Register (ADCD_L)

Bit	7	6	5	4	3	2	1	0
Field	ADCD_L		Reserved					
RESET	X							
R/W	R							
Address	F73H							

Hex Addresses: F74–FBF

This address range is reserved.

Interrupt Request Control Registers

For more information about the IRQ registers, see the Interrupt Control Register Definitions section on page 45.

Hex Address: FC0

Table 172. Interrupt Request 0 Register (IRQ0)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	T1I	T0I	U0RXI	U0TXI	I2CI	SPII	ADCI
RESET	0							
R/W	R/W							
Address	FC0H							

Hex Address: FC1

Table 173. IRQ0 Enable High Bit Register (IRQ0ENH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	T1ENH	T0ENH	U0RENH	U0TENH	I2CENH	SPIENH	ADCENH
RESET	0							
R/W	R/W							
Address	FC1H							

Hex Address: FC2

Table 174. IRQ0 Enable Low Bit Register (IRQ0ENL)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	T1ENL	T0ENL	U0RENL	U0TENL	I2CENL	SPIENL	ADCENL
RESET	0							
R/W	R/W							
Address	FC2H							

Hex Address: FC3

Table 175. Interrupt Request 1 Register (IRQ1)

Bit	7	6	5	4	3	2	1	0
Field	PA7I	PA6I	PA5I	PA4I	PA3I	PA2I	PA1I	PA0I
RESET	0							
R/W	R/W							
Address	FC3H							

Hex Address: FC4

Table 176. IRQ1 Enable High Bit Register (IRQ1ENH)

Bit	7	6	5	4	3	2	1	0
Field	PA7ENH	PA6ENH	PA5ENH	PA4ENH	PA3ENH	PA2ENH	PA1ENH	PA0ENH
RESET	0							
R/W	R/W							
Address	FC4H							

Hex Address: FC5

Table 177. IRQ1 Enable Low Bit Register (IRQ1ENL)

Bit	7	6	5	4	3	2	1	0
Field	PA7ENL	PA6ENL	PA5ENL	PA4ENL	PA3ENL	PA2ENL	PA1ENL	PA0ENL
RESET	0							
R/W	R/W							
Address	FC5H							

Hex Address: FC6

Table 178. Interrupt Request 2 Register (IRQ2)

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

Hex Address: FC7

Table 179. IRQ2 Enable High Bit Register (IRQ2ENH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				C3ENH	C2ENH	C1ENH	C0ENH
RESET	0							
R/W	R/W							
Address	FC7H							

Hex Address: FC8

Table 180. IRQ2 Enable Low Bit Register (IRQ2ENL)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				C3ENL	C2ENL	C1ENL	C0ENL
RESET	0							
R/W	R/W							
Address	FC8H							

Hex Addresses: FC9–FCE

This address range is reserved.

Hex Address: FCF

Table 181. Interrupt Control Register (IRQCTL)

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

GPIO Control Registers

For more information about the GPIO control registers, see the GPIO Control Register Definitions section on page 31.

Hex Address: FD0

Table 182. Port A–C GPIO Address Registers (P*x*ADDR)

Bit	7	6	5	4	3	2	1	0
Field	PADDR[7:0]							
RESET	00H							
R/W	R/W							
Address	FD0H, FD4H, FD8H							

Hex Address: FD1

Table 183. Port A–C Control Registers (P*x*CTL)

Bit	7	6	5	4	3	2	1	0
Field	PCTL							
RESET	00H							
R/W	R/W							
Address	FD1H, FD5H, FD9H							

Hex Address: FD2

Table 184. Port A–C Input Data Registers (PxIN)

Bit	7	6	5	4	3	2	1	0
Field	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X							
R/W	R							
Address	FD2H, FD6H, FDAH							

Hex Address: FD3

Table 185. Port A–C Output Data Register (PxOUT)

Bit	7	6	5	4	3	2	1	0
Field	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0							
R/W	R/W							
Address	FD3H, FD7H, FDBH							

Hex Address: FD4

Table 186. Port A–C GPIO Address Registers (PxADDR)

Bit	7	6	5	4	3	2	1	0
Field	PADDR[7:0]							
RESET	00H							
R/W	R/W							
Address	FD0H, FD4H, FD8H							

Hex Address: FD5

Table 187. Port A–C Control Registers (PxCTL)

Bit	7	6	5	4	3	2	1	0
Field	PCTL							
RESET	00H							
R/W	R/W							
Address	FD1H, FD5H, FD9H							

Hex Address: FD6

Table 188. Port A–C Input Data Registers (PxIN)

Bit	7	6	5	4	3	2	1	0
Field	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X							
R/W	R							
Address	FD2H, FD6H, FDAH							

Hex Address: FD7

Table 189. Port A–C Output Data Register (PxOUT)

Bit	7	6	5	4	3	2	1	0
Field	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0							
R/W	R/W							
Address	FD3H, FD7H, FDBH							

Hex Address: FD8

Table 190. Port A–C GPIO Address Registers (PxADDR)

Bit	7	6	5	4	3	2	1	0
Field	PADDR[7:0]							
RESET	00H							
R/W	R/W							
Address	FD0H, FD4H, FD8H							

Hex Address: FD9

Table 191. Port A–C Control Registers (PxCTL)

Bit	7	6	5	4	3	2	1	0
Field	PCTL							
RESET	00H							
R/W	R/W							
Address	FD1H, FD5H, FD9H							

Hex Address: FDA

Table 192. Port A–C Input Data Registers (PxIN)

Bit	7	6	5	4	3	2	1	0
Field	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X							
R/W	R							
Address	FD2H, FD6H, FDAH							

Hex Address: FDB

Table 193. Port A–C Output Data Register (PxOUT)

Bit	7	6	5	4	3	2	1	0
Field	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0							
R/W	R/W							
Address	FD3H, FD7H, FDBH							

Hex Addresses: FDC–FEF

This address range is reserved.

Watchdog Timer Control Registers

For more information about the Watchdog Timer control registers, see the Watchdog Timer Control Register Definitions section on page 73.

Hex Address: FF0

Table 194. Watchdog Timer Control Register (WDTCTL)

Bit	7	6	5	4	3	2	1	0
Field	POR	STOP	WDT	EXT	Reserved			
RESET	See Table 49 on page 74.				0			
R/W	R							
Address	FF0H							

Hex Address: FF1

Table 195. Watchdog Timer Reload Upper Byte Register (WDTU)

Bit	7	6	5	4	3	2	1	0
Field	WDTU							
RESET	1							
R/W	R/W*							
Address	FF1H							
Note: *R/W = a read returns the current WDT count value; a write sets the appropriate reload value.								

Hex Address: FF2

Table 196. Watchdog Timer Reload High Byte Register (WDTH)

Bit	7	6	5	4	3	2	1	0
Field	WDTH							
RESET	1							
R/W	R/W*							
Address	FF2H							
Note: *R/W = a read returns the current WDT count value; a write sets the appropriate reload value.								

Hex Address: FF3

Table 197. Watchdog Timer Reload Low Byte Register (WDTL)

Bit	7	6	5	4	3	2	1	0
Field	WDTL							
RESET	1							
R/W	R/W*							
Address	FF3H							
Note: *R/W = a read returns the current WDT count value; a write sets the appropriate reload value.								

Hex Addresses: FF4–FF7

This address range is reserved.

Flash Control Registers

For more information about the Flash control registers, see the Flash Control Register Definitions section on page 149.

Hex Address: FF8

Table 198. Flash Control Register (FCTL)

Bit	7	6	5	4	3	2	1	0
Field	FCMD							
RESET	0							
R/W	W							
Address	FF8H							

Table 199. Flash Status Register (FSTAT)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		FSTAT					
RESET	0							
R/W	R							
Address	FF8H							

Hex Address: FF9

Table 200. Page Select Register (FPS)

Bit	7	6	5	4	3	2	1	0
Field	INFO_EN	PAGE						
RESET	0							
R/W	R/W							
Address	FF9H							

Table 201. Flash Sector Protect Register (FPROT)

Bit	7	6	5	4	3	2	1	0
Field	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
RESET	0							
R/W	R/W*							
Address	FF9H							

Note: *R/W = this register is accessible for read operations, but can only be written to 1 (via user code).

Hex Address: FFA

Table 202. Flash Frequency High Byte Register (FFREQH)

Bit	7	6	5	4	3	2	1	0
Field	FFREQH							
RESET	0							
R/W	R/W							
Address	FFAH							

Hex Address: FFB

Table 203. Flash Frequency Low Byte Register (FFREQL)

Bit	7	6	5	4	3	2	1	0
Field	FFREQL							
RESET	0							
R/W	R/W							
Address	FFBH							

Hex Addresses: FFC–FFF

This address range is reserved.

Index

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

Customer Support

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

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

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

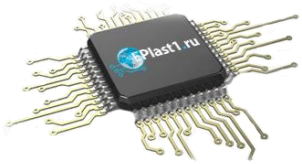
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



Как с нами связаться

Телефон: 8 (812) 309 58 32 (многоканальный)
Факс: 8 (812) 320-02-42
Электронная почта: org@eplast1.ru
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.