



High Performance 8-Bit Microcontrollers

**Z8 Encore! XP[®] F0822
Series**

Product Specification

PS022517-0508



Warning: DO NOT USE IN LIFE SUPPORT

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

©2008 by Zilog, Inc. All rights reserved. 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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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.

Z8, Z8 Encore!, Z8 Encore! XP, Z8 Encore! MC, Crimzon, eZ80, and ZNEO are trademarks or registered trademarks of Zilog, Inc. All other product or service names are the property of their respective owners.



ISO 9001:2000
FS 507510

Zilog products are designed and manufactured under an ISO registered 9001:2000 Quality Management System. For more details, please visit www.zilog.com/quality.



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 Number
May 2008	17	Removed Flash Microcontrollers from the title throughout the document.	All
February 2008	16	Updated the flag status for BCLR, BIT, and BSET in Table 126.	219
December 2007	15	Updated Zilog logo, Zilog text, Disclaimer section, and implemented style guide. Updated Z8 Encore! 8K Series to Z8 Encore! XP F0822 Series Flash Microcontrollers throughout the document.	All
June 2007 and August 2007	13 and 14	No Changes.	All
December 2006	12	Updated Ordering Information and minor edits done.	All



Table of Contents

Introduction	x
About This Manual	x
Intended Audience	x
Manual Conventions	x
Safeguards	xii
Abbreviations/Acronyms	xii
Introduction	1
Features	1
Part Selection Guide	2
Block Diagram	3
CPU and Peripheral Overview	3
eZ8 CPU Features	3
General Purpose Input/Output	4
Flash Controller	4
10-Bit Analog-to-Digital Converter	4
UART	4
I ² C	5
Serial Peripheral Interface	5
Timers	5
Interrupt Controller	5
Reset Controller	5
On-Chip Debugger	5
Signal and Pin Descriptions	7
Available Packages	7
Pin Configurations	7
Signal Descriptions	9
Pin Characteristics	12
Address Space	13
Register File	13
Program Memory	13
Data Memory	14
Information Area	14
Register File Address Map	15
Control Register Summary	19
Reset and Stop Mode Recovery	39
Reset Types	39
System Reset	40



Reset Sources	40
Power-On Reset.	41
Voltage Brownout Reset.	41
Watchdog Timer Reset.	42
External Pin Reset	43
On-Chip Debugger Initiated Reset	43
Stop Mode Recovery	43
Stop Mode Recovery Using WDT Time-Out	44
Stop Mode Recovery Using a GPIO Port Pin Transition	44
Low-Power Modes	45
STOP Mode	45
HALT Mode	45
General-Purpose Input/Output	47
GPIO Port Availability by Device	47
Architecture	47
GPIO Alternate Functions	47
GPIO Interrupts	49
GPIO Control Register Definitions	49
Port A–C Address Registers	50
Port A–C Control Registers	51
Port A–C Input Data Registers	54
Port A–C Output Data Register	55
Interrupt Controller	57
Interrupt Vector Listing	57
Architecture	59
Operation	59
Master Interrupt Enable	59
Interrupt Vectors and Priority	60
Interrupt Assertion	60
Software Interrupt Assertion	60
Interrupt Control Register Definitions	61
Interrupt Request 0 Register	61
Interrupt Request 1 Register	62
Interrupt Request 2 Register	63
IRQ0 Enable High and Low Bit Registers	63
IRQ1 Enable High and Low Bit Registers	64
IRQ2 Enable High and Low Bit Registers	65
Interrupt Edge Select Register	67
Interrupt Control Register	67
Timers	69
Architecture	69



Operation	69
Timer Operating Modes	70
Reading the Timer Count Values	77
Timer Output Signal Operation	78
Timer Control Register Definitions	78
Timer 0–1 High and Low Byte Registers	78
Timer Reload High and Low Byte Registers	79
Timer 0–1 PWM High and Low Byte Registers	79
Timer 0–3 Control 0 Registers	80
Timer 0–1 Control 1 Registers	81
Watchdog Timer	83
Operation	83
Watchdog Timer Refresh	84
Watchdog Timer Time-Out Response	84
Watchdog Timer Reload Unlock Sequence	85
Watchdog Timer Control Register Definitions	86
Watchdog Timer Control Register	86
Watchdog Timer Reload Upper, High and Low Byte Registers	87
Universal Asynchronous Receiver/Transmitter	89
Architecture	89
Operation	90
Data Format	90
Transmitting Data using Polled Method	91
Transmitting Data Using Interrupt-Driven Method	92
Receiving Data using the Polled Method	93
Receiving Data Using Interrupt-Driven Method	94
Clear To Send Operation	95
Multiprocessor (9-bit) Mode	95
External Driver Enable	96
UART Interrupts	97
UART Baud Rate Generator	99
UART Control Register Definitions	100
UART Transmit Data Register	100
UART Receive Data Register	101
UART Status 0 Register	101
UART Status 1 Register	102
UART Control 0 and Control 1 Registers	103
UART Address Compare Register	105
UART Baud Rate High and Low Byte Registers	106
Infrared Encoder/Decoder	109
Architecture	109



Operation	109
Transmitting IrDA Data	110
Receiving IrDA Data	111
Infrared Endec Control Register Definitions	112
Serial Peripheral Interface	113
Architecture	113
Operation	114
SPI Signals	115
SPI Clock Phase and Polarity Control	116
Multi-Master Operation	118
Slave Operation	118
Error Detection	119
SPI Interrupts	119
SPI Baud Rate Generator	120
SPI Control Register Definitions	121
SPI Data Register	121
SPI Control Register	122
SPI Status Register	123
SPI Mode Register	124
SPI Diagnostic State Register	125
SPI Baud Rate High and Low Byte Registers	125
I2C Controller	127
Architecture	127
Operation	128
SDA and SCL Signals	128
I ² C Interrupts	128
Software Control of I2C Transactions	129
Start and Stop Conditions	130
Master Write and Read Transactions	130
Address Only Transaction with a 7-bit Address	131
Write Transaction with a 7-Bit Address	132
Address Only Transaction with a 10-bit Address	133
Write Transaction with a 10-Bit Address	134
Read Transaction with a 7-Bit Address	136
Read Transaction with a 10-Bit Address	137
I2C Control Register Definitions	139
I2C Data Register	139
I2C Status Register	140
I2C Control Register	141
I2C Baud Rate High and Low Byte Registers	143
I2C Diagnostic State Register	143



I2C Diagnostic Control Register	145
Analog-to-Digital Converter	147
Architecture	147
Operation	148
Automatic Power-Down	148
Single-Shot Conversion	148
Continuous Conversion	148
ADC Control Register Definitions	150
ADC Control Register	150
ADC Data High Byte Register	151
ADC Data Low Bits Register	151
Flash Memory	153
Information Area	154
Operation	155
Timing Using the Flash Frequency Registers	155
Flash Read Protection	156
Flash Write/Erase Protection	156
Byte Programming	157
Page Erase	158
Mass Erase	158
Flash Controller Bypass	158
Flash Controller Behavior in Debug Mode	159
Flash Control Register Definitions	159
Flash Control Register	159
Flash Status Register	160
Page Select Register	160
Flash Sector Protect Register	161
Flash Frequency High and Low Byte Registers	161
Option Bits	163
Operation	163
Option Bit Configuration By Reset	163
Option Bit Address Space	163
Flash Memory Address 0000H	164
Flash Memory Address 0001H	165
On-Chip Oscillator	167
Operating Modes	167
Crystal Oscillator Operation	167
Oscillator Operation with an External RC Network	168
On-Chip Debugger	171
Architecture	171
Operation	171



OCD Interface	171
Debug Mode	173
OCD Data Format	173
OCD Auto-Baud Detector/Generator	174
OCD Serial Errors	174
Breakpoints	175
OCDCNTR Register	176
On-Chip Debugger Commands	176
On-Chip Debugger Control Register Definitions	181
OCD Control Register	181
OCD Status Register	183
Electrical Characteristics	185
Absolute Maximum Ratings	185
DC Characteristics	187
AC Characteristics	194
On-Chip Peripheral AC and DC Electrical Characteristics	195
General Purpose I/O Port Input Data Sample Timing	200
General Purpose I/O Port Output Timing	201
On-Chip Debugger Timing	202
SPI MASTER Mode Timing	203
SPI SLAVE Mode Timing	204
I2C Timing	205
UART Timing	206
eZ8 CPU Instruction Set	209
Assembly Language Programming Introduction	209
Assembly Language Syntax	210
eZ8 CPU Instruction Notation	210
Condition Codes	213
eZ8 CPU Instruction Classes	214
eZ8 CPU Instruction Summary	218
Flags Register	227
Opcode Maps	229
Packaging	233
Ordering Information	236
Part Number Suffix Designations	240
Index	241
Customer Support	251

Introduction

This Product Specification provides detailed operating information for Z8 Encore! XP[®] F0822 Series devices within the Z8 Encore! XP Microcontroller (MCU) family of products. Within this document, Z8 Encore! XP[®] F0822 Series is referred as Z8 Encore! XP or the F0822 Series unless specifically stated otherwise.

About This Manual

Zilog recommends that you read and understand everything in this manual before setting up and using the product. We have designed this Product Specification to be used either as a *how to* procedural manual or a reference guide to important data.

Intended Audience

This document is written for Zilog customers who are experienced at working with microcontrollers, integrated circuits, or printed circuit assemblies.

Manual Conventions

The following assumptions and conventions are adopted to provide clarity and ease of use:

Courier Typeface

Commands, code lines and fragments, bits, equations, hexadecimal addresses, and various executable items are distinguished from general text by the use of the `Courier` typeface. Where the use of the font is not indicated, as in the Index, the name of the entity is presented in upper case.

- **Example:** `FLAGS[1]` is `smrf`.

Hexadecimal Values

Hexadecimal values are designated by uppercase *H* suffix and appear in the `Courier` typeface.

- **Example:** R1 is set to `F8H`.

Brackets

The square brackets `[ ]`, indicate a register or bus.

- **Example:** For the register `R1[7:0]`, R1 is an 8-bit register, `R1[7]` is the most significant bit, and `R1[0]` is the least significant bit.

Braces

The curly braces { }, indicate a single register or bus created by concatenating some combination of smaller registers, buses, or individual bits.

- **Example:** The 12-bit register address {0H, RP[7:4], R1[3:0]} is composed of a 4-bit hexadecimal value (0H) and two 4-bit register values taken from the Register Pointer (RP) and Working Register R1. 0H is the most significant nibble (4-bit value) of the 12-bit register, and R1[3:0] is the least significant nibble of the 12-bit register.

Parentheses

The parentheses (), indicate an indirect register address lookup.

- **Example:** (R1) is the memory location referenced by the address contained in the Working Register R1.

Parentheses/Bracket Combinations

The parentheses (), indicate an indirect register address lookup and the square brackets, [], indicate a register or bus.

- **Example:** Assume PC[15:0] contains the value 1234h. (PC [15:0]) then refers to the contents of the memory location at address 1234h.

Use of the Words *Set*, *Reset* and *Clear*

The word *set* implies that a register bit or a condition contains a logical 1. The words *reset* or *clear* imply that a register bit or a condition contains a logical 0. When either of these terms is followed by a number, the word *logical* cannot be included; however, it is implied.

Notation for Bits and Similar Registers

A field of bits within a register is designated as: Register[n:n].

- **Example:** ADDR[15:0] refers to bits 15 through bit 0 of the Address.

Use of the Terms *LSB*, *MSB*, *lsb*, and *msb*

In this document, the terms *LSB* and *MSB*, when appearing in upper case, mean *least significant byte* and *most significant byte*, respectively. The lowercase forms, *lsb* and *msb*, mean *least significant bit* and *most significant bit*, respectively.

Use of Initial Uppercase Letters

Initial uppercase letters designate settings and conditions in general text.

- **Example 1:** The receiver forces the SCL line to Low.
- **Example 2:** The Master generates a STOP condition to abort the transfer.

Use of All Uppercase Letters

The use of all uppercase letters designates the names of states, modes, and commands.

- **Example 1:** The bus is considered BUSY after the Start condition.
- **Example 2:** A START command triggers the processing of the initialization sequence.
- **Example 3:** STOP mode.

Bit Numbering

Bits are numbered from 0 to $n-1$ where n indicates the total number of bits. For example, the 8 bits of a register are numbered from 0 to 7.

Safeguards

It is important that you understand the following safety terms, which are defined here.



Caution: *Indicates a procedure or file can become corrupted if you does not follow directions.*

Abbreviations/Acronyms

This document uses the following abbreviations or acronyms.

Abbreviations/ Acronyms	Expansion
ADC	Analog-to-Digital Converter
LPO	Low-Power Operational Amplifier
SPI	Serial Peripheral Interface
WDT	Watchdog Timer
GPIO	General-Purpose Input/Output
OCD	On-Chip Debugger
POR	Power-On Reset
LVD	Low-Voltage Detection
VBO	Voltage Brownout
ISR	Interrupt Service Routine
UART	Universal Asynchronous Receiver/Transmitter
IrDA	Infrared Data Association
I ² C	Inter-Integrated Circuit



Abbreviations/ Acronyms	Expansion
PDIP	Plastic Dual Inline Package
SOIC	Small Outline Integrated Circuit
SSOP	Small Shrink Outline Package
PC	Program Counter
IRQ	Interrupt Request

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, hereafter referred as Z8 Encore! XP or the 8K Series 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[®] instructions. The rich peripheral set of Z8 Encore! XP makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Features

The features of Z8 Encore! XP MCU product family include:

- 20 MHz eZ8 CPU core
- Up to 8 KB Flash with in-circuit programming capability
- 1 KB 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 (I²C)
- 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 Brownout (VBO) protection
- Power-On Reset (POR)
- Crystal oscillator with three power settings and RC oscillator option

- 2.7 V to 3.6 V operating voltage with 5 V-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 product line.

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 ² C	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.

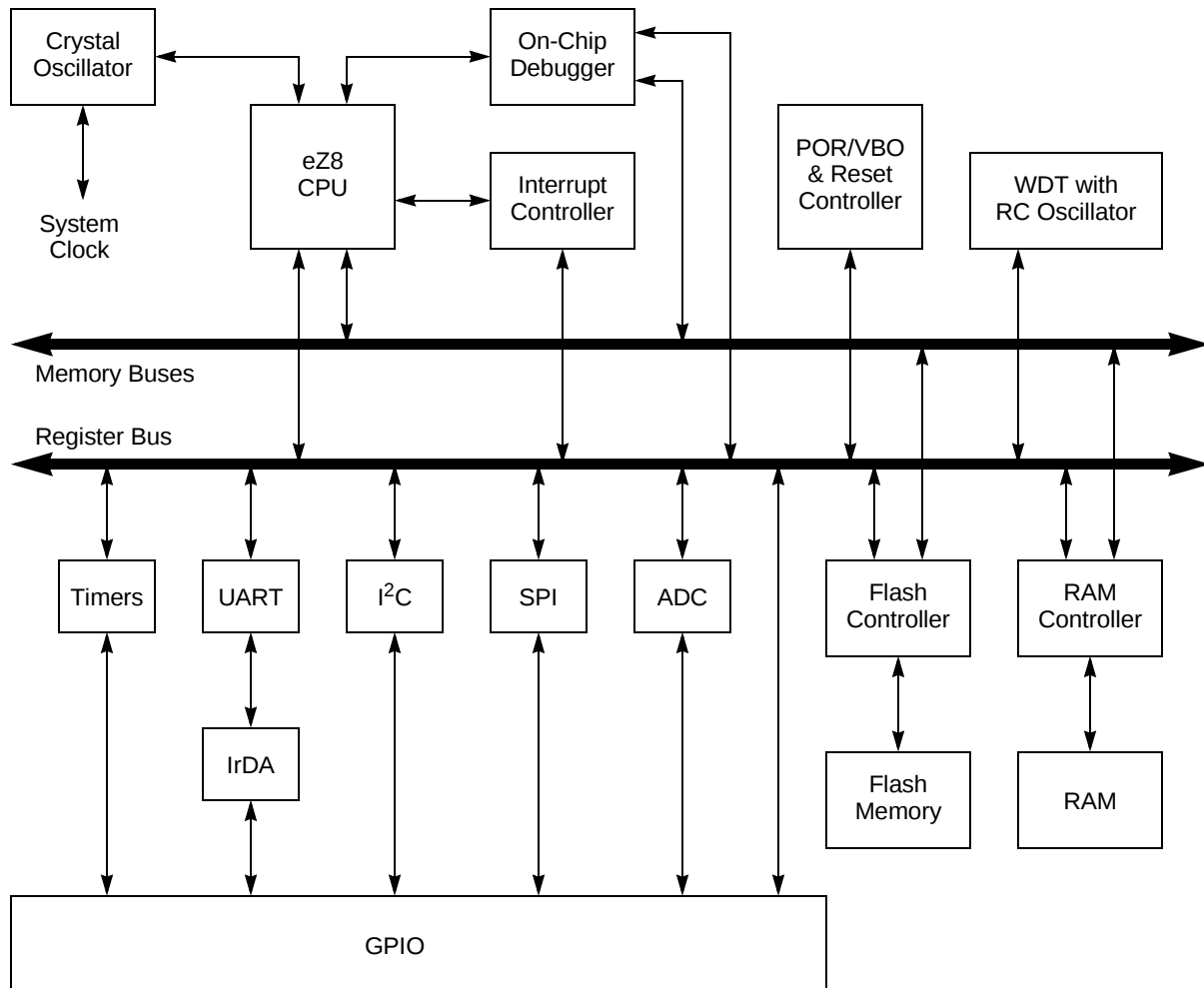


Figure 1. Z8 Encore! XP[®] F0822 Series Block Diagram

CPU and Peripheral Overview

eZ8 CPU Features

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 include:

- 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 4 KB.
- 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 regarding the eZ8 CPU, refer to *eZ8 CPU Core User Manual (UM0128)* 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 GPIO pins available is a function of package. Each pin is individually programmable. Ports A and C supports 5 V-tolerant inputs.

Flash Controller

The Flash Controller programs and erases the 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.

I²C

The Inter-Integrated Circuit (I²C) controller makes the Z8 Encore! XP compatible with the I²C protocol. The I²C 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 the Flash, 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 [Packaging](#) on page 233.

Available Packages

[Table 2](#) identifies the package styles available for each device within Z8 Encore! XP F0822 Series product line.

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

[Figure 2](#) through [Figure 5](#) display the pin configurations for all of the packages available in Z8 Encore! XP F0822 Series. See [Table 4](#) for a description of the signals.

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

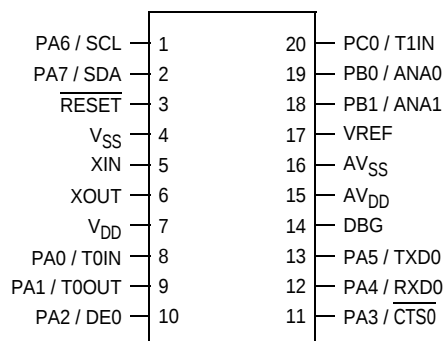


Figure 2. Z8F0821 and Z8F0421 in 20-Pin SSOP and PDIP Packages

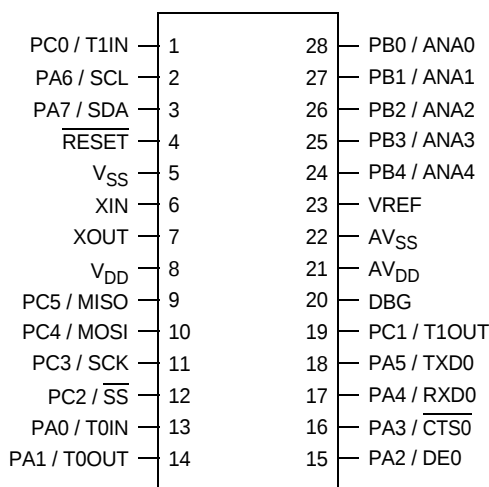


Figure 3. Z8F0822 and Z8F0422 in 28-Pin SOIC and PDIP Packages

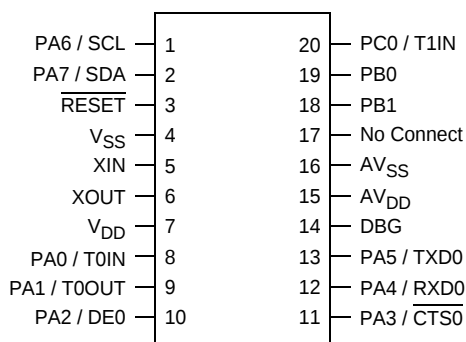


Figure 4. Z8F0811 and Z8F0411 in 20-Pin SSOP and PDIP Packages

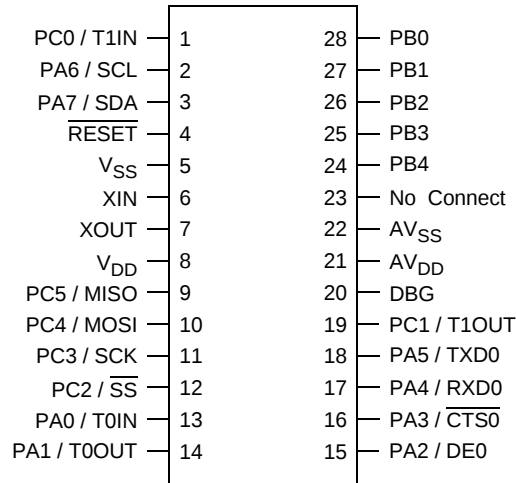


Figure 5. Z8F0812 and Z8F0412 in 28-Pin SOIC and PDIP Packages

Signal Descriptions

Table 3 describes Z8 Encore! XP[®] F0822 Series signals. See [Pin Configurations](#) 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 5 V-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 5 V-tolerant inputs.
I²C Controller		
SCL	I/O	Serial Clock —This open-drain pin clocks data transfers in accordance with the I ² C 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 ² C 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.

Table 3. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
SPI Controller		
$\overline{SS}$	I/O	Slave Select —This signal can be an output or an input. If the Z8 Encore! XP [®] is the SPI Master, this pin can be configured as the Slave Select output. If the Z8 Encore! XP 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 is the SPI Master, this pin is the output. If the Z8 Encore! XP 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.
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.
$\overline{CTS0}$	I	Clear To Send —This signal is control inputs for the UART. The $\overline{CTS}$ 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.
VREF	I	Analog-to-Digital Converter reference voltage input —As an output, the VREF signal is not recommended for use as a reference voltage for external devices. If the ADC is configured to use the internal reference voltage generator, this pin should be left unconnected or capacitively coupled to analog ground (AVSS).

Table 3. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
Oscillators		
XIN	I	External Crystal Input —This is the input pin to the crystal oscillator. A crystal is connected between the external crystal input and the XOUT 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.
XOUT	O	External Crystal Output —This pin is the output of the crystal oscillator. A crystal is connected between external crystal output and the XIN 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 be left 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. This pin is open-drain.
 Caution: <i>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 have an external pull-up resistor to ensure proper operation.</i>		
Reset		
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 V_{DD} , even if not using analog features.
V_{SS}	I	Digital Ground.
AV_{SS}	I	Analog Ground —Must be grounded and connected to V_{SS} , even if not using analog features.

Pin Characteristics

Table 4 provides detailed information on the characteristics for each pin available on Z8 Encore! XP[®] F0822 Series products. Table 4 data is sorted alphabetically by the 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
VREF	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
XIN	I	I	N/A	N/A	No	No	N/A
XOUT	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 on the eZ8 CPU and its address space, refer to *eZ8 CPU Core User Manual (UM0128)* available for download at www.zilog.com.

Register File

The Register File address space in the Z8 Encore! XP® is 4 KB (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 1 KB 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 and 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 1 KB 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 64 KB of Program Memory address space. Z8 Encore! XP® F0822 Series contain 4 KB to 8 KB 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 57 for a list of the interrupt vectors.	

Data Memory

Z8 Encore! XP[®] F0822 Series does not use the eZ8 CPU's 64 KB 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 as 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	78
F01	Timer 0 Low Byte	T0L	01	78
F02	Timer 0 Reload High Byte	T0RH	FF	79
F03	Timer 0 Reload Low Byte	T0RL	FF	79
F04	Timer 0 PWM High Byte	T0PWMH	00	79
F05	Timer 0 PWM Low Byte	T0PWML	00	79
F06	Timer 0 Control 0	T0CTL0	00	81
F07	Timer 0 Control 1	T0CTL1	00	81
Timer 1				
F08	Timer 1 High Byte	T1H	00	78
F09	Timer 1 Low Byte	T1L	01	78
F0A	Timer 1 Reload High Byte	T1RH	FF	79
F0B	Timer 1 Reload Low Byte	T1RL	FF	79
F0C	Timer 1 PWM High Byte	T1PWMH	00	79
F0D	Timer 1 PWM Low Byte	T1PWML	00	79
F0E	Timer 1 Control 0	T1CTL0	00	81
F0F	Timer 1 Control 1	T1CTL1	00	81
F10-F3F	Reserved	—	XX	
UART 0				
F40	UART0 Transmit Data	U0TXD	XX	100
	UART0 Receive Data	U0RXD	XX	101
F41	UART0 Status 0	U0STAT0	0000011Xb	101
F42	UART0 Control 0	U0CTL0	00	103
F43	UART0 Control 1	U0CTL1	00	103
F44	UART0 Status 1	U0STAT1	00	101
F45	UART0 Address Compare Register	U0ADDR	00	105
F46	UART0 Baud Rate High Byte	U0BRH	FF	106
XX=Undefined				

Table 7. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
F47	UART0 Baud Rate Low Byte	U0BRL	FF	106
F48-F4F	Reserved	—	XX	
I²C				
F50	I ² C Data	I2CDATA	00	139
F51	I ² C Status	I2CSTAT	80	140
F52	I ² C Control	I2CCTL	00	141
F53	I ² C Baud Rate High Byte	I2CBRH	FF	143
F54	I ² C Baud Rate Low Byte	I2CBRL	FF	143
F55	I ² C Diagnostic State	I2CDST	XX000000b	143
F56	I ² C Diagnostic Control	I2CDIAG	00	145
F57-F5F	Reserved	—	XX	
Serial Peripheral Interface (SPI) Unavailable in 20-Pin Package Devices				
F60	SPI Data	SPIDATA	01	121
F61	SPI Control	SPICTL	00	122
F62	SPI Status	SPISTAT	00	123
F63	SPI Mode	SPIMODE	00	124
F64	SPI Diagnostic State	SPIDST	00	125
F65	Reserved	—	XX	
F66	SPI Baud Rate High Byte	SPIBRH	FF	125
F67	SPI Baud Rate Low Byte	SPIBRL	FF	125
F68-F6F	Reserved	—	XX	
Analog-to-Digital Converter (ADC)				
F70	ADC Control	ADCCTL	20	150
F71	Reserved	—	XX	
F72	ADC Data High Byte	ADCD_H	XX	151
F73	ADC Data Low Bits	ADCD_L	XX	151
F74-FBF	Reserved	—	XX	
Interrupt Controller				
FC0	Interrupt Request 0	IRQ0	00	61
FC1	IRQ0 Enable High Bit	IRQ0ENH	00	63
FC2	IRQ0 Enable Low Bit	IRQ0ENL	00	63
FC3	Interrupt Request 1	IRQ1	00	62
FC4	IRQ1 Enable High Bit	IRQ1ENH	00	64
FC5	IRQ1 Enable Low Bit	IRQ1ENL	00	64
FC6	Interrupt Request 2	IRQ2	00	63
FC7	IRQ2 Enable High Bit	IRQ2ENH	00	65
FC8	IRQ2 Enable Low Bit	IRQ2ENL	00	65
FC9-FCC	Reserved	—	XX	
FCD	Interrupt Edge Select	IRQES	00	67
XX=Undefined				



Table 7. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FCE	Reserved	—	00	
FCF	Interrupt Control	IRQCTL	00	67
GPIO Port A				
FD0	Port A Address	PAADDR	00	50
FD1	Port A Control	PACTL	00	51
FD2	Port A Input Data	PAIN	XX	54
FD3	Port A Output Data	PAOUT	00	55
GPIO Port B				
FD4	Port B Address	PBADDR	00	50
FD5	Port B Control	PBCTL	00	51
FD6	Port B Input Data	PBIN	XX	54
FD7	Port B Output Data	PBOUT	00	55
GPIO Port C				
FD8	Port C Address	PCADDR	00	50
FD9	Port C Control	PCCTL	00	51
FDA	Port C Input Data	PCIN	XX	54
FDB	Port C Output Data	PCOUT	00	55
FDC-FEF	Reserved	—	XX	
Watchdog Timer (WDT)				
FF0	Watchdog Timer Control	WDTCTL	XXX00000b	86
FF1	Watchdog Timer Reload Upper Byte	WDTU	FF	87
FF2	Watchdog Timer Reload High Byte	WDTH	FF	87
FF3	Watchdog Timer Reload Low Byte	WDTL	FF	87
FF4-FF7	Reserved	—	XX	
Flash Memory Controller				
FF8	Flash Control	FCTL	00	159
FF8	Flash Status	FSTAT	00	160
FF9	Page Select	FPS	00	160
FF9 (if enabled)	Flash Sector Protect	FPROT	00	161
FFA	Flash Programming Frequency High Byte	FFREQH	00	161
FFB	Flash Programming Frequency Low Byte	FFREQL	00	161
Read-Only Memory				
FF8	Reserved	—	XX	
FF9	Page Select	RPS	00	160
FFA-FFB	Reserved	—	XX	
eZ8 CPU				
XX=Undefined				



Table 7. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FFC	Flags	—	XX	Refer to eZ8
FFD	Register Pointer	RP	XX	CPU User
FFE	Stack Pointer High Byte	SPH	XX	Manual
FFF	Stack Pointer Low Byte	SPL	XX	
XX=Undefined				

Control Register Summary

Timer 0 High Byte

T0H (F00H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

Timer 0 current count value

Timer 0 Low Byte

T0L (F01H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

Timer 0 current count value

Timer 0 Reload High Byte

T0RH (F02H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

Timer 0 reload value [15:8]

Timer 0 Reload Low Byte

T0RL (F03H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

Timer 0 reload value [7:0]

Timer 0 PWM High Byte

T0PWMH (F04H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

Timer 0 PWM value [15:8]

Timer 0 Control 0

T0CTL0 (F06H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

Reserved
Cascade Timer
0 = Timer 0 Input signal is
GPIO pin
1 = Timer 0 Input signal is
Timer 1 out
Reserved

Timer 0 Control 1

T0CTL1 (F07H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

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
Prescale Value
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
Timer Input/Output Polarity
Operation of this bit is a
function of
the current operating mode
of the timer
Timer Enable
0 = Timer is disabled
1 = Timer is enabled

Timer 1 High Byte

T1H (F08H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

Timer 1 current count value

Timer 1 Low Byte

T1L (F09H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

Timer 1 current count value

Timer 1 Reload High Byte

T1RH (F0AH - Read/Write)

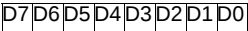
D7 D6 D5 D4 D3 D2 D1 D0

Timer 1 reload value [15:8]



Timer 1 Reload Low Byte

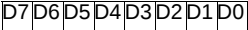
T1RL (F0BH - Read/Write)



Timer 1 reload value [7:0]

Timer 1 PWM High Byte

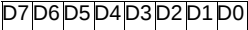
T1PWMH (F0CH - Read/Write)



Timer 1 PWM value [15:8]

Timer 1 PWM Low Byte

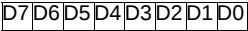
T1PWML (F0DH - Read/Write)



Timer 1 PWM value [7:0]

Timer 1 Control 0

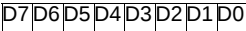
T1CTL0 (F0EH - Read/Write)



Reserved
Cascade Timer
0 = Timer 1 Input signal is
GPIO pin
1 = Timer 1 Input signal is
Timer 0 out
Reserved

Timer 1 Control 1

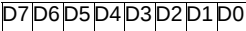
T1CTL1 (F0FH - Read/Write)



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
Prescale Value
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
Timer Input/Output Polarity
Operation of this bit is a
function of
the current operating mode
of the timer
Timer Enable
0 = Timer is disabled
1 = Timer is enabled

UART0 Transmit Data

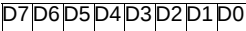
U0TXD (F40H - Write Only)



UART0 transmitter data byte

UART0 Receive Data

U0RXD (F40H - Read Only)

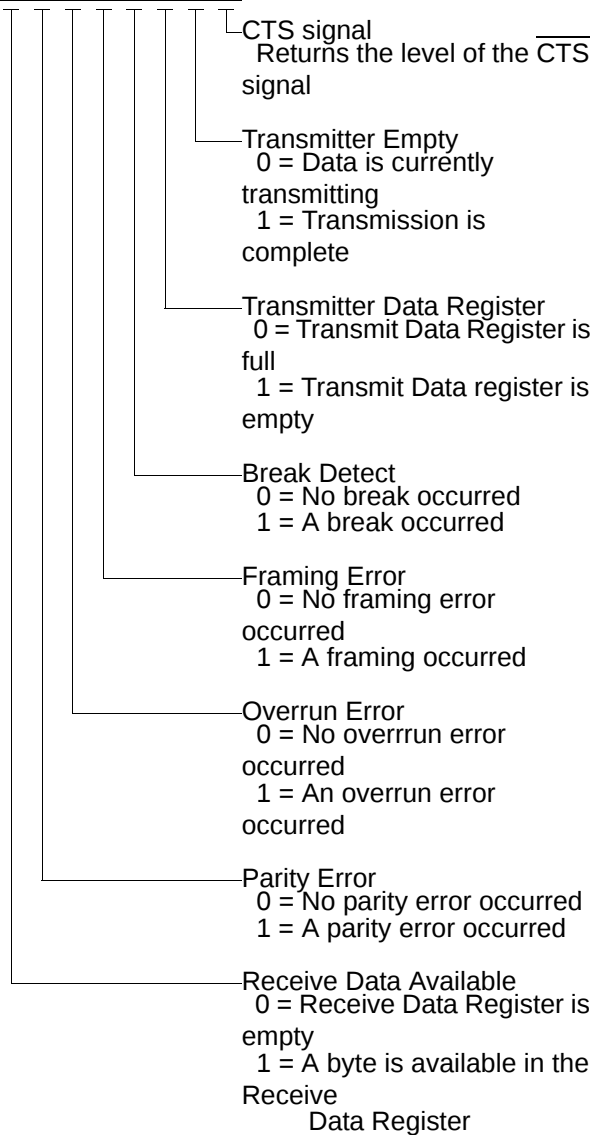


UART0 receiver data byte

UART0 Status 0

U0STAT0 (F41H - Read Only)

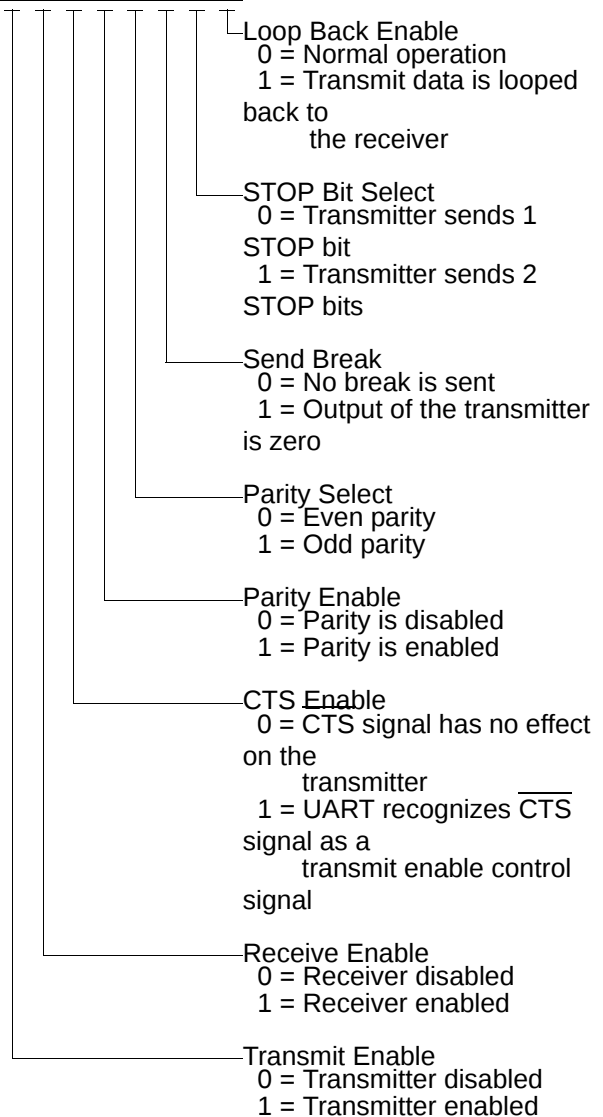
D7 D6 D5 D4 D3 D2 D1 D0



UART0 Control 0

U0CTL0 (F42H - Read/Write)

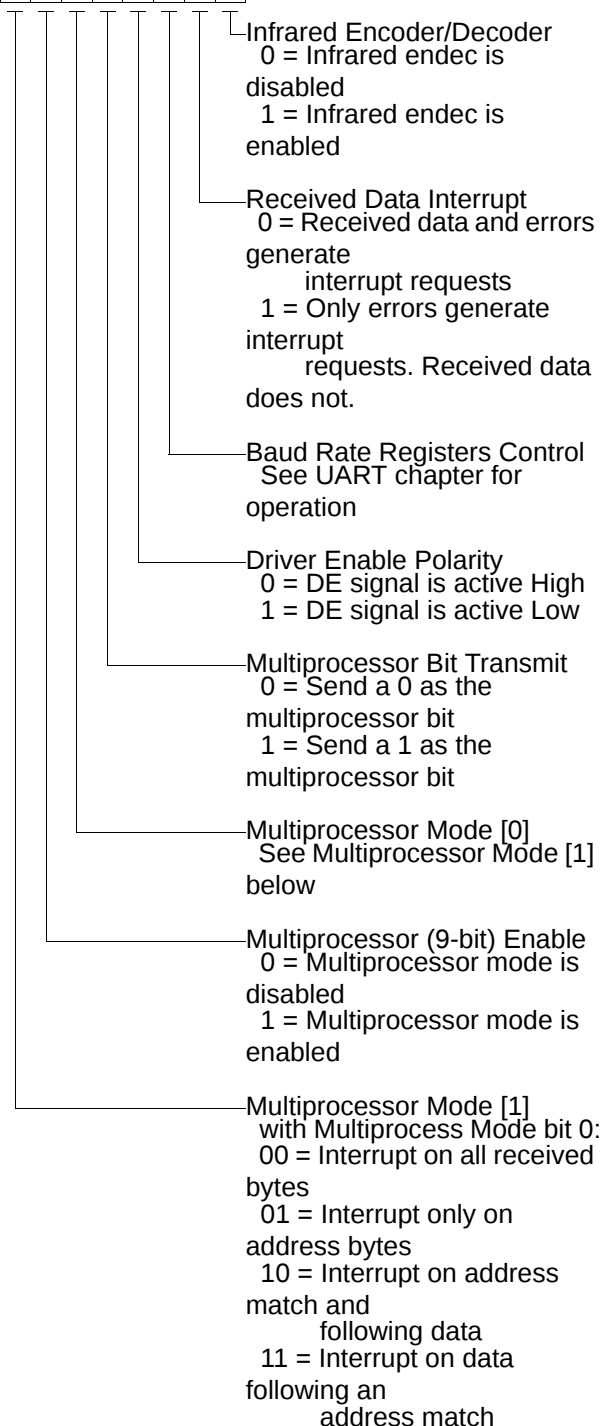
D7 D6 D5 D4 D3 D2 D1 D0



UART0 Control 1

U0CTL1 (F43H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0



UART0 Status 1

U0STAT1 (F44H - Read Only)

D7 D6 D5 D4 D3 D2 D1 D0

- └─ Multitprocessor Receive
Returns value of last
multitprocessor bit
- └─ New Frame
0 = Current byte is not start
of frame
1 = Current byte is start of
new frame
- └─ Reserved

UART0 Address Compare

U0ADDR (F45H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

└─ UART0 Address Compare

UART0 Baud Rate Generator High Byte

U0BRH (F46H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

└─ UART0 Baud Rate divisor

UART0 Baud Rate Generator Low Byte

U0BRL (F47H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

└─ UART0 Baud Rate divisor

I2C Data

I2CDATA (F50H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

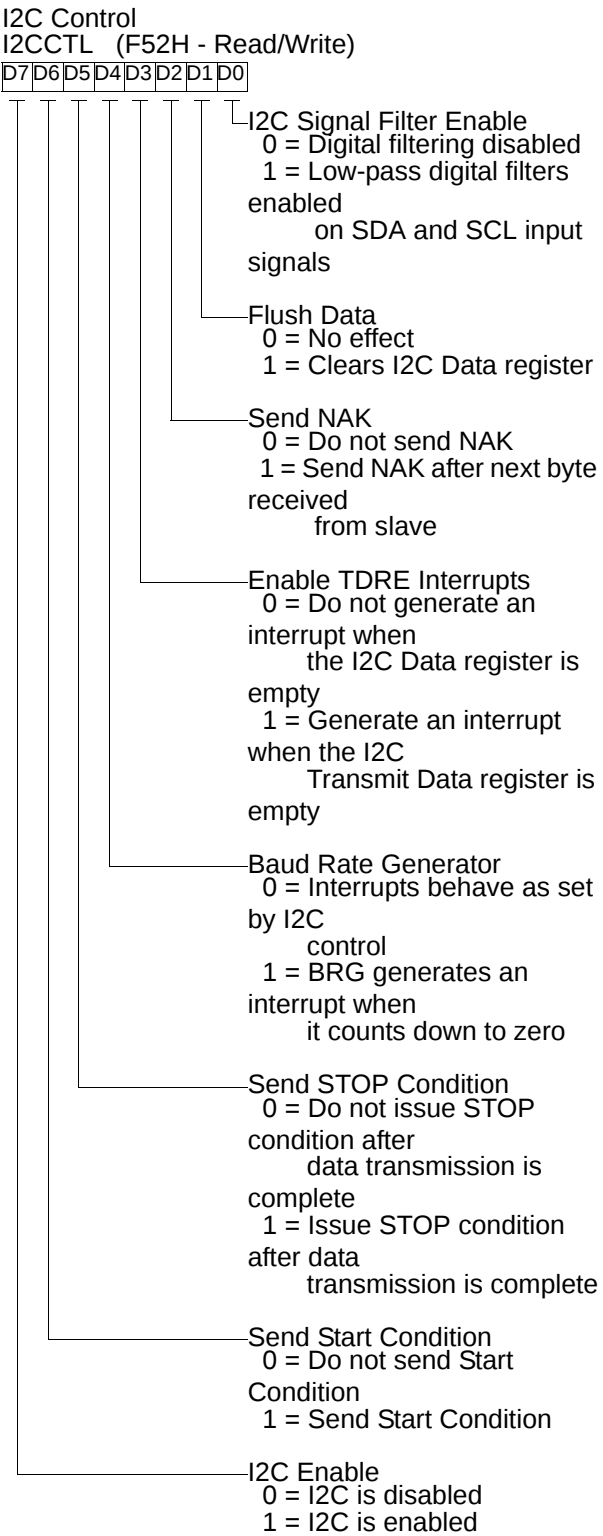
└─ I2C data [7:0]

I2C Status

I2CSTAT (F51H - Read Only)

D7 D6 D5 D4 D3 D2 D1 D0

- └─ NACK Interrupt
0 = No action required to
service NAK
1 = START/STOP not set
after NAK
- └─ Data Shift State
0 = Data is not being
transferred
1 = Data is being transferred
- └─ Transmit Address State
0 = Address is not being
transferred
1 = Address is being
transferred
- └─ Read
0 = Write operation
1 = Read operation
- └─ 10-Bit Address
0 = 7-bit address being
transmitted
1 = 10-bit address being
transmitted
- └─ Acknowledge
0 = Acknowledge not
transmitted/received
1 = For last byte,
Acknowledge was
transmitted/received
- └─ Receive Data Register Full
0 = I2C has not received
data
1 = Data register contains
received data
- └─ Transmit Data Register Empty
0 = Data register is full
1 = Data register is empty



I2C Baud Rate Generator High Byte

I2CBRH (F53H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

I2C Baud Rate divisor [15:8]

I2C Baud Rate Generator Low Byte

I2CBRL (F54H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

I2C Baud Rate divisor [7:0]

SPI Data

SPIDATA (F60H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

SPI Data [7:0]

SPI Control

SPICTL (F61H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

SPI Enable

0 = SPI disabled
1 = SPI enabled

Master Mode Enabled

0 = SPI configured in Slave mode
1 = SPI configured in Master mode

Wire-OR (open-drain) Mode

0 = SPI signals not configured for open-drain
1 = SPI signals (SCK, $\overline{SS}$, MISO, and MOSI) configured for open-drain

Clock Polarity

0 = SCK idles Low
1 = SPI idles High

Phase Select

Sets the phase relationship of the data to the clock.

BRG Timer Interrupt Request

0 = BRG timer function is disabled
1 = BRG time-out interrupt is enabled

Start an SPI Interrupt Request

0 = No effect
1 = Generate an SPI interrupt request

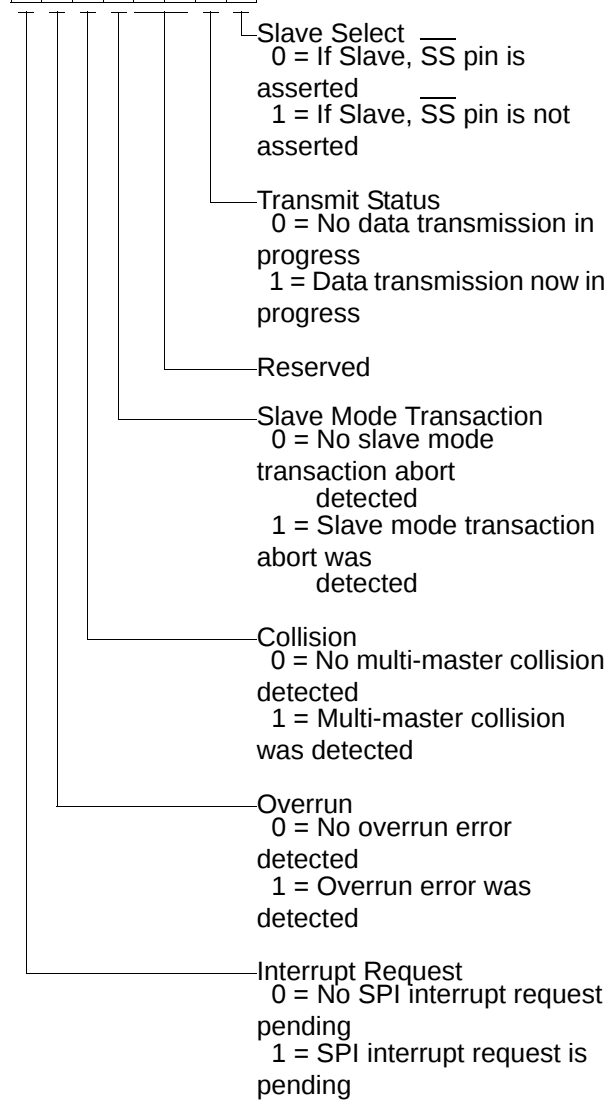
Interrupt Request Enable

0 = SPI interrupt requests are disabled
1 = SPI interrupt requests are enabled

SPI Status

SPISAT (F62H - Read Only)

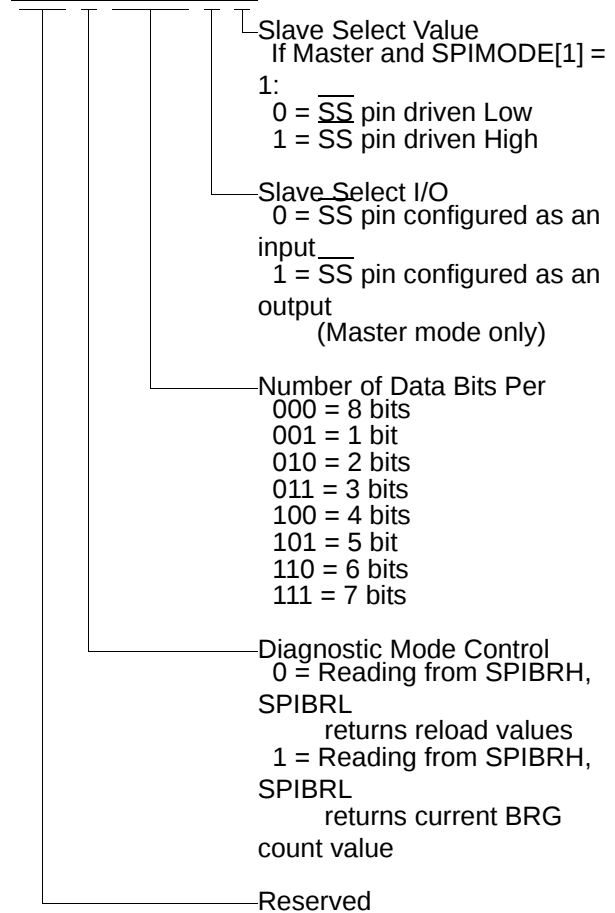
D7 D6 D5 D4 D3 D2 D1 D0

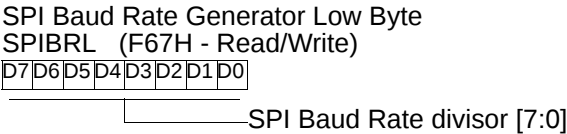
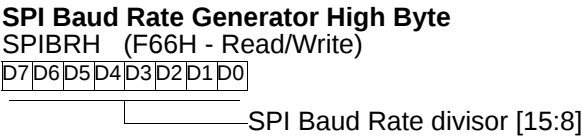
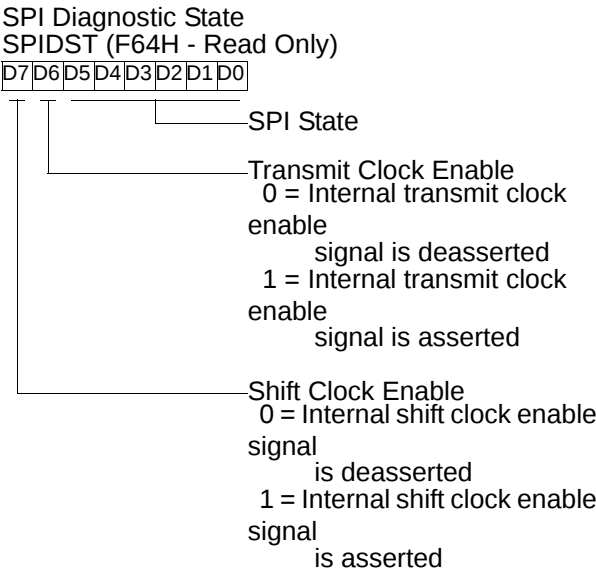


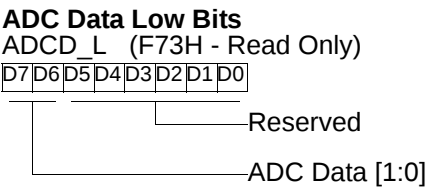
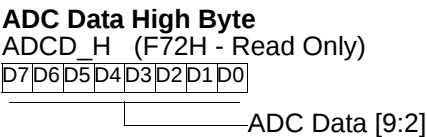
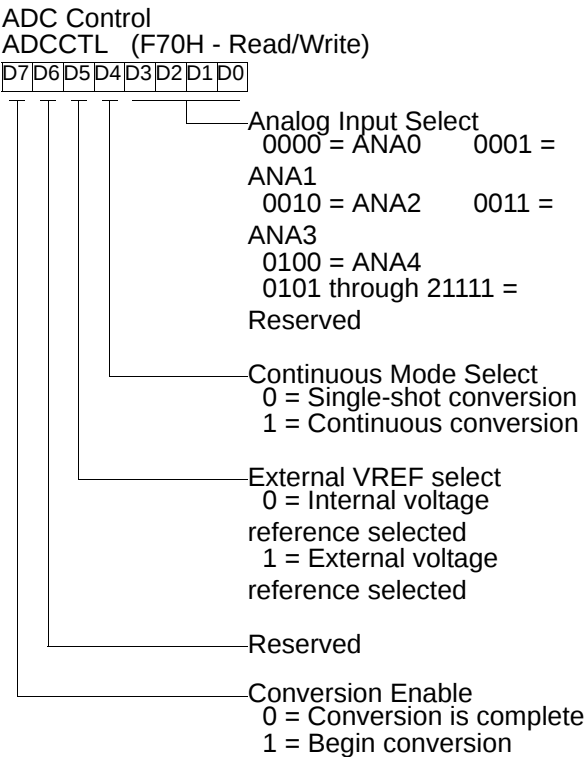
SPI Mode

SPIMODE (F63H - Read/Write)

D7 D6 D5 D4 D3 D2 D1 D0

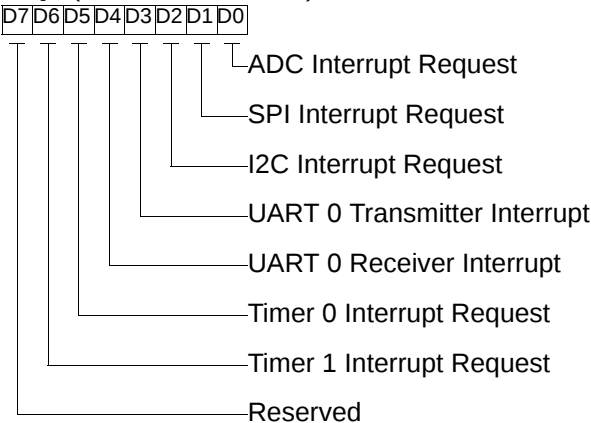






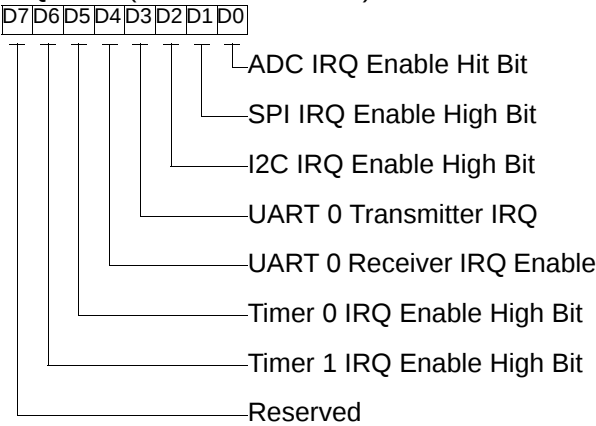


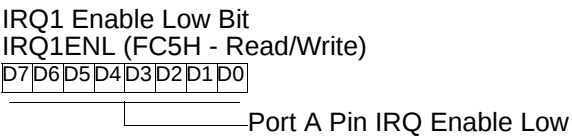
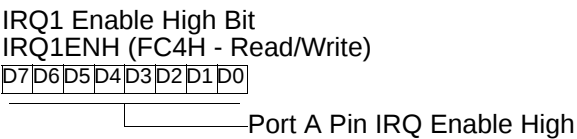
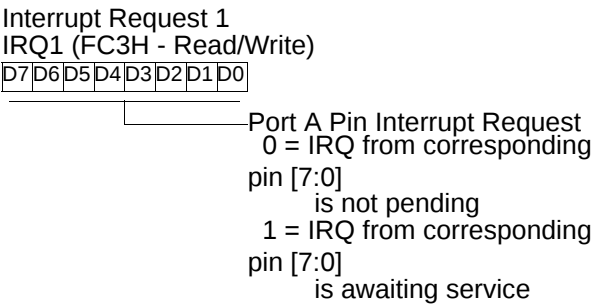
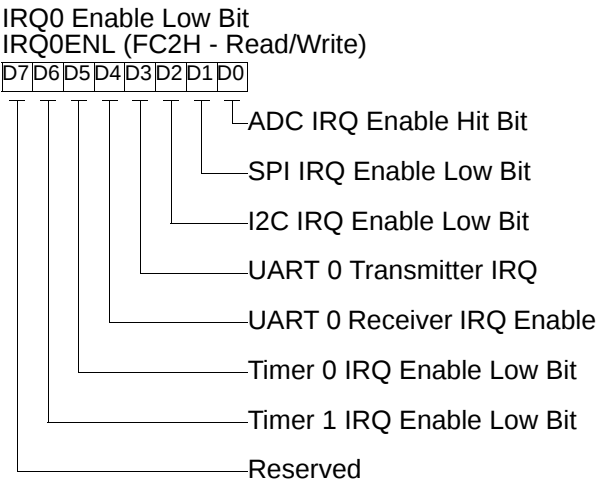
Interrupt Request 0
IRQ0 (FC0H - Read/Write)

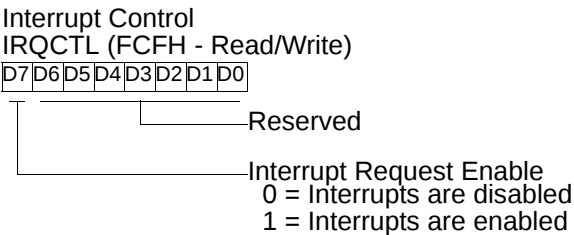
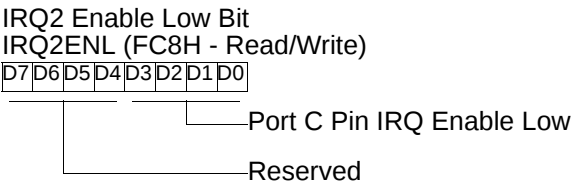
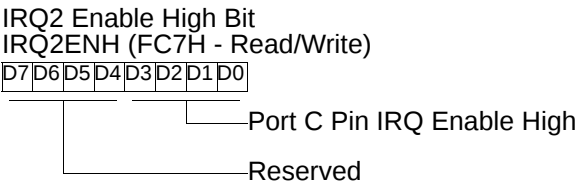
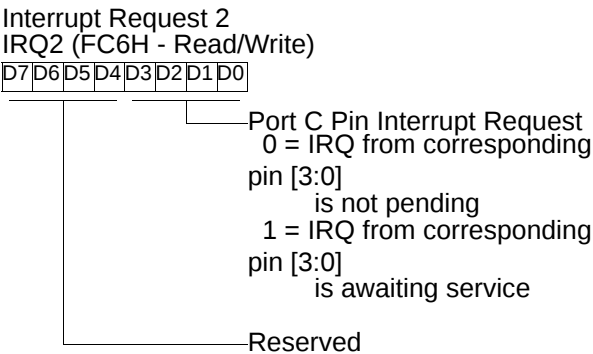


For all of the above
peripherals:
0 = Peripheral IRQ is not
pending
1 = Peripheral IRQ is
awaiting service

IRQ0 Enable High Bit
IRQ0ENH (FC1H - Read/Write)







Port A Address

PAADDR (FD0H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

Port A Address[7:0]
Selects Port Sub-Registers:
00H = No function
01H = Data direction
02H = Alternate function
03H = Output control (open-drain)
04H = High drive enable
05H = STOP mode recovery enable
06H = Pull-up enable
07H-FFH = No function

Port A Control

PACTL (FD1H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

Port A Control[7:0]
Provides Access to Port Sub-Registers

Port A Input Data

PAIN (FD2H - Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

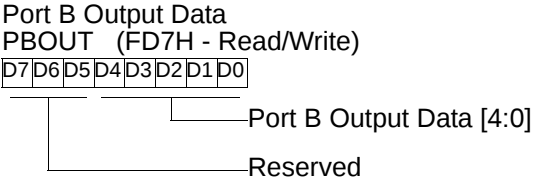
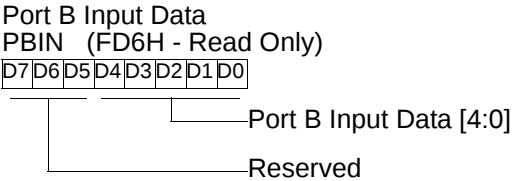
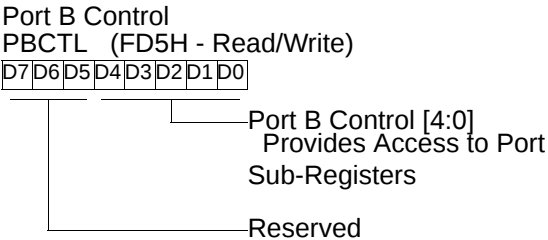
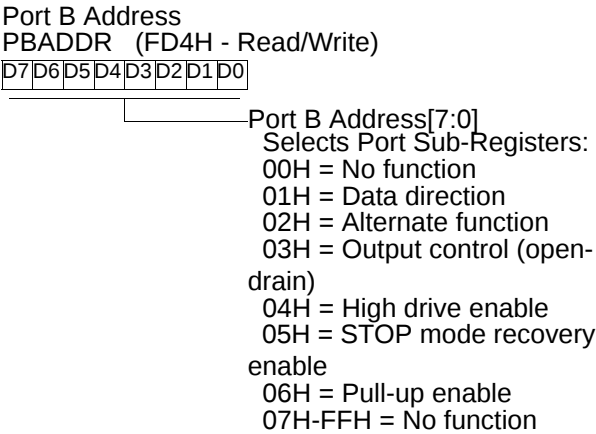
Port A Input Data [7:0]

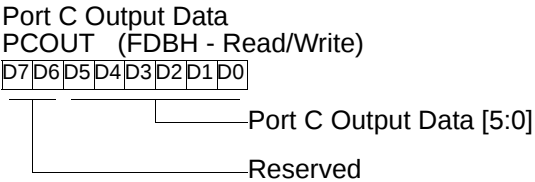
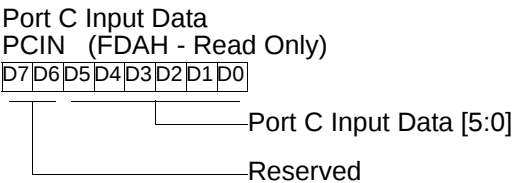
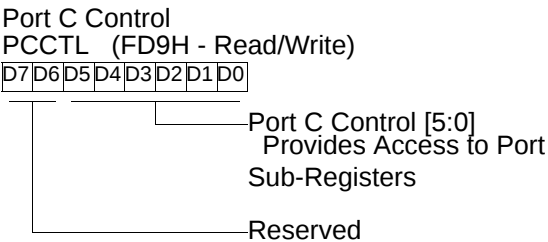
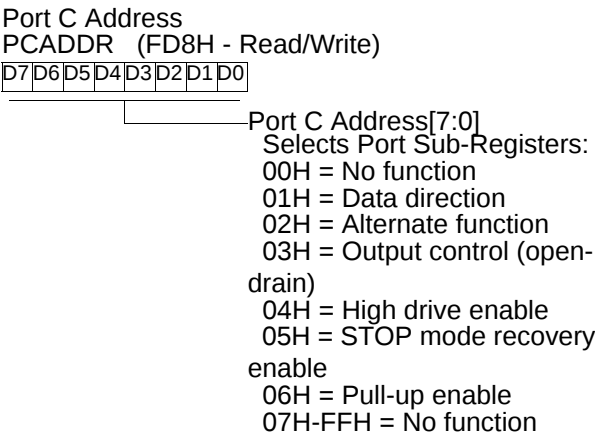
Port A Output Data

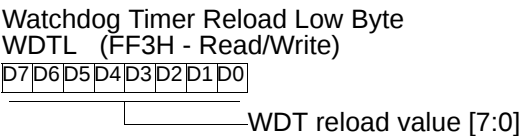
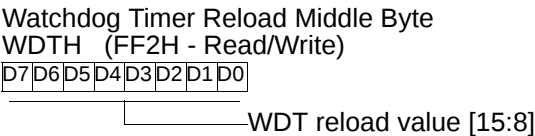
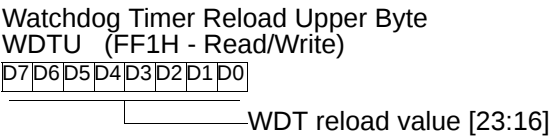
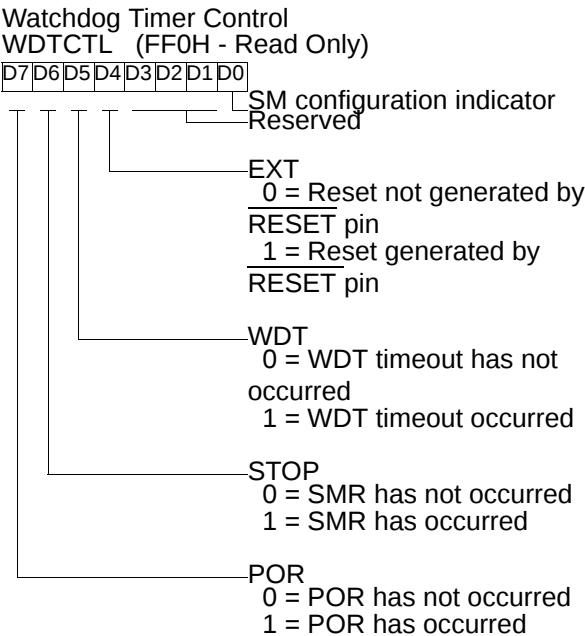
PAOUT (FD3H - Read/Write)

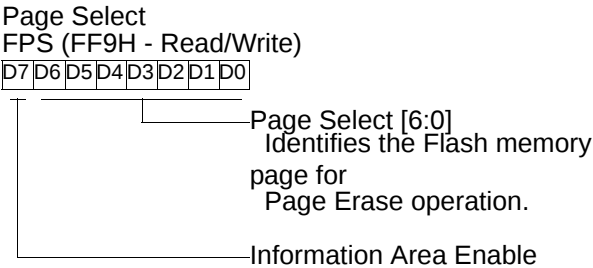
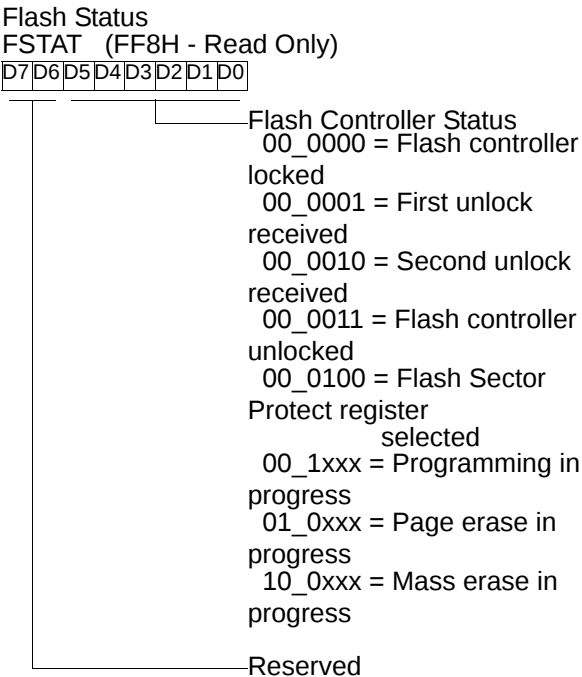
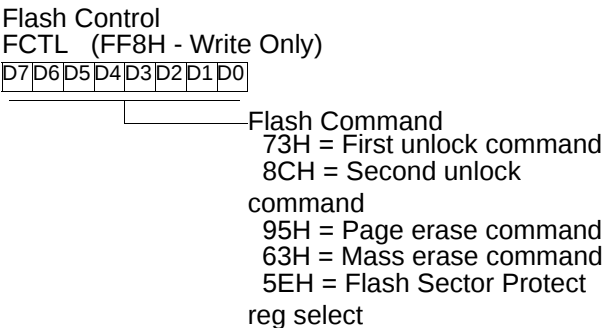
D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

Port A Output Data [7:0]



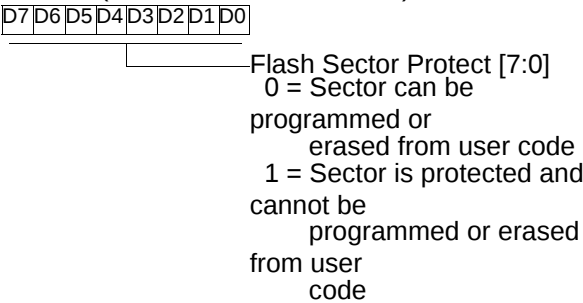








Flash Sector Protect
FPROT (FF9H - Read/Write to 1's)



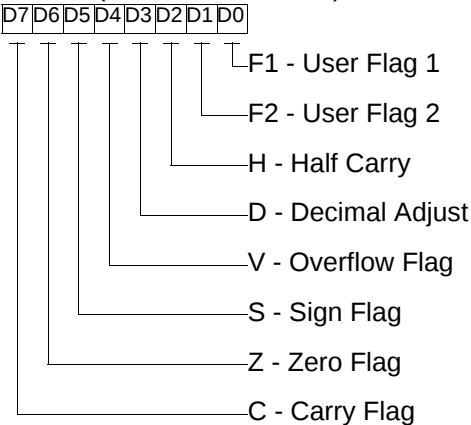
Flash Frequency High Byte
FFREQH (FFAH - Read/Write)



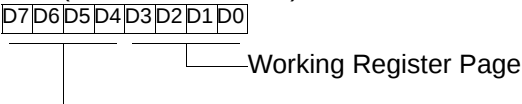
Flash Frequency Low Byte
FFREQL (FFBH - Read/Write)

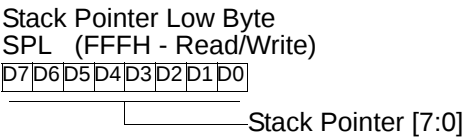
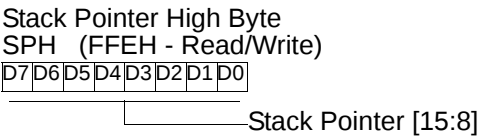
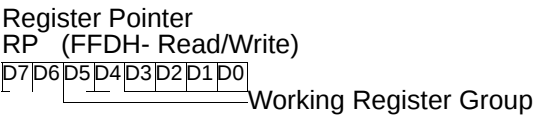


Flags
FLAGS (FFCH - Read/Write)



Register Pointer
RP (FFDH- Read/Write)





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 Brownout
- 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	Reset Latency (Delay)
		CPU	
System Reset	Reset (as applicable)	Reset	66 WDT Oscillator cycles + 16 System Clock cycles
Stop Mode Recovery	Unaffected, except 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 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 text following provides more detailed information on the individual reset sources.

► **Note:** A POR/VBO event always has priority over all other possible reset sources to insure 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 [Electrical Characteristics](#) for POR threshold voltage (V_{POR}).

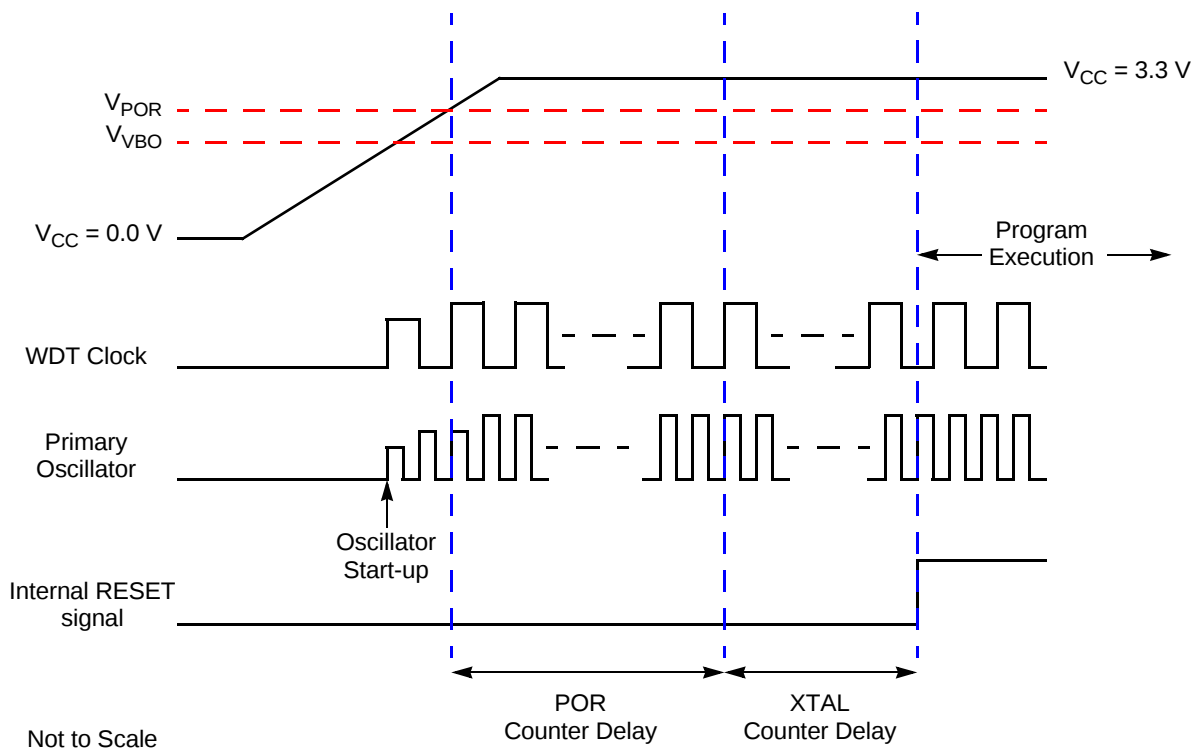


Figure 6. Power-On Reset Operation

Voltage Brownout Reset

The devices in Z8 Encore! XP F0822 Series provide low Voltage Brownout 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 [Electrical Characteristics](#) on page 185 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 on configuring VBO_AO, see [Option Bits](#) on page 163.

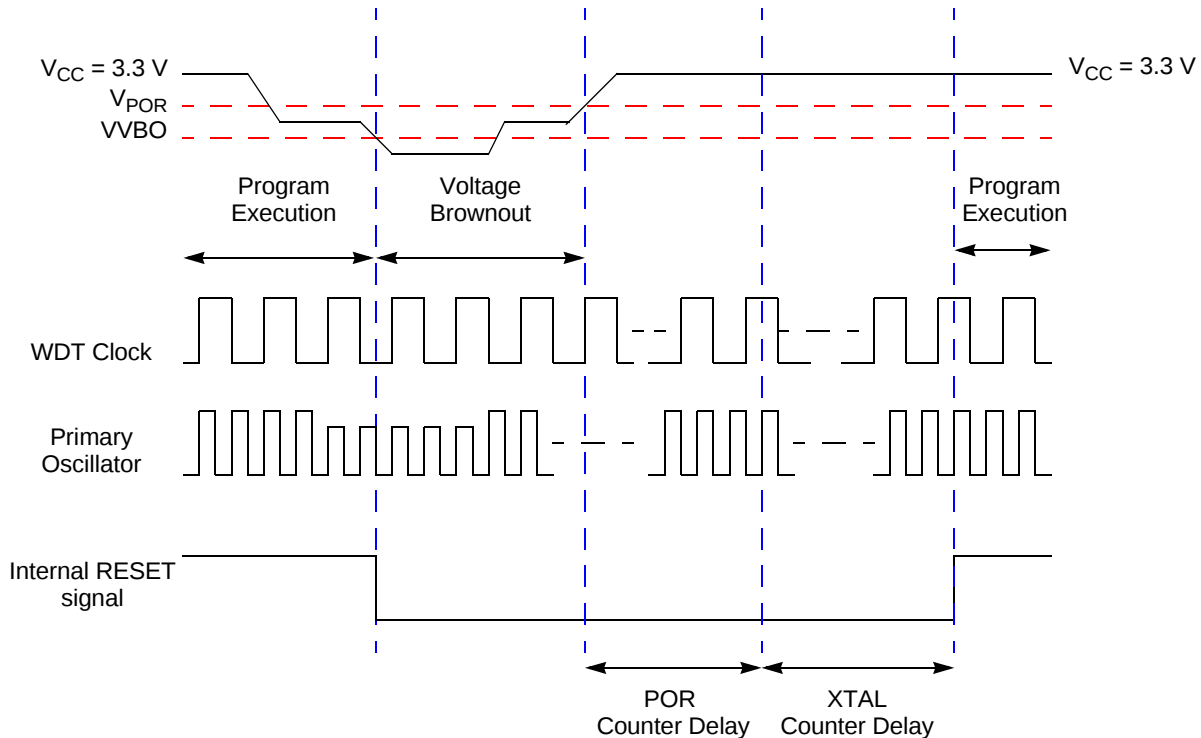


Figure 7. Voltage Brownout Reset Operation

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 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 on STOP mode, see [Low-Power Modes](#) on page 45. 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 on 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 10 ns (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 XIN pin is driven High and the XOUT 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 on Stop Mode Recovery, see [Reset and Stop Mode Recovery](#) on page 39.



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 Brownout 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 pins (Ports A–C) 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 sub-registers 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.

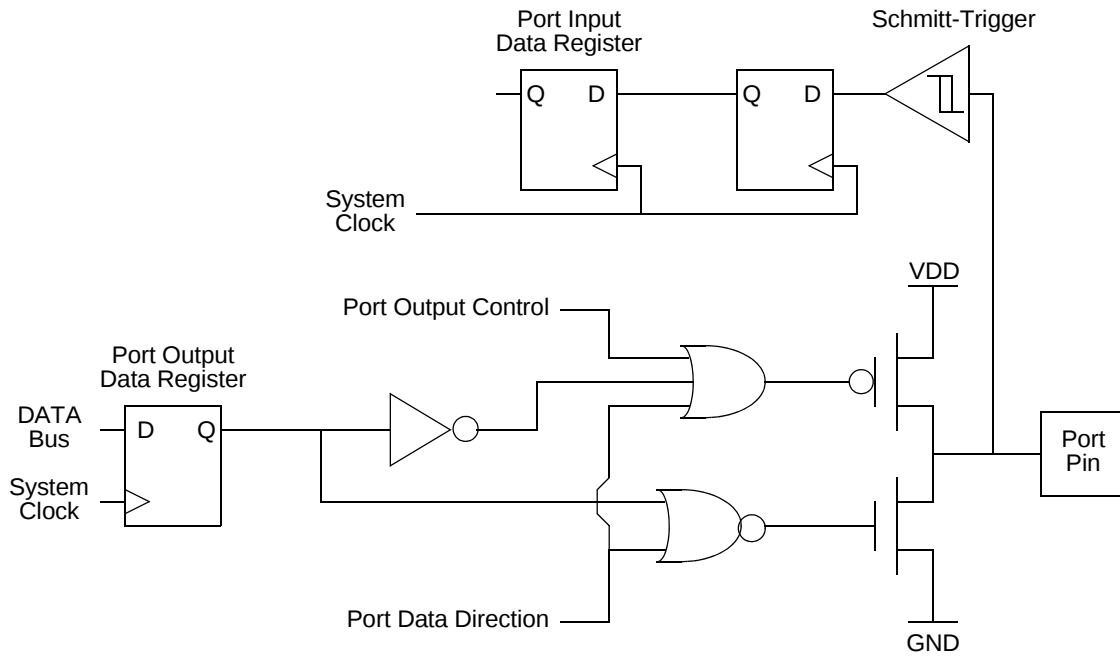


Figure 8. GPIO Port Pin Block Diagram

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	CTS0	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 ² C Clock (automatically open-drain)
	PA7	SDA	I ² C 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 on interrupts using the GPIO pins, see [GPIO Port Pin Block Diagram](#) on page 48.

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 Sub-Registers. Use the Port A–C Address and Control Registers together to provide access to sub-registers for Port configuration and control.

Table 13. GPIO Port Registers and Sub-Registers

Port Register Mnemonic	Port Register Name
PxADDR	Port A–C Address Register (selects sub-registers)
PxCTL	Port A–C Control Register (provides access to sub-registers)
PxIN	Port A–C Input Data Register
PxOUT	Port A–C Output Data Register
Port Sub-Register Mnemonic	Port Register Name
PxDD	Data Direction
PxAF	Alternate Function

Table 13. GPIO Port Registers and Sub-Registers (Continued)

Port Register Mnemonic	Port Register Name
PxOC	Output Control (Open-Drain)
PxHDE	High Drive Enable
PxSMRE	Stop Mode Recovery Source Enable
PxPUE	Pull-up Enable

Port A–C Address Registers

The Port A–C Address Registers select the GPIO Port functionality accessible through the Port A–C Control Registers. The Port A–C Address and Control Registers combine to provide access to all GPIO Port control ([Table 14](#)).

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

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

PADDR[7:0]—Port Address

The Port Address selects one of the sub-registers accessible through the Port Control register.

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

Port A–C Control Registers

The Port A–C Control Registers set the GPIO port operation. The value in the corresponding Port A–C Address Register determines the control sub-registers accessible using the Port A–C Control Register (Table 15).

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

BITS	7	6	5	4	3	2	1	0
FIELD	PCTL							
RESET	00H							
R/W	R/W							
ADDR	FD1H, FD5H, FD9H							

PCTL[7:0]—Port Control

The Port Control Register provides access to all sub-registers that configure the GPIO Port operation.

Port A–C Data Direction Sub-Registers

The Port A–C Data Direction sub-register is accessed through the Port A–C Control register by writing 01H to the Port A–C Address Register (Table 16).

Table 16. Port A–C Data Direction Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
RESET	1							
R/W	R/W							
ADDR	If 01H in Port A–C Address Register, accessible through the Port A–C Control Register							

DD[7:0]—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.

Port A–C Alternate Function Sub-Registers

The Port A–C Alternate Function sub-register (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 sub-registers select the alternate functions for the selected

pins. To determine the alternate function associated with each port pin, see [GPIO Port Pin Block Diagram](#) on page 48.



Caution: Do not enable alternate function 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 Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	AF7	AF6	AF5	AF4	AF3	AF2	AF1	AF0
RESET	0							
R/W	R/W							
ADDR	If 02H in Port A–C Address Register, accessible through the Port A–C Control Register							

AF[7:0]—Port Alternate Function enabled

0 = The port pin is in NORMAL mode and the DDx bit in the Port A–C Data Direction sub-register determines the direction of the pin.

1 = The alternate function is selected. Port pin operation is controlled by the alternate function.

Port A–C Output Control Sub-Registers

The Port A–C Output Control sub-register ([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 sub-registers to 1 configures the specified port pins for open-drain operation. These sub-registers affect the pins directly and, as a result, alternate functions are also affected.

Table 18. Port A–C Output Control Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	POC7	POC6	POC5	POC4	POC3	POC2	POC1	POC0
RESET	0							
R/W	R/W							
ADDR	If 03H in Port A–C Address Register, accessible through the Port A–C Control Register							

POC[7:0]—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).

Port A–C High Drive Enable Sub-Registers

The Port A–C High Drive Enable sub-register (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 sub-registers to 1 configures the specified port pins for high current output drive operation. The Port A–C High Drive Enable sub-register affects the pins directly and, as a result, alternate functions are also affected.

Table 19. Port A–C High Drive Enable Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	PHDE7	PHDE6	PHDE5	PHDE4	PHDE3	PHDE2	PHDE1	PHDE0
RESET	0							
R/W	R/W							
ADDR	If 04H in Port A–C Address Register, accessible through the Port A–C Control Register							

PHDE[7:0]—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.

Port A–C Stop Mode Recovery Source Enable Sub-Registers

The Port A–C Stop Mode Recovery Source Enable sub-register (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 sub-registers 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 Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	PSMRE7	PSMRE6	PSMRE5	PSMRE4	PSMRE3	PSMRE2	PSMRE1	PSMRE0
RESET	0							
R/W	R/W							
ADDR	If 05H in Port A–C Address Register, accessible through the Port A–C Control Register							

PSMRE[7:0]—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.

Port A–C Pull-up Enable Sub-Registers

The Port A–C Pull-Up Enable sub-register ([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 sub-registers enables a weak internal resistive pull-up on the specified Port pins.

Table 21. Port A–C Pull-Up Enable Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	PPUE7	PPUE6	PPUE5	PPUE4	PPUE3	PPUE2	PPUE1	PPUE0
RESET	0							
R/W	R/W							
ADDR	If 06H in Port A–C Address Register, accessible through the Port A–C Control Register							

PPUE[7:0]—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.

Port A–C Input Data Registers

Reading from the Port A–C Input Data Registers ([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)

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

PIN[7:0]—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).

Port A–C Output Data Register

The Port A–C Output Data Register ([Table 23](#)) controls the output data to the pins.

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

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

POUT[7:0]—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). High value is not driven if the drain has been disabled by setting the corresponding Port Output Control Register bit to 1.



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 on interrupt servicing, refer to *eZ8 CPU Core User Manual (UM0128)* available for download at www.zilog.com.

Interrupt Vector Listing

Table 24 lists all 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 Watchdog Timer on page 83)
	0006H	Illegal Instruction Trap (not an interrupt)

Table 24. Interrupt Vectors in Order of Priority (Continued)

Priority	Program Memory Vector Address	Interrupt Source
	0008H	Reserved
	000AH	Timer 1
	000CH	Timer 0
	000EH	UART 0 receiver
	0010H	UART 0 transmitter
	0012H	I ² C
	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.

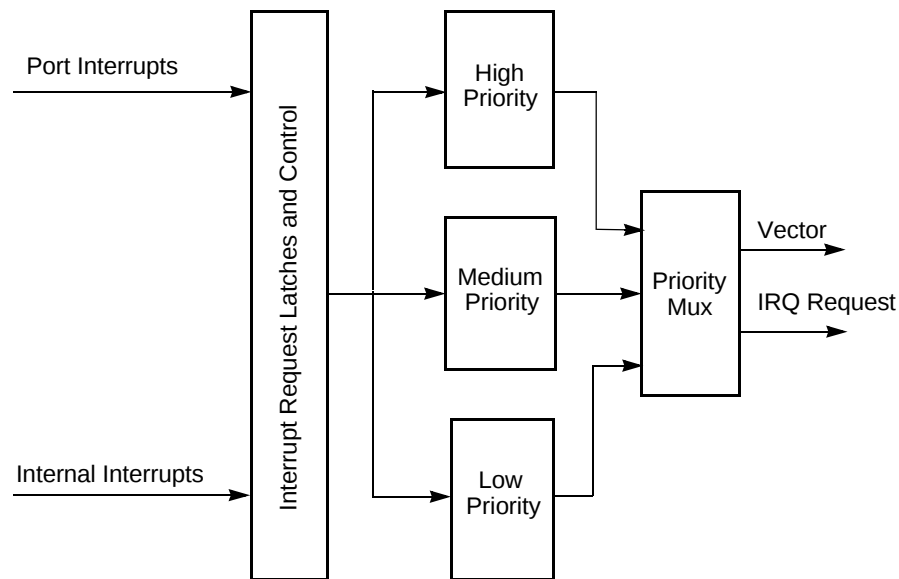


Figure 9. Interrupt Controller Block Diagram

Operation

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 EI (Enable Interrupt) instruction.
- Execution of an IRET (Return from Interrupt) 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 DI (Disable Interrupt) 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 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](#). Reset, WDT interrupt (if enabled), and Illegal Instruction Trap always have 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: *The following style of coding to clear bits in the Interrupt Request Registers is not recommended. All incoming interrupts received between execution of the first LDX command and the last LDX command is lost.*

Poor coding style resulting in lost interrupt requests:

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



Note: *To avoid missing interrupts, the following style of coding to clear bits in the Interrupt Request 0 register is recommended:*

Good coding style that avoids lost interrupt requests:

```
ANDX IRQ0, MASK
```

Software Interrupt Assertion

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



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

Poor coding style that resulting in lost interrupt requests:

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

► **Note:** To avoid missing interrupts, the following style of coding to set bits in the Interrupt Request Registers is recommended

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 (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)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	T1I	T0I	U0RXI	U0TXI	I2CI	SPII	ADCI
RESET	0							
R/W	R/W							
ADDR	FC0H							

Reserved—Must be 0

T1I—Timer 1 Interrupt Request

0 = No interrupt request is pending for Timer 1.

1 = An interrupt request from Timer 1 is awaiting service.

T0I—Timer 0 Interrupt Request

0 = No interrupt request is pending for Timer 0.

1 = An interrupt request from Timer 0 is awaiting service.

U0RXI—UART 0 Receiver Interrupt Request

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

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

U0TXI—UART 0 Transmitter Interrupt Request

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

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

I2CI—I²C Interrupt Request

0 = No interrupt request is pending for the I²C.

1 = An interrupt request from the I²C is awaiting service.

SPII—SPI Interrupt Request

0 = No interrupt request is pending for the SPI.

1 = An interrupt request from the SPI is awaiting service.

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 (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)

BITS	7	6	5	4	3	2	1	0
FIELD	PA7I	PA6I	PA5I	PA4I	PA3I	PA2I	PA1I	PA0I
RESET	0							
R/W	R/W							
ADDR	FC3H							

PAxI—Port A Pin x Interrupt Request

0 = No interrupt request is pending for GPIO Port A pin x.

1 = An interrupt request from GPIO Port A pin x is awaiting service.

Where x indicates the specific GPIO Port pin number (0 through 7).

Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register (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)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				PC3I	PC2I	PC1I	PC0I
RESET	0							
R/W	R/W							
ADDR	FC6H							

Reserved—Must be 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.

Where x indicates the specific GPIO Port C pin number (0 through 3).

IRQ0 Enable High and Low Bit Registers

The IRQ0 Enable High and Low Bit Registers (Table 29 and Table 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 describes the priority control for IRQ0.

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

where x indicates the register bits from 0 through 7.

Table 29. IRQ0 Enable High Bit Register (IRQ0ENH)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	T1ENH	T0ENH	U0RENH	U0TENH	I2CENH	SPIENH	ADCENH
RESET	0							
R/W	R/W							
ADDR	FC1H							

Reserved—Must be 0

T1ENH—Timer 1 Interrupt Request Enable High Bit

T0ENH—Timer 0 Interrupt Request Enable High Bit

U0RENH—UART 0 Receive Interrupt Request Enable High Bit

U0TENH—UART 0 Transmit Interrupt Request Enable High Bit

I2CENH—I²C Interrupt Request Enable High Bit

SPIENH—SPI Interrupt Request Enable High Bit

ADCENH—ADC Interrupt Request Enable High Bit

Table 30. IRQ0 Enable Low Bit Register (IRQ0ENL)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	T1ENL	T0ENL	U0RENL	U0TENL	I2CENL	SPIENL	ADCENL
RESET	0							
R/W	R/W							
ADDR	FC2H							

Reserved—Must be 0

T1ENL—Timer 1 Interrupt Request Enable Low Bit

T0ENL—Timer 0 Interrupt Request Enable Low Bit

U0RENL—UART 0 Receive Interrupt Request Enable Low Bit

U0TENL—UART 0 Transmit Interrupt Request Enable Low Bit

I2CENL—I²C Interrupt Request Enable Low Bit

SPIENL—SPI Interrupt Request Enable Low Bit

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 (Table 32 and Table 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

where x indicates the register bits from 0 through 7.

Table 32. IRQ1 Enable High Bit Register (IRQ1ENH)

BITS	7	6	5	4	3	2	1	0
FIELD	PA7ENH	PA6ENH	PA5ENH	PA4ENH	PA3ENH	PA2ENH	PA1ENH	PA0ENH
RESET	0							
R/W	R/W							
ADDR	FC4H							

PAxENH—Port A Bit[x] Interrupt Request Enable High Bit

Table 33. IRQ1 Enable Low Bit Register (IRQ1ENL)

BITS	7	6	5	4	3	2	1	0
FIELD	PA7ENL	PA6ENL	PA5ENL	PA4ENL	PA3ENL	PA2ENL	PA1ENL	PA0ENL
RESET	0							
R/W	R/W							
ADDR	FC5H							

PAxENL—Port A Bit[x] Interrupt Request Enable Low Bit

IRQ2 Enable High and Low Bit Registers

[Table 34](#) describes the priority control for IRQ2. The IRQ2 Enable High and Low Bit Registers ([Table 35](#) and [Table 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

where x indicates the register bits from 0 through 7.

Table 35. IRQ2 Enable High Bit Register (IRQ2ENH)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				C3ENH	C2ENH	C1ENH	C0ENH
RESET	0							
R/W	R/W							
ADDR	FC7H							

Reserved—Must be 0.

C3ENH—Port C3 Interrupt Request Enable High Bit

C2ENH—Port C2 Interrupt Request Enable High Bit

C1ENH—Port C1 Interrupt Request Enable High Bit

C0ENH—Port C0 Interrupt Request Enable High Bit

Table 36. IRQ2 Enable Low Bit Register (IRQ2ENL)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				C3ENL	C2ENL	C1ENL	C0ENL
RESET	0							
R/W	R/W							
ADDR	FC8H							

Reserved—Must be 0.

C3ENL—Port C3 Interrupt Request Enable Low Bit

C2ENL—Port C2 Interrupt Request Enable Low Bit

C1ENL—Port C1 Interrupt Request Enable Low Bit

C0ENL—Port C0 Interrupt Request Enable Low Bit

Interrupt Edge Select Register

The Interrupt Edge Select (IRQES) register (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)

BITS	7	6	5	4	3	2	1	0
FIELD	IES7	IES6	IES5	IES4	IES3	IES2	IES1	IES0
RESET	0							
R/W	R/W							
ADDR	FCDH							

IESx—Interrupt Edge Select x

0 = An interrupt request is generated on the falling edge of the PAX input.

1 = An interrupt request is generated on the rising edge of the PAX input.

Where x indicates the specific GPIO Port pin number (0 through 7).

Interrupt Control Register

The Interrupt Control (IRQCTL) register (Table 38) contains the master enable bit for all interrupts.

Table 38. Interrupt Control Register (IRQCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	IRQE	Reserved						
RESET	0							
R/W	R/W	R						
ADDR	FCFH							

IRQE—Interrupt Request Enable

This bit is set to 1 by execution of an EI (Enable Interrupts) or IRET (Interrupt Return) 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.

Reserved—Must be 0



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 I²C peripherals can also be used to provide basic timing functionality. See the respective serial communication peripheral chapters for information on using the Baud Rate Generators as timers.

Architecture

[Figure 10](#) displays the architecture of the timers.

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.

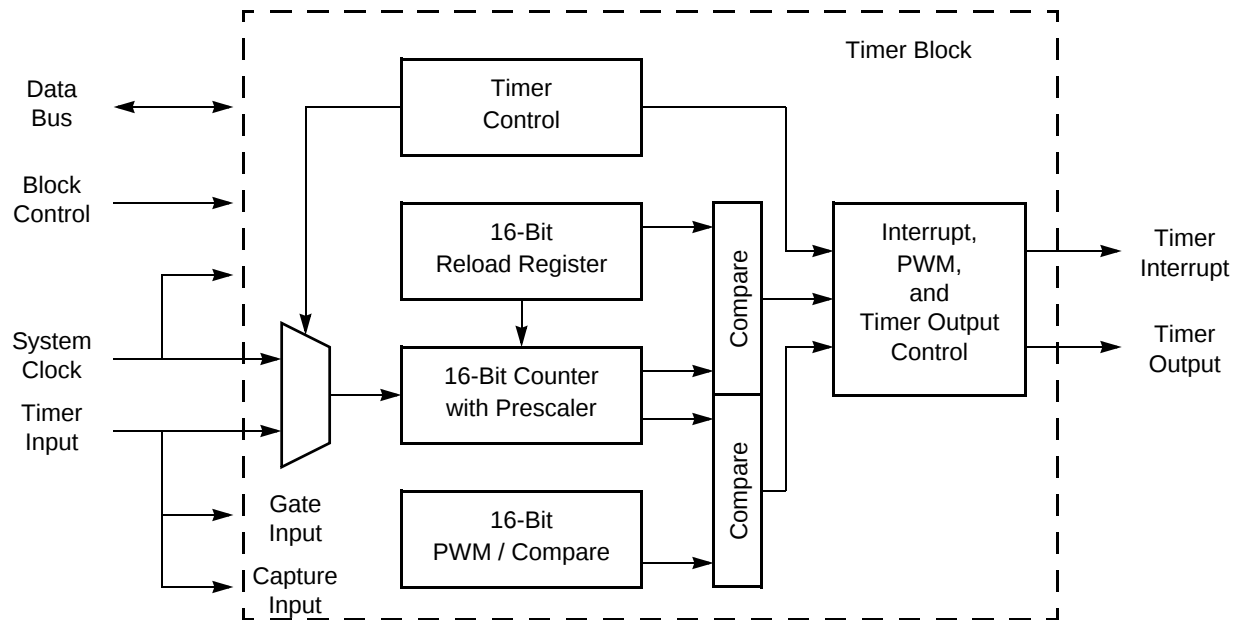


Figure 10. Timer Block Diagram

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. On 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.

Follow the steps below 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 desired, 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 given by 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.

Follow the steps below 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 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 desired, 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 given by 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.

Follow the steps below 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 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 given by the following equation:

$$\text{COUNTER Mode Timer Input Transitions} = \text{Current Count Value} - \text{Start Value}$$

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.

Follow the steps below 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 given by the following equation.

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

If an initial starting value other than 0001H is loaded into the Timer High and Low Byte Registers, the ONE-SHOT mode equation 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 given by

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

If TPOL is set to 1, the ratio of the PWM output High time to the total period is given by

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

CAPTURE Mode

In CAPTURE mode, the current timer count value is recorded when the desired 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.

Follow the steps below 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.

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

1. Write to the Timer Control Register to:
 - Disable the timer
 - Configure the timer for COMPARE mode
 - Set the prescale value
 - Set the initial logic level (High or Low) for the Timer Output alternate function, if 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.

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

1. Write to the Timer Control Register to:
 - Disable the timer
 - Configure the timer for GATED mode
 - Set the prescale value
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 desired, 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 desired transition (after the first) of the Timer Input signal captures the current count value. The Capture value is written to the Timer PWM High and Low Byte Registers. When the Capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte Registers is reset to 0001H and counting resumes.

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.

Follow the steps below 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 desired, 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

Timer 0–1 High and Low Byte Registers

The Timer 0–1 High and Low Byte (TxH and TxL) Registers ([Table 39](#)) 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.

Writing to the Timer High and Low Byte Registers while the timer is enabled is not recommended. 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)

BITS	7	6	5	4	3	2	1	0
FIELD	TH							
RESET	0							
R/W	R/W							
ADDR	F00H, F08H							

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

BITS	7	6	5	4	3	2	1	0
FIELD	TL							
RESET	0							1
R/W	R/W							
ADDR	F01H, F09H							

TH and 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 ([Table 41](#)) 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)

BITS	7	6	5	4	3	2	1	0
FIELD	TRH							
RESET	1							
R/W	R/W							
ADDR	F02H, F0AH							

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

BITS	7	6	5	4	3	2	1	0
FIELD	TRL							
RESET	1							
R/W	R/W							
ADDR	F03H, F0BH							

TRH and 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 ([Table 43](#) and [Table 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)

BITS	7	6	5	4	3	2	1	0
FIELD	PWMH							
RESET	0							
R/W	R/W							
ADDR	F04H, F0CH							

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

BITS	7	6	5	4	3	2	1	0
FIELD	PWML							
RESET	0							
R/W	R/W							
ADDR	F05H, F0DH							

PWMH and 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 Register (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 ([Table 45](#)) allow cascading of the Timers.

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

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved			CSC	Reserved			
RESET	0							
R/W	R/W							
ADDR	F06H, F0EH, F16H, F1EH							

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, input signal is connected to Timer 0 output.

Timer 0–1 Control 1 Registers

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

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

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	TPOL	PRES			TMODE		
RESET	0							
R/W	R/W							
ADDR	F07H, F0FH							

TEN—Timer Enable

0 = Timer is disabled.

1 = Timer enabled to count.

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.

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.

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. This insures 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

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—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 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 given by the following equation:

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

where the WDT reload value is the decimal value of the 24-bit value given by {WDTU[7:0], WDTM[7:0], WDTL[7:0]} and the typical Watchdog Timer RC oscillator frequency is 10 kHz. WDT cannot be refreshed once it reaches 000002H. The WDT Reload Value must not be set to values below 000004H. [Table 47](#) provides information on approximate time-out delays for minimum and maximum WDT reload values.

Table 47. Watchdog Timer Approximate Time-Out Delays

WDT Reload Value	WDT Reload Value	Approximate Time-Out Delay (with 10 kHz typical WDT Oscillator Frequency)	
(Hex)	(Decimal)	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 [Option Bits](#) on page 163.

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 FFFFFFFH 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 [Reset and Stop Mode Recovery](#) on page 39. 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 on Reset, see [Reset and Stop Mode Recovery](#) on page 39.

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 on Reset, see [Reset and Stop Mode Recovery](#) on page 39. 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, WDTM, 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, WDTM, 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 (WDTM).
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

Watchdog Timer Control Register

The Watchdog Timer Control Register (WDTCTL), detailed 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, WDTL, 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)

BITS	7	6	5	4	3	2	1	0
FIELD	POR	STOP	WDT	EXT	Reserved			
RESET	See descriptions below			0				
R/W	R							
ADDR	FF0H							

Reset or Stop Mode Recovery Event	POR	STOP	WDT	EXT
Power-On Reset	1	0	0	0
Reset through 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

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.

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.

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.

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.

Reserved

These bits are reserved and must be 0.

Watchdog Timer Reload Upper, High and Low Byte Registers

The Watchdog Timer Reload Upper, High and Low Byte (WDTU, WDTL, WDTL) Registers (Table 49 through Table 51) 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], WDTL[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 49. Watchdog Timer Reload Upper Byte Register (WDTU)

BITS	7	6	5	4	3	2	1	0
FIELD	WDTU							
RESET	1							
R/W	R/W*							
ADDR	FF1H							
R/W*—Read returns the current WDT count value. Write sets the desired Reload Value.								

WDTU—WDT Reload Upper Byte

Most significant byte (MSB), Bits[23:16], of the 24-bit WDT reload value.

Table 50. Watchdog Timer Reload High Byte Register (WDTH)

BITS	7	6	5	4	3	2	1	0
FIELD	WDTH							
RESET	1							
R/W	R/W*							
ADDR	FF2H							
R/W*—Read returns the current WDT count value. Write sets the desired Reload Value.								

WDTH—WDT Reload High Byte

Middle byte, Bits[15:8], of the 24-bit WDT reload value.

Table 51. Watchdog Timer Reload Low Byte Register (WDTL)

BITS	7	6	5	4	3	2	1	0
FIELD	WDTL							
RESET	1							
R/W	R/W*							
ADDR	FF3H							
R/W*—Read returns the current WDT count value. Write sets the desired Reload Value.								

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. [Figure11](#) on page 90 displays the UART architecture.

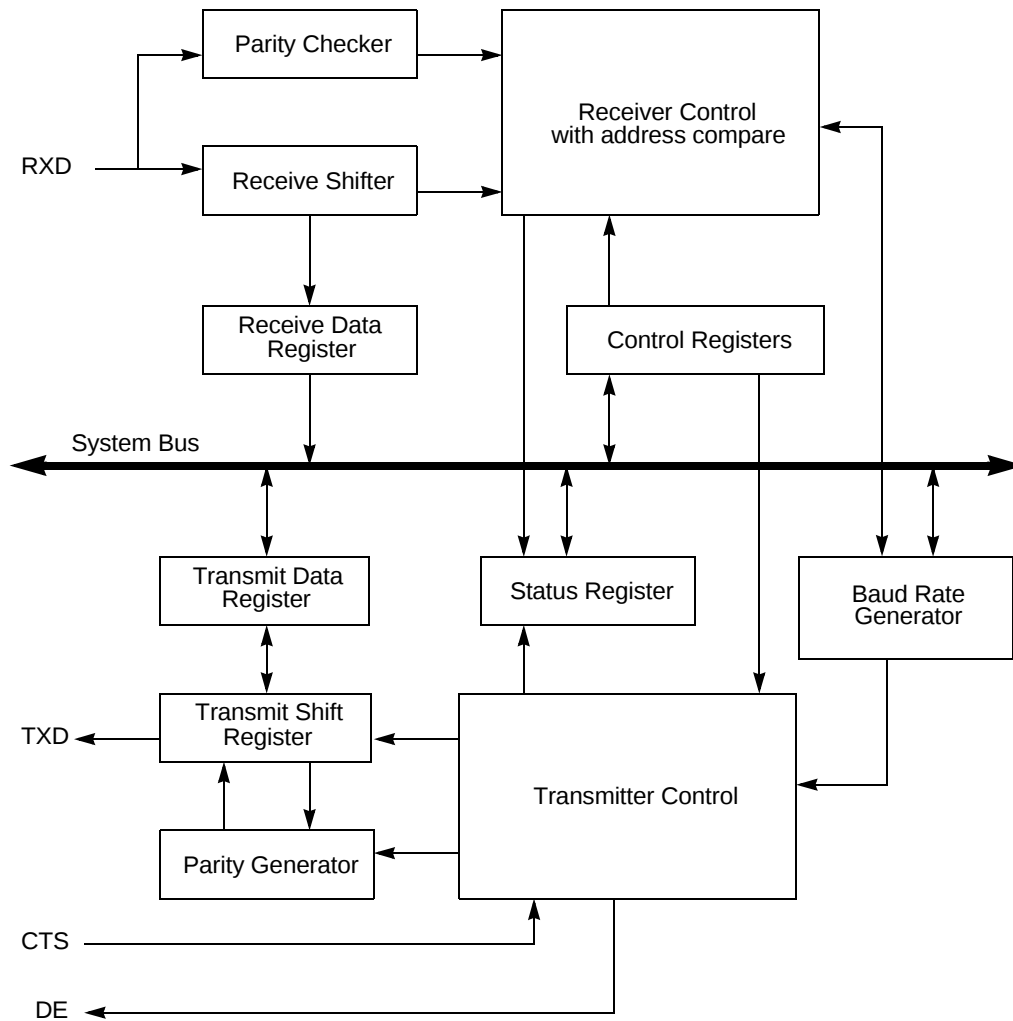


Figure 11. UART Block Diagram

Operation

Data Format

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. [Figure12](#) on page 91 and [Figure13](#) on page 91 display the asynchronous data format used by the UART without parity and with parity, respectively.

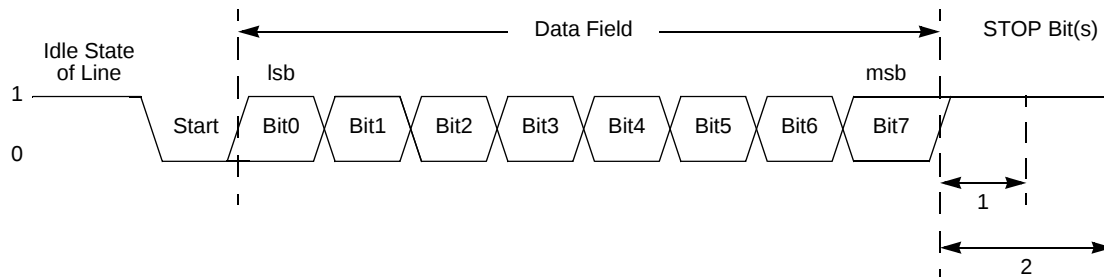


Figure 12. UART Asynchronous Data Format without Parity

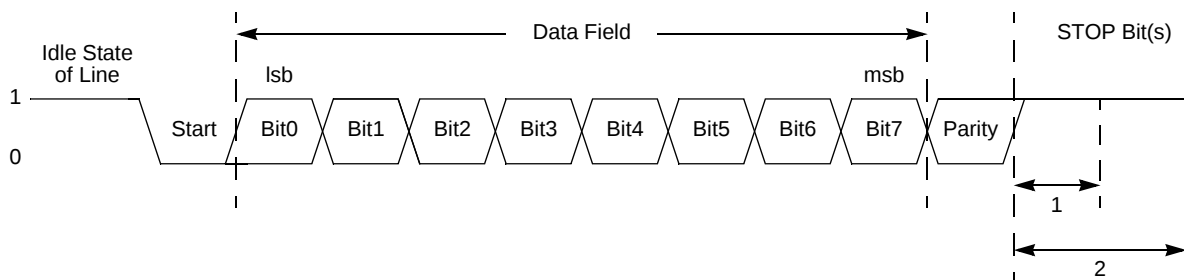


Figure 13. UART Asynchronous Data Format with Parity

Transmitting Data using Polled Method

Follow the steps below 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 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

Follow the steps below 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 desired.
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). Follow the steps below 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 desired.
 - 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:

1. Check the UART Status 0 Register to determine the source of the interrupt-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 has a MULTIPROCESSOR (9-bit) mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In MULTIPROCESSOR mode (also referred to as 9-bit mode), the multiprocessor bit is transmitted following the 8-bits of data and immediately preceding the STOP bit(s) as displayed in Figure 14. The character format is as displayed in Figure 14.

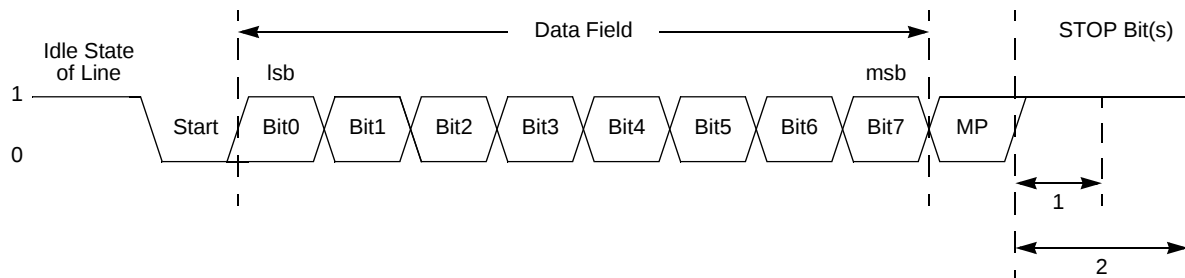


Figure 14. UART Asynchronous MULTIPROCESSOR Mode Data Format

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 does not need to access the UART when it receives data directed to other devices on the

multi-node 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 multi-transceiver bus, such as RS-485.

Driver Enable is an active High signal that envelopes the entire transmitted data frame including parity and STOP bits as displayed in [Figure15](#) on page 97. 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.

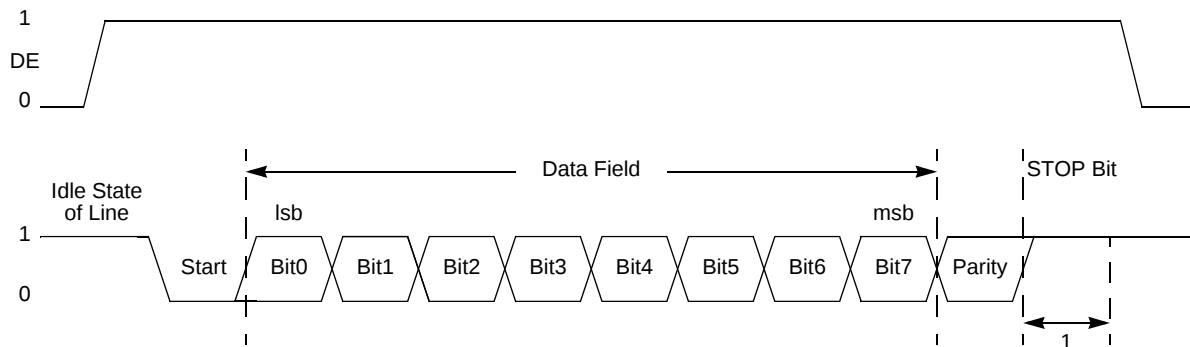


Figure 15. UART Driver Enable Signal Timing (with 1 STOP Bit and Parity)

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 once 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 in the UART Status 0 Register. Updates to the Receive Data Register occur only when the next data word is received.

UART Data and Error Handling Procedure

[Figure16](#) on page 99 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.

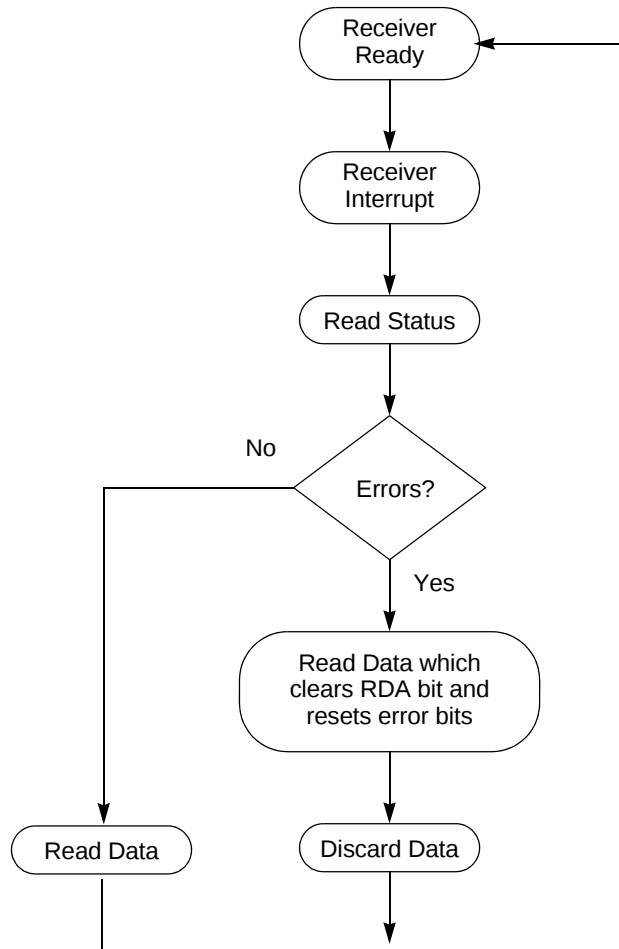


Figure 16. UART Receiver Interrupt Service Routine Flow

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 on time-out. Follow the steps below to configure the BRG as a timer with interrupt on time-out:

1. Disable the UART by clearing the REN and TEN bits in the UART Control 0 Register to 0.
2. Load the desired 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 [Infrared Encoder/Decoder](#) on page 109 for more information on the infrared operation.

UART Transmit Data Register

Data bytes written to the UART Transmit Data Register ([Table 52](#)) 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 52. UART Transmit Data Register (U0TXD)

BITS	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
ADDR	F40H							

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 (Table 53). The Read-only UART Receive Data Register shares a Register File address with the Write-only UART Transmit Data Register.

Table 53. UART Receive Data Register (U0RXD)

BITS	7	6	5	4	3	2	1	0
FIELD	RXD							
RESET	X							
R/W	R							
ADDR	F40H							

RXD—Receive Data

UART receiver data byte from the RXD_x pin

UART Status 0 Register

The UART Status 0 and Status 1 registers (Table 54 and Table 55 on page 102) identify the current UART operating configuration and status.

Table 54. UART Status 0 Register (U0STAT0)

BITS	7	6	5	4	3	2	1	0
FIELD	RDA	PE	OE	FE	BRKD	TDRE	TXE	CTS
RESET	0					1		X
R/W	R							
ADDR	F41H							

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.

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.

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.

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.

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.

TDRE—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.

TXE—Transmitter Empty

This bit indicates that the transmit shift register is empty and character transmission is finished.

0 = Data is currently transmitting.

1 = Transmission is complete.

CTS— $\overline{\text{CTS}}$ Signal

When this bit is read it returns the level of the $\overline{\text{CTS}}$ signal.

UART Status 1 Register

This register contains multiprocessor control and status bits.

Table 55. UART Status 1 Register (U0STAT1)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved						NEWFRM	MPRX
RESET	0							
R/W	R				R/W		R	
ADDR	F44H							

Reserved—Must be 0

NEWFRM—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.

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 (Table 56 and Table 57 on page 104) configure the properties of the UART's transmit and receive operations. The UART Control Registers must not be written while the UART is enabled.

Table 56. UART Control 0 Register (U0CTL0)

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
RESET	0							
R/W	R/W							
ADDR	F42H							

TEN—Transmit Enable

This bit enables or disables the transmitter. The enable is also controlled by the $\overline{\text{CTS}}$ signal and the CTSE bit. If the $\overline{\text{CTS}}$ signal is low and the CTSE bit is 1, the transmitter is enabled.

0 = Transmitter disabled.

1 = Transmitter enabled.

REN—Receive Enable

This bit enables or disables the receiver.

0 = Receiver disabled.

1 = Receiver enabled.

CTSE—CTS Enable

0 = The $\overline{\text{CTS}}$ signal has no effect on the transmitter.

1 = The UART recognizes the $\overline{\text{CTS}}$ signal as an enable control from the transmitter.

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.

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.

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 STOP Bit time desired following the Break.

0 = No break is sent.

1 = The output of the transmitter is zero.

STOP—STOP Bit Select

0 = The transmitter sends one stop bit.

1 = The transmitter sends two stop bits.

LBEN—Loop Back Enable

0 = Normal operation.

1 = All transmitted data is looped back to the receiver.

Table 57. UART Control 1 Register (U0CTL1)

BITS	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							
ADDR	F43H							

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.

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.

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).

DEPOL—Driver Enable Polarity

0 = DE signal is Active High.

1 = DE signal is Active Low.

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.

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.

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 stores the multi-node 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 58. UART Address Compare Register (U0ADDR)

BITS	7	6	5	4	3	2	1	0
FIELD	COMP_ADDR							
RESET	0							
R/W	R/W							
ADDR	F45H							

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 ([Table 59](#) and [Table 60](#)) 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 59. UART Baud Rate High Byte Register (U0BRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1							
R/W	R/W							
ADDR	F46H							

Table 60. UART Baud Rate Low Byte Register (U0BRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	1							
R/W	R/W							
ADDR	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 61](#) provides information on data rate errors for popular baud rates and commonly used crystal oscillator frequencies.

Table 61. UART Baud Rates

10.0 MHz System Clock				5.5296 MHz System Clock			
Desired Rate	BRG Divisor	Actual Rate	Error	Desired Rate	BRG Divisor	Actual Rate	Error
(kHz)	(Decimal)	(kHz)	(%)	(kHz)	(Decimal)	(kHz)	(%)
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

Table 61. UART Baud Rates (Continued)

3.579545 MHz System Clock				1.8432 MHz System Clock			
Desired Rate	BRG Divisor	Actual Rate	Error	Desired Rate	BRG Divisor	Actual Rate	Error
(kHz)	(Decimal)	(kHz)	(%)	(kHz)	(Decimal)	(kHz)	(%)
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.

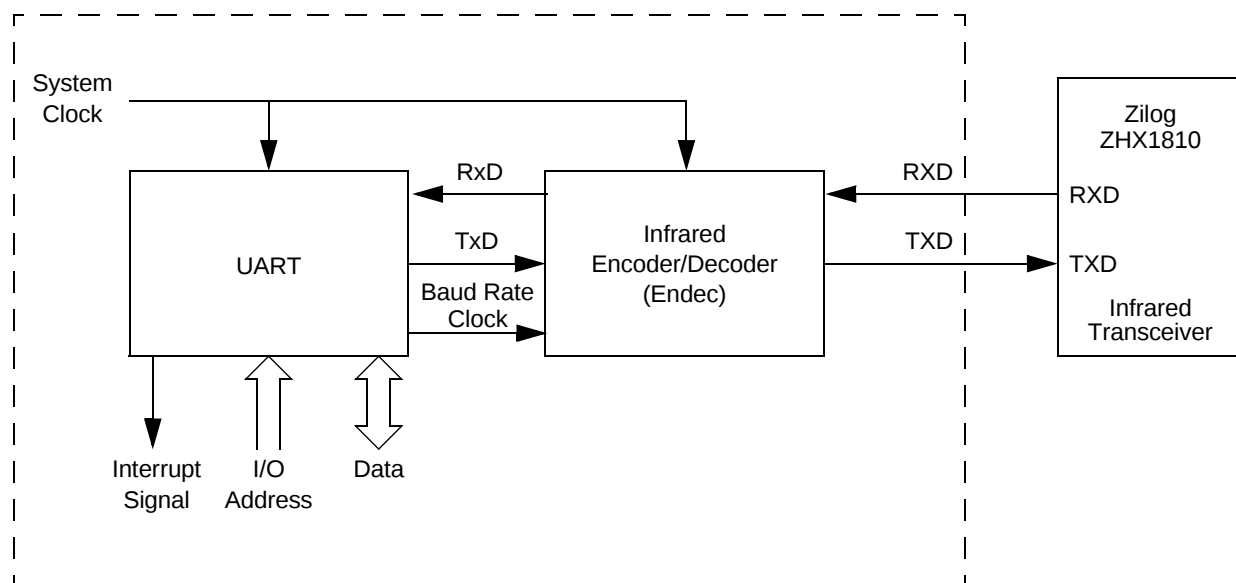


Figure 17. Infrared Data Communication System Block Diagram

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.

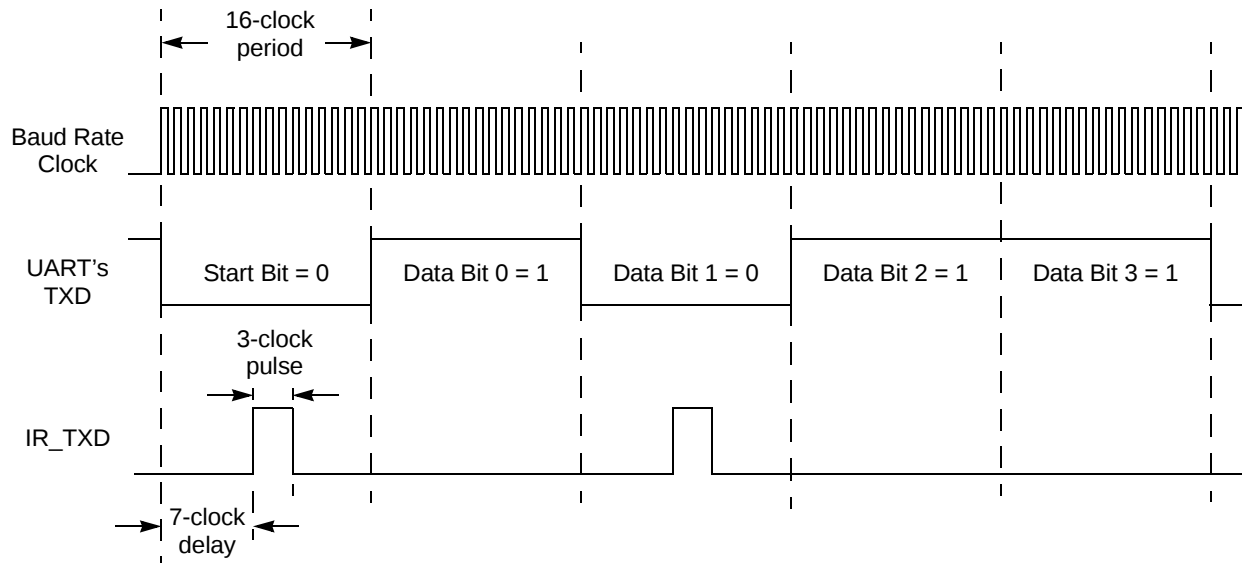


Figure 18. Infrared Data Transmission

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.

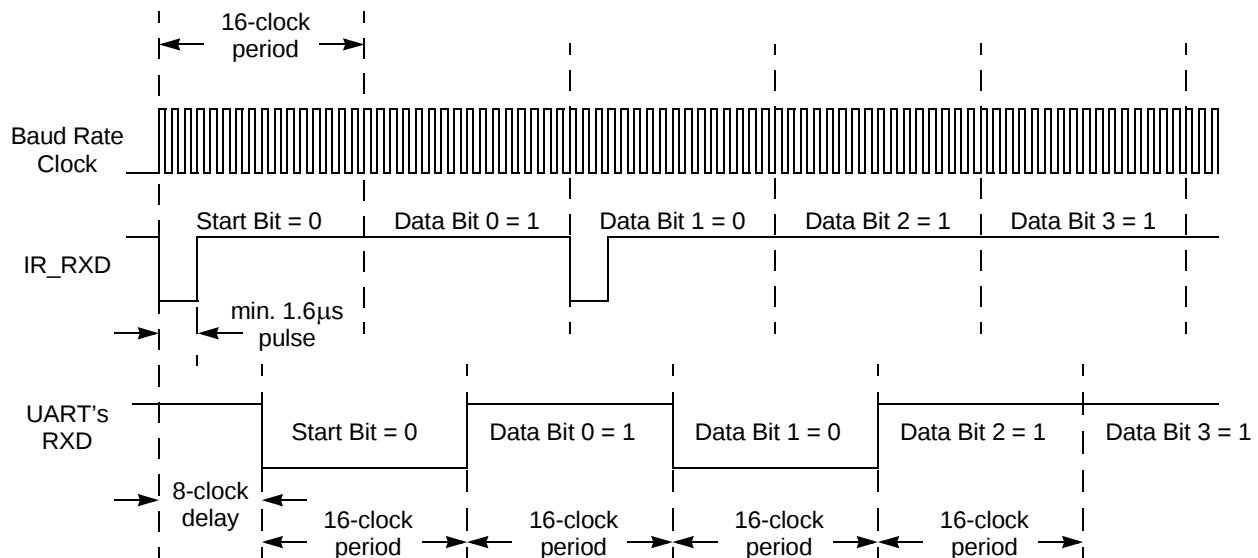


Figure 19. Infrared Data Reception



Caution: The system clock frequency must be at least 1.0 MHz to ensure proper reception of the 1.6 μs minimum width pulses allowed by the IrDA standard.

Endec Receiver Synchronization

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

of minus four baud rate clocks to plus eight 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 the initial state and waits for the next falling edge. As each falling edge is detected, the Endec clock counter is reset, resynchronizing the Endec to the incoming signal. This procedure 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 [UART Control Register Definitions](#) on page 100.



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 configured as either a Master (in single or multi-master systems) or a Slave as displayed in [Figure 20](#) through [Figure 22](#).

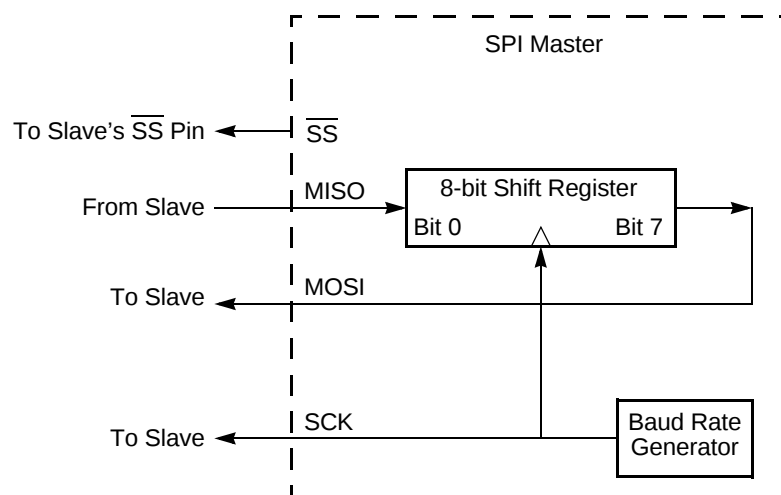


Figure 20. SPI Configured as a Master in a Single Master, Single Slave System

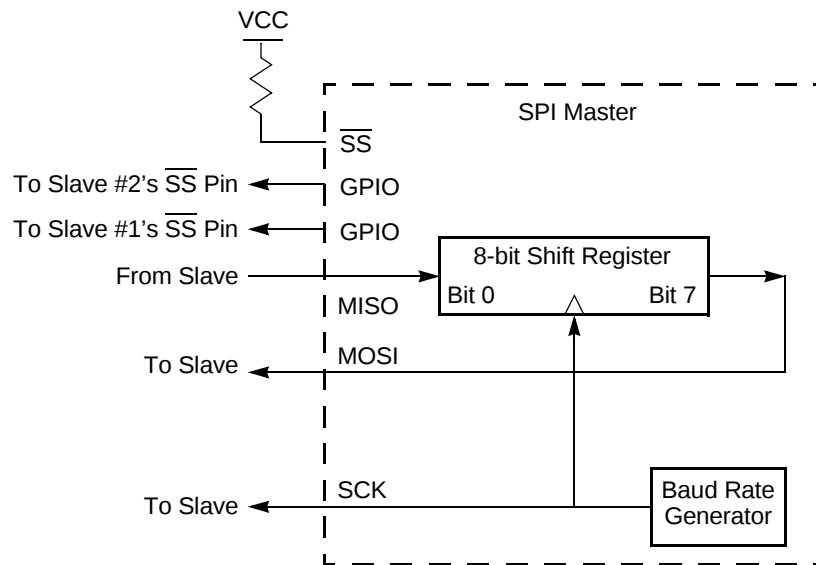


Figure 21. SPI Configured as a Master in a Single Master, Multiple Slave System

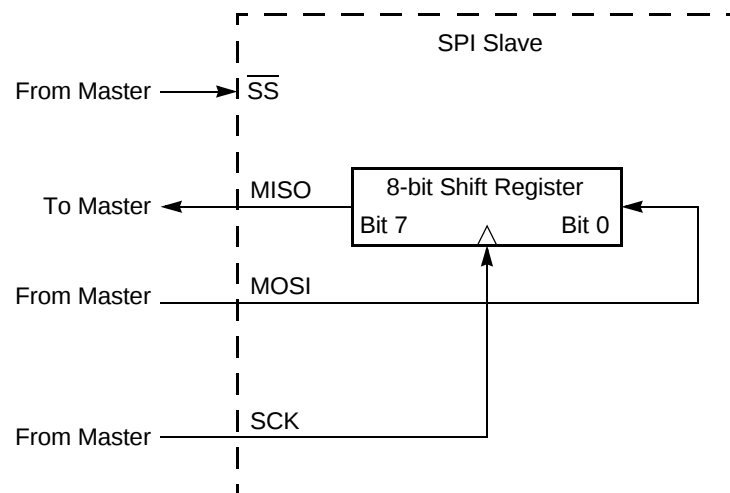


Figure 22. SPI Configured as a Slave

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 multi-bit (typically 8-bit) character is shifted out one data pin and an multi-bit 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 (XIN) clock period.

The Master and Slave are each capable of exchanging a character of data during a sequence of NUMBITS clock cycles (see NUMBITS field in the SPIMODE 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 multi-master 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 62 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 62. 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 since the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the Master and the Slave.

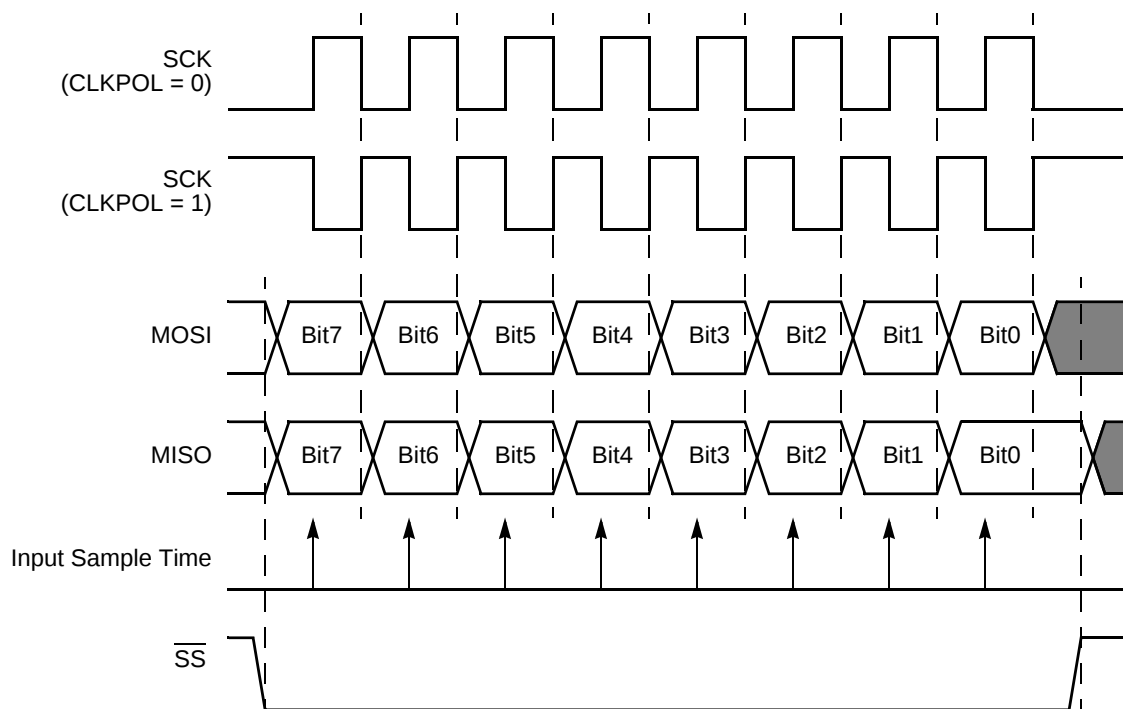


Figure 23. SPI Timing When PHASE is 0

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.

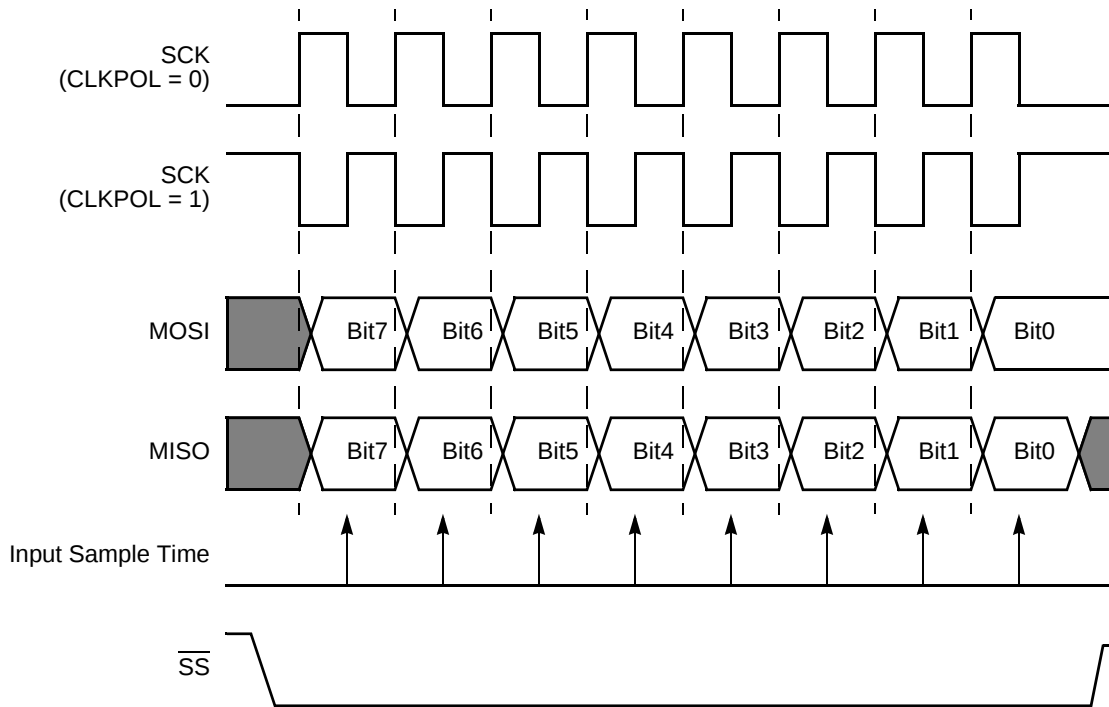


Figure 24. SPI Timing When PHASE is 1

Multi-Master Operation

In a multi-master 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 multi-master 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 multi-master 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 desired to force a “startup” 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 so the SPIBRH and SPIBRL Registers need not 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 (XIN) 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 (Multi-Master Collision)

A mode fault indicates when more than one Master is trying to communicate at the same time (a multi-master 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 as well as the IRQ bit (indicating 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 the bits in a character (see description of Slave Abort Error). 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:

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

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

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

1. Disable the SPI by clearing the $SPIEN$ bit in the SPI Control Register to 0.
2. Load the desired 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:

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

SPI Control Register Definitions

SPI Data Register

The SPI Data Register 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 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 63. SPI Data Register (SPIDATA)

BITS	7	6	5	4	3	2	1	0
FIELD	DATA							
RESET	X							
R/W	R/W							
ADDR	F60H							

DATA—Data

Transmit and/or receive data.

SPI Control Register

The SPI Control Register configures the SPI for transmit and receive operations.

Table 64. SPI Control Register (SPICTL)

BITS	7	6	5	4	3	2	1	0
FIELD	IRQE	STR	BIRQ	PHASE	CLKPOL	WOR	MMEN	SPIEN
RESET	0							
R/W	R/W							
ADDR	F61H							

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.

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.

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.

PHASE—Phase Select

Sets the phase relationship of the data to the clock. For more information on operation of the `PHASE` bit, see [SPI Clock Phase and Polarity Control](#) on page 116.

CLKPOL—Clock Polarity

0 = SCK idles Low (0).
1 = SCK idle High (1).

WOR—Wire-OR (Open-Drain) Mode Enabled

0 = SPI signal pins not configured for open-drain.
1 = All four SPI signal pins (`SCK`, `SS`, `MISO`, `MOSI`) configured for open-drain function. This setting is typically used for multi-master and/or multi-slave configurations.

MMEN—SPI MASTER Mode Enable

0 = SPI configured in SLAVE mode.
1 = SPI configured in MASTER mode.

SPIEN—SPI Enable

0 = SPI disabled.
1 = SPI enabled.

SPI Status Register

The SPI Status Register 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 65. SPI Status Register (SPISTAT)

BITS	7	6	5	4	3	2	1	0
FIELD	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0							1
R/W	R/W*				R			
ADDR	F62H							
R/W* = Read access. Write a 1 to clear the bit to 0.								

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.

OVR—Overflow

0 = An overrun error has not occurred.

1 = An overrun error has been detected.

COL—Collision

0 = A multi-master collision (mode fault) has not occurred.

1 = A multi-master collision (mode fault) has been detected.

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.

Reserved—Must be 0

TXST—Transmit Status

0 = No data transmission currently in progress.

1 = Data transmission currently in progress.

SLAS—Slave Select

If SPI enabled as a Slave

0 = $\overline{SS}$ input pin is asserted (Low)

1 = $\overline{SS}$ input is not asserted (High).

If SPI enabled as a Master, this bit is not applicable.

SPI Mode Register

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

Table 66. SPI Mode Register (SPIMODE)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved		DIAG	NUMBITS[2:0]			SSIO	SSV
RESET	0							
R/W	R		R/W					
ADDR	F63H							

Reserved—Must be 0

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: Take precautions if you are reading the values while BRG is counting.

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 SPI Data Register description for information on 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

SSIO—Slave Select I/O

0 = $\overline{SS}$ pin configured as an input.

1 = $\overline{SS}$ pin configured as an output (MASTER mode only).

SSV—Slave Select Value

If SSIO = 1 and SPI 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 configured as a Slave

SPI Diagnostic State Register

The SPI Diagnostic State Register provides observability of internal state. This is a read only register used for SPI diagnostics.

Table 67. SPI Diagnostic State Register (SPIDST)

BITS	7	6	5	4	3	2	1	0
FIELD	SCKEN	TCKEN	SPISTATE					
RESET	0							
R/W	R							
ADDR	F64H							

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 updates on next system clock)

TCKEN–Transmit Clock Enable

0 = The internal Transmit Clock Enable signal is deasserted.

1 = The internal Transmit Clock Enable signal is asserted. When this is asserted the serial data out is updated on the next system clock (MOSI or MISO).

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 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:

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

Table 68. SPI Baud Rate High Byte Register (SPIBRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1							
R/W	R/W							
ADDR	F66H							



BRH = SPI Baud Rate High Byte
Most significant byte, BRG[15:8], of the SPI Baud Rate Generator’s reload value.

Table 69. SPI Baud Rate Low Byte Register (SPIBRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	1							
R/W	R/W							
ADDR	F67H							

BRL = SPI Baud Rate Low Byte
Least significant byte, BRG[7:0], of the SPI Baud Rate Generator’s reload value.

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.

In order 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 (I²C) drives all Start and Stop signals and initiates all transactions. To start a transaction, the I²C Controller generates a START condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the I²C 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 I²C Control Register control the sending of the Start and Stop conditions. A Master is also allowed to end one transaction and begin a new one by issuing a Restart. This is accomplished by setting the START bit at the end of a transaction, rather than the STOP bit.

► **Note:** *The Start condition not sent until the START bit is set and data has been written to the I²C Data Register.*

Master Write and Read Transactions

The following sections provide a recommended procedure for performing I²C write and read transactions from the I²C Controller (Master) to slave I²C 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 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 I²C 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 I²C Status register equal to 1). In this scenario where software is not keeping up with the I²C 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 I²C 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 I²C 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. Once the I²C Data Register has been read, the I²C reads the next data byte.

Address Only Transaction with a 7-bit Address

In the situation where software determines if a slave with a 7-bit address is responding without sending or receiving data, a transaction can be done which only consists of an address phase. Figure 26 on page 131 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 done to a EEPROM to determine when the EEPROM completes its internal write operation and is once again responding to I²C transactions. If the slave does not Acknowledge, the transaction is repeated until the slave does Acknowledge.

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

Figure 26. 7-Bit Address Only Transaction Format

Follow the steps below 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	$\overline{A/A}$	P/S
---	---------------	-------	---	------	---	------	---	------	------------------	-----

Figure 27. 7-Bit Addressed Slave Data Transfer Format

Follow the steps below 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 with [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 following steps).

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 the situation where software wants to determine if a slave with a 10-bit address is responding without sending or receiving data, a transaction is done which only consists of an address phase. [Figure 28](#) displays this “address only” transaction to determine if a slave with 10-bit address will acknowledge. As an example, this transaction is used after a “write” has been done to a EEPROM to determine when the EEPROM completes its internal write operation and is once again responding to I²C transactions. If the slave does not Acknowledge the transaction is repeated until the slave is able to Acknowledge.

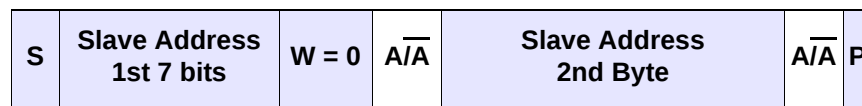


Figure 28. 10-Bit Address Only Transaction Format

Follow the steps below 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 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 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 the I²C Controller sets the ACK bit in the I²C Status register. Continue with [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 response 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).

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 7 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 bits XX 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.

Follow the steps below for a transmit operation on 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.
4. Software responds to the TDRE interrupt by writing the first slave address byte to the I²C Data Register. 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 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 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 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 with [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 following steps).

12. The I²C Controller loads the I²C Shift register with the contents of the I²C Data Register.
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 writing a data byte to the I²C Data Register.
15. The I²C 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 with [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 following steps).

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 also clears the TXI bit of the I²C Control Register at the same time.
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	$\bar{A}$	P/S
---	---------------	-------	---	------	---	------	-----------	-----

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

Follow the steps below 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 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 with [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 following steps).

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	$\bar{A}$	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 bits XX are the two most-significant bits of the 10-bit address. The lowest bit of the first byte transferred is the write control bit.

Follow the steps below 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 with [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 with 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 following steps).

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 following steps).

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

I²C Data Register

The I²C Data Register ([Table 70](#)) 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 70. I²C Data Register (I2CDATA)

BITS	7	6	5	4	3	2	1	0
FIELD	DATA							
RESET	0							
R/W	R/W							
ADDR	F50H							

I²C Status Register

The Read-only I²C Status register (Table 71) indicates the status of the I²C Controller.

Table 71. I²C Status Register (I2CSTAT)

BITS	7	6	5	4	3	2	1	0
FIELD	TDRE	RDRF	ACK	10B	RD	TAS	DSS	NCKI
RESET	1	0						
R/W	R							
ADDR	F51H							

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.

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).

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: Software must be cautious in making decisions based on this bit within a transaction because software cannot tell when the bit is updated by hardware. In the case of write transactions, the I²C 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 [Address Only Transaction with a 7-bit Address](#) on page 131 and [Address Only Transaction with a 10-bit Address](#) on page 133.

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 once the first byte of the address has been sent.

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 I²C Shift register after the START bit is set.

TAS—Transmit Address State

This bit is active High while the address is being shifted out of the I²C Shift Register.

DSS—Data Shift State

This bit is active High while data is being shifted to or from the I²C Shift Register.

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 (Table 72) enables the I²C operation.

Table 72. I²C Control Register (I2CCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	IEN	START	STOP	BIRQ	TXI	NAK	FLUSH	FILTEN
RESET	0							
R/W	R/W	R/W1	R/W1	R/W	R/W	R/W1	W1	R/W
ADDR	F52H							

IEN—I²C Enable

1 = The I²C transmitter and receiver are enabled.

0 = The I²C transmitter and receiver are disabled.

START—Send Start Condition

This bit sends the Start condition. Once 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.

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. Once

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.

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 one.

0 = No BRG interrupt occurs.

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.

NAK—Send NAK

This bit sends a Not Acknowledge condition after the next byte of data is read from the I²C Slave. Once 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.

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.

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 (Tables 73 and 73) 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:

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

Table 73. I²C Baud Rate High Byte Register (I2CBRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	FFH							
R/W	R/W							
ADDR	F53H							

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 74. I²C Baud Rate Low Byte Register (I2CBRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	FFH							
R/W	R/W							
ADDR	F54H							

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 (Table 75) provides observability of internal state. This is a read only register used for I²C diagnostics and manufacturing test.

Table 75. I²C Diagnostic State Register (I2CDST)

BITS	7	6	5	4	3	2	1	0
FIELD	SCLIN	SDAIN	STPCNT	TXRXSTATE				
RESET	X		0					
R/W	R							
ADDR	F55H							

SCLIN—Value of Serial Clock input signal

SDAIN—Value of the Serial Data input signal

STPCNT—Value of the internal Stop Count control signal

TXRXSTATE—Value of the internal I²C 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

TXRXSTATE	State Description
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 ([Table 76](#)) provides control over diagnostic modes. This register is a read/write register used for I²C diagnostics.

Table 76. I²C Diagnostic Control Register (I2CDIAG)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved							DIAG
RESET	0							
R/W	R							R/W
ADDR	F56H							

DIAG = Diagnostic Control Bit—Selects 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 5-input analog multiplexer selects one of the 5 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 VREF pin or generated internally by the voltage reference generator.

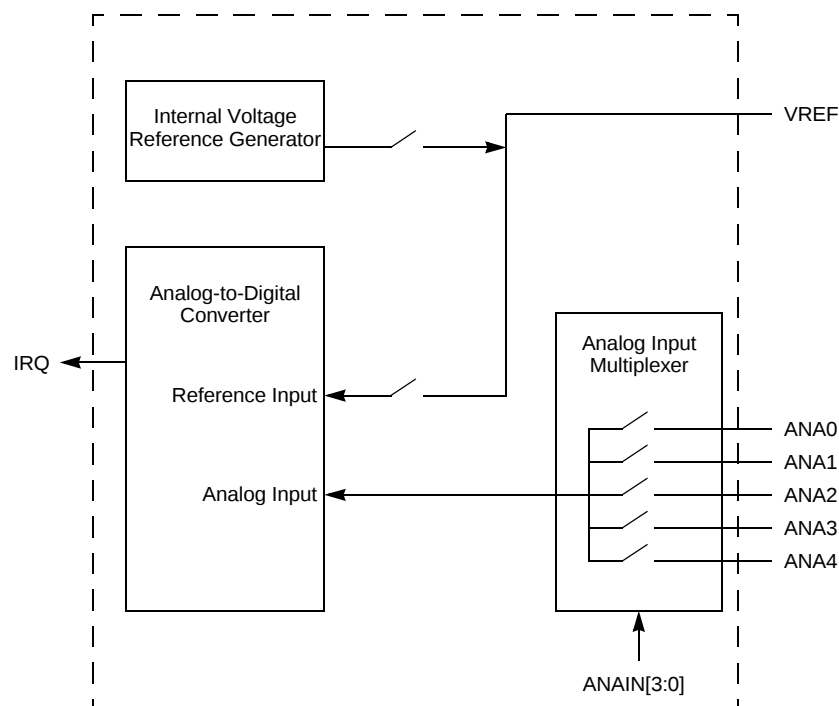


Figure 32. Analog-to-Digital Converter Block Diagram

Operation

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 power-down state, the ADC requires 40 system clock cycles to power-up. The ADC powers up when a conversion is requested using 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. Follow the steps below for setting up the ADC and initiating a single-shot conversion:

1. Enable the desired 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 bit fields in the ADC Control Register is 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 `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.*

Follow the steps below for setting up the ADC and initiating continuous conversion:

1. Enable the desired 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 ANA_{IN}[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

ADC Control Register

The ADC Control Register selects the analog input channel and initiates the analog-to-digital conversion.

Table 77. ADC Control Register (ADCCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	CEN	Reserved	VREF	CONT	ANAIN[3:0]			
RESET	0		1	0				
R/W	R/W							
ADDR	F70H							

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.

Reserved—Must be 0

VREF

0 = Internal reference generator enabled. The VREF pin must be left unconnected or capacitively coupled to analog ground (AVSS).

1 = Internal voltage reference generator disabled. An external voltage reference must be provided through the VREF pin.

CONT

0 = SINGLE-SHOT conversion. ADC data is output once at completion of the 5129 system clock cycles.

1 = Continuous conversion. ADC data updated every 256 system clock cycles.

ANAIN—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 [Signal and Pin Descriptions](#) 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 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 given 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 78. ADC Data High Byte Register (ADCD_H)

BITS	7	6	5	4	3	2	1	0
FIELD	ADCD_H							
RESET	X							
R/W	R							
ADDR	F72H							

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 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 given by {ADCD_H[7:0], ADCD_L[7:6]}.

Table 79. ADC Data Low Bits Register (ADCD_L)

BITS	7	6	5	4	3	2	1	0
FIELD	ADCD_L		Reserved					
RESET	X							
R/W	R							
ADDR	F73H							



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.

Reserved

These bits are reserved and are always undefined.

Flash Memory

The products in Z8 Encore! XP[®] F0822 Series feature either 8 KB (8192) or 4 KB (4096) bytes of Flash memory with Read/Write/Erase capability. The 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. The Flash memory is divided into eight sectors which is protected from programming and erase operations on a per sector basis.

[Table 80](#) describes the Flash memory configuration for each device in the Z8F082x family. [Table 81](#) lists the sector address ranges. [Figure 33](#) on page 154 displays the Flash memory arrangement.

Table 80. Flash Memory Configurations

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

Table 81. Flash Memory Sector Addresses

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

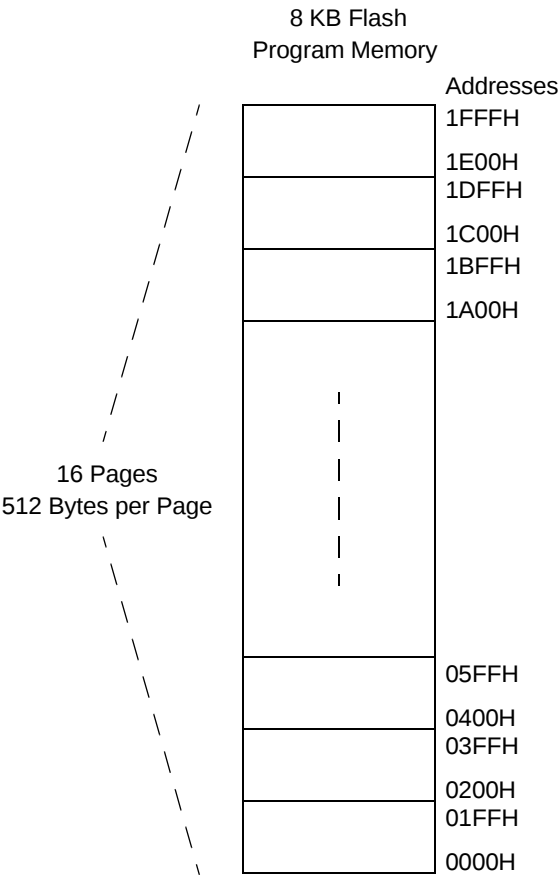


Figure 33. Flash Memory Arrangement

Information Area

Table 82 on page 155 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 82. 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 Byte Programming, Page Erase, and Mass Erase of the 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 on 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 on the Flash memory, you must first configure the Flash Frequency High and Low Byte registers. The Flash Frequency registers allow programming and erasure of the Flash with system clock frequencies ranging from 20 kHz through 20 MHz (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:

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

**Caution:**

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

Flash Read Protection

The user code contained within the 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 [Option Bits](#) on page 163 and [On-Chip Debugger](#) on page 171.

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 the Flash memory. To program or erase the 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.

Follow the steps below 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. Re-write 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. Once 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 desired.

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 re-enables access to the Page Select Register.

Follow the steps below 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 [Option Bits](#) on page 163.

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 *eZ8 CPU Core User Manual (UM0128)* for a description of the LDC and LDCI instructions.

While the Flash Controller programs the 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 once 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.*

Follow the steps below 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. Re-write the page written in [step 2](#) to the Page Select Register.
6. Write Flash Memory using LDC or LDCI instructions to program the Flash.
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 once 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.

Follow the steps below 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. Re-write 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

The Flash memory cannot be Mass Erased by user code.

Flash Controller Bypass

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

Flash Controller Bypass is recommended for gang programming applications and large volume customers who do not require in-circuit programming of the Flash memory.

For more information on bypassing the Flash Controller, refer to *Third-Party Flash Programming Support for Z8 Encore! XP*, 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

Flash Control Register

The Flash Control Register ([Table 83](#)) 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 83. Flash Control Register (FCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	FCMD							
RESET	0							
R/W	W							
ADDR	FF8H							

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.

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

Flash Status Register

The Flash Status Register (Table 84) 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 84. Flash Status Register (FSTAT)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved		FSTAT					
RESET	0							
R/W	R							
ADDR	FF8H							

Reserved

These bits are reserved and must be 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 (Table 85) 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 given 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 85. Page Select Register (FPS)

BITS	7	6	5	4	3	2	1	0
FIELD	INFO_EN	PAGE						
RESET	0							
R/W	R/W							
ADDR	FF9H							

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.

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 (Table 86) 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).

Table 86. Flash Sector Protect Register (FPROT)

BITS	7	6	5	4	3	2	1	0
FIELD	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
RESET	0							
R/W	R/W1							
ADDR	FF9H							

R/W1 = Register is accessible for Read operations. Register can be written to 1 only (using user code).

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.

Flash Frequency High and Low Byte Registers

The Flash Frequency High and Low Byte Registers (Table 87 and Table 88) 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:

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

**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 insure proper program and erase times.

Table 87. Flash Frequency High Byte Register (FFREQH)

BITS	7	6	5	4	3	2	1	0
FIELD	FFREQH							
RESET	0							
R/W	R/W							
ADDR	FFAH							

Table 88. Flash Frequency Low Byte Register (FFREQL)

BITS	7	6	5	4	3	2	1	0
FIELD	FFREQL							
RESET	0							
R/W	R/W							
ADDR	FFBH							

FFREQH and 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 Brownout configuration—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

Option Bit Configuration By Reset

During any reset operation (System Reset, Reset, or Stop Mode Recovery), the Option Bits are automatically read from the 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 ([Table 89](#) on page 164) and 0001H ([Table 90](#) on page 165) 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 be left in its unprogrammed state.

Flash Memory Address 0000H

Table 89. Option Bits at Flash Memory Address 0000H for 8K Series Flash Devices

BITS	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							
ADDR	Program Memory 0000H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

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.

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. Once enabled, the Watchdog Timer can only be disabled by a Reset or Stop Mode Recovery. This setting is the default for unprogrammed (erased) Flash.

OSC_SEL[1:0]—OSCILLATOR Mode Selection

- 00 = On-chip oscillator configured for use with external RC networks (<4 MHz).
- 01 = Minimum power for use with very low frequency crystals (32 kHz to 1.0 MHz).
- 10 = Medium power for use with medium frequency crystals or ceramic resonators (0.5 MHz to 10.0 MHz).
- 11 = Maximum power for use with high frequency crystals (8.0 MHz to 20.0 MHz). This setting is the default for unprogrammed (erased) Flash.

VBO_AO—Voltage Brownout Protection Always On

- 0 = Voltage Brownout Protection is disabled in STOP mode to reduce total power consumption.
- 1 = Voltage Brownout Protection is always enabled including during STOP mode. This setting is the default for unprogrammed (erased) Flash.

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.



Reserved

These Option Bits are reserved for future use and must always be 1.

The following information applies only to the Flash versions of the F0822 Series devices:

FWP—Flash Write Protect

These two Option Bits combine to provide three levels of Program Memory protection:

FWP	Description
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.

Flash Memory Address 0001H

Table 90. Options Bits at Flash Memory Address 0001H

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved							
RESET	U							
R/W	R/W							
ADDR	Program Memory 0001H							

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

Reserved

These Option Bits are reserved for future use and must always be 1. This setting is the default for unprogrammed (erased) Flash.



On-Chip Oscillator

Z8 Encore! XP[®] F0822 Series products feature an on-chip oscillator for use with external crystals with frequencies from 32 kHz to 20 MHz. In addition, the oscillator can support external RC networks with oscillation frequencies up to 4 MHz or ceramic resonators with oscillation frequencies up to 20 MHz. 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 (32 kHz–20 MHz). If an external clock generator is used, the X_{OUT} pin must be left 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 (<4 MHz).
- Minimum power for use with very low frequency crystals (32 kHz to 1.0 MHz).
- Medium power for use with medium frequency crystals or ceramic resonators (0.5 MHz to 10.0 MHz).
- Maximum power for use with high frequency crystals or ceramic resonators (8.0 MHz to 20.0 MHz).

The oscillator mode is selected through user-programmable Option Bits. For more information, see [Option Bits](#) on page 163.

Crystal Oscillator Operation

[Figure 34](#) on page 168 displays a recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 20 MHz. Recommended 20 MHz crystal specifications are provided in [Table 91](#) on page 168. Resistor R1 is optional and limits total power dissipation by the crystal. The printed circuit board layout must add no more than 4 pF 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.

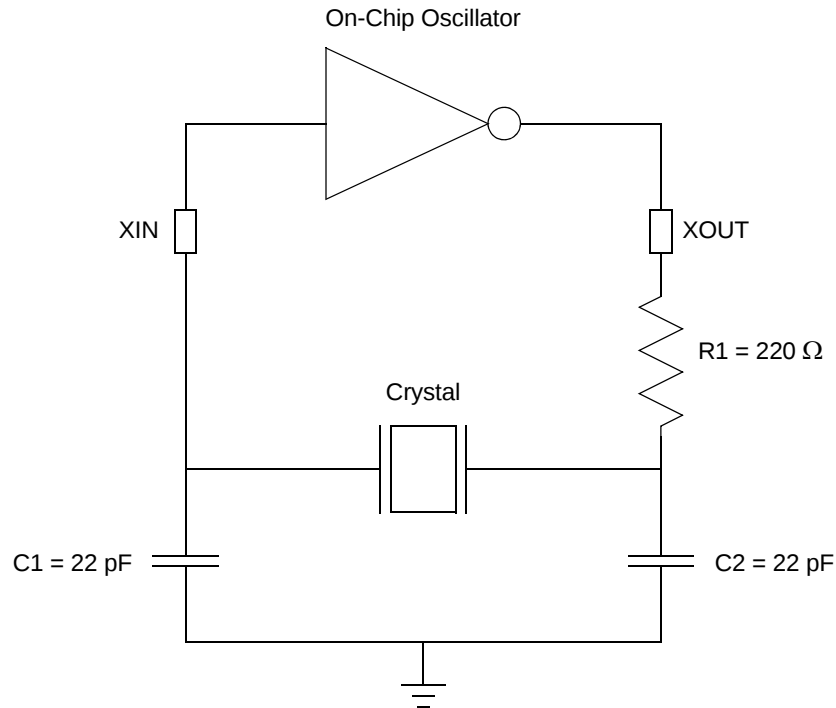


Figure 34. Recommended 20 MHz Crystal Oscillator Configuration

Table 91. Recommended Crystal Oscillator Specifications (20 MHz Operation)

Parameter	Value	Units	Comments
Frequency	20	MHz	
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	25	W	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 35](#) on page 169 displays a recommended configuration for connection with an external resistor-capacitor (RC) network.

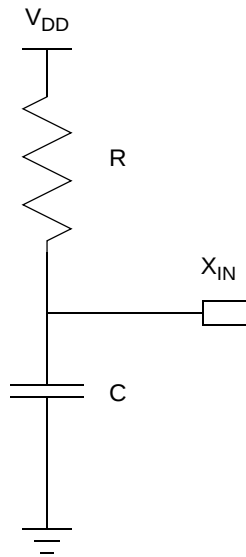


Figure 35. Connecting the On-Chip Oscillator to an External RC Network

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

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

[Figure 36](#) on page 170 displays the typical (3.3 V and 25 $^{\circ}\text{C}$) oscillator frequency as a function of the capacitor (C in pF) employed in the RC network assuming a 45 k Ω external resistor. For very small values of C , the parasitic capacitance of the oscillator XIN 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 20 pF are recommended.

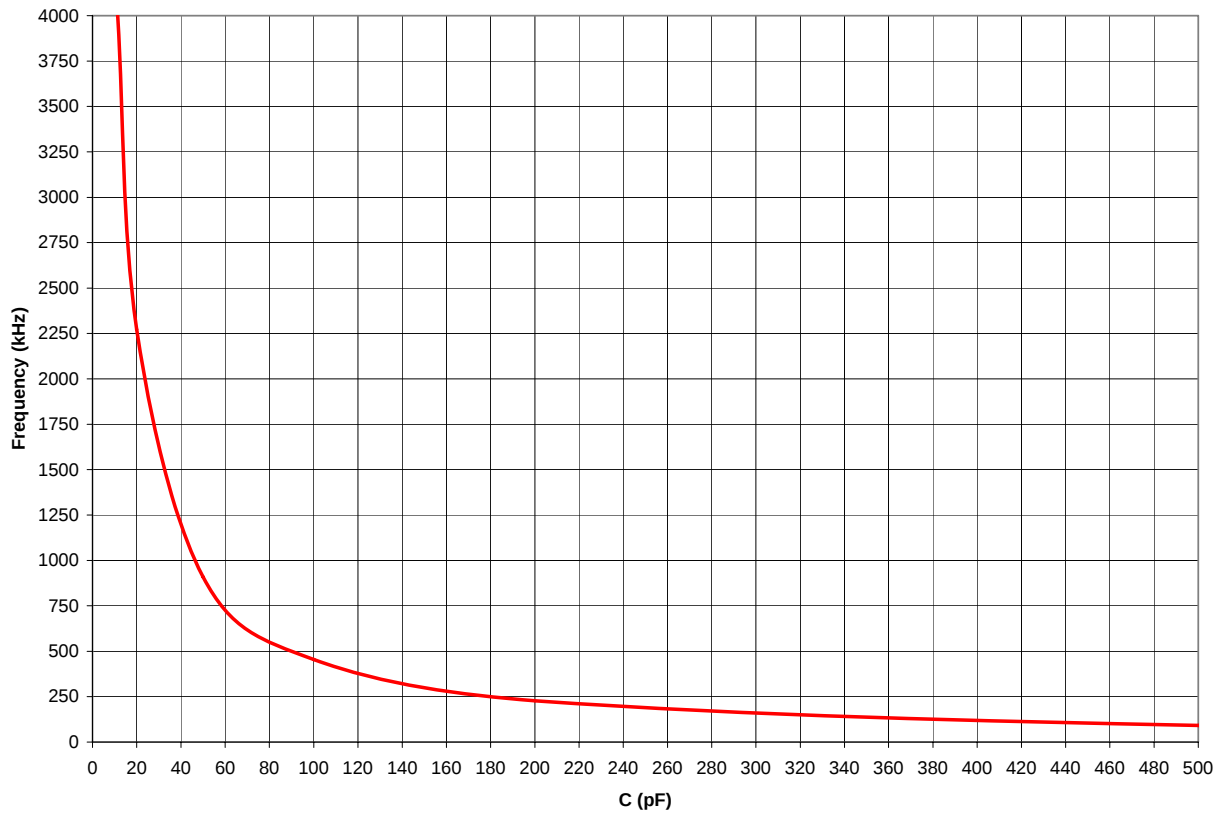


Figure 36. Typical RC Oscillator Frequency as a Function of the External Capacitance with a 45 kΩ Resistor



Caution:

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

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 37](#) displays the architecture of the OCD.

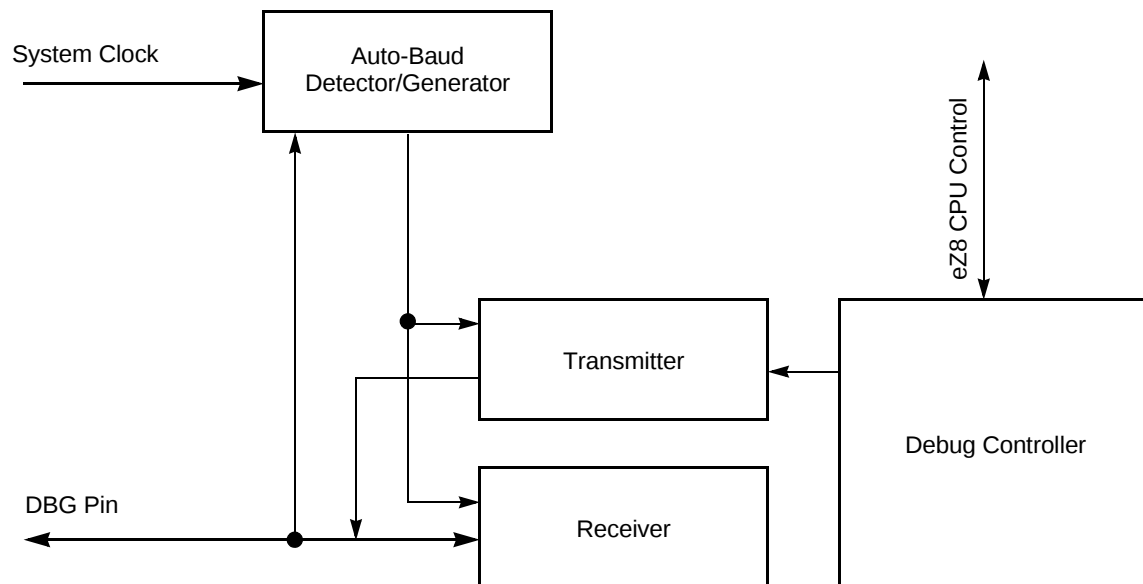


Figure 37. On-Chip Debugger Block Diagram

Operation

OCD Interface

The OCD uses the DBG pin for communication with an external host. This one-pin interface is a bi-directional 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 displayed in Figure 38 and Figure 39.



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 insure proper operation.

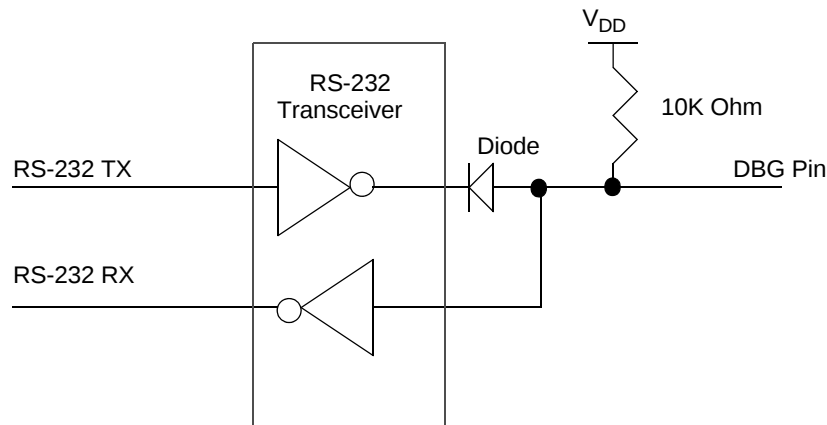


Figure 38. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (1)

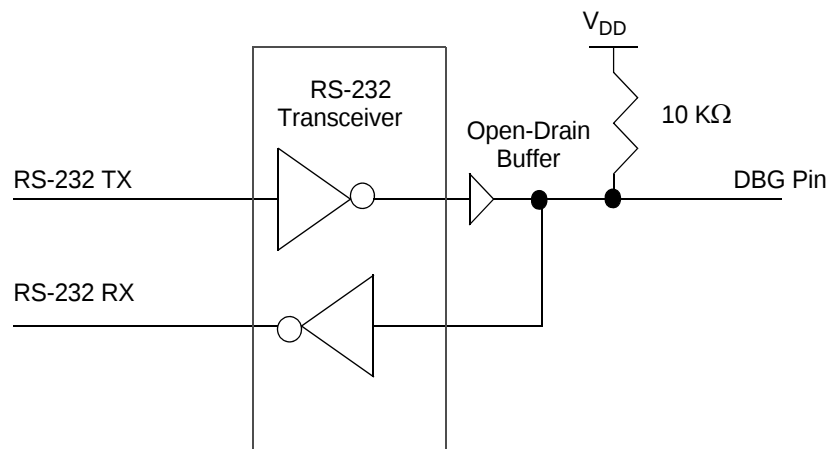


Figure 39. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (2)

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 BRK (Breakpoint) instruction.
- Match of PC to OCDCNTR register (when enabled)
- OCDCNTR register decrements to 0000H (when enabled)
- If the DBG pin is Low when the device exits Reset, the OCD automatically puts 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 Brownout 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 40](#)).

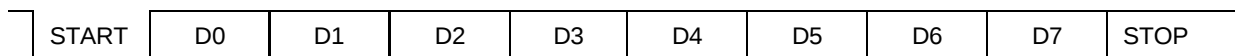


Figure 40. OCD Data Format

OCD Auto-Baud Detector/Generator

To run over a range of baud rates (bits per second) with various system clock frequencies, the OCD contains an Auto-Baud 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 Auto-Baud Detector measures this period and sets the OCD Baud Rate Generator accordingly.

The Auto-Baud 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 (32 kHz)	4.096	0.064

If the OCD receives a Serial Break (nine or more continuous bits Low) the Auto-Baud Detector/Generator resets. The Auto-Baud 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 4096 System Clock cycles long back to the host, and resets the Auto-Baud 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 Auto-Baud 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 (opcode 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 an 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 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, since 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 opcode 00H, which corresponds to the fully programmed state of a byte in Flash memory. To implement a Breakpoint, write 00H to the desired 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 OCD-CNTR is being used to generate a BRK, its value should be written as a last step before leaving DEBUG mode.

Since 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). The 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](#) on page 177 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.
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

Table 93. On-Chip Debugger Commands (Continued)

Debug Command	Command Byte	Enabled when NOT in DEBUG mode?	Disabled by Read Protect Option Bit
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. Since 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]

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 effected (WDTSTAT register is affected by 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 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 opcode. 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

OCD Control Register

The OCD Control Register 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)

BITS	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

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.

BRKEN—Breakpoint Enable

This bit controls the behavior of the BRK instruction (opcode 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 BRK-LOOP bit.

0 = BRK instruction is disabled.

1 = BRK instruction is enabled.

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.

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.

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

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

Reserved

These bits are reserved and must be 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 reports status information about the current state of the debugger and the system.

Table 95. OCD Status Register (OCDSTAT)

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

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.

HALT—HALT Mode

0 = The device is not in HALT mode.

1 = The device is in HALT mode.



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.

Reserved. Must be 0

Electrical Characteristics

Absolute Maximum Ratings



Caution: Stresses greater than those listed in [Table 96](#) can cause permanent damage to the device.

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}).

Table 96. 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	
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	

Table 96. Absolute Maximum Ratings (Continued)

Parameter	Minimum	Maximum	Units	Notes
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 supporting analog input (Port B), and where noted otherwise.				

DC Characteristics

Table 97 lists the DC characteristics of the Z8 Encore! XP[®] F0822 Series products. All voltages are referenced to V_{SS} , the primary system ground.

Table 97. DC Characteristics

Symbol	Parameter	$T_A = -40\text{ }^{\circ}\text{C to }105\text{ }^{\circ}\text{C}$			Units	Conditions
		Minimum	Typical	Maximum		
V_{DD}	Supply Voltage	2.7	—	3.6	V	
V_{IL1}	Low Level Input Voltage	-0.3	—	$0.3 \cdot V_{DD}$	V	For all input pins except $\overline{\text{RESET}}$, DBG, and XIN.
V_{IL2}	Low Level Input Voltage	-0.3	—	$0.2 \cdot V_{DD}$	V	For $\overline{\text{RESET}}$, DBG, and XIN.
V_{IH1}	High Level Input Voltage	$0.7 \cdot V_{DD}$	—	5.5	V	Ports A and C pins when their programmable pull-ups are disabled.
V_{IH2}	High Level Input Voltage	$0.7 \cdot V_{DD}$	—	$V_{DD} + 0.3$	V	Port B pins. Ports A and C pins when their programmable pull-ups are enabled.
V_{IH3}	High Level Input Voltage	$0.8 \cdot V_{DD}$	—	$V_{DD} + 0.3$	V	$\overline{\text{RESET}}$, DBG, and XIN pins.
V_{OL1}	Low Level Output Voltage	—	—	0.4	V	$I_{OL} = 2\text{ mA}$; $V_{DD} = 3.0\text{ V}$ High Output Drive disabled.
V_{OH1}	High Level Output Voltage	2.4	—	—	V	$I_{OH} = -2\text{ mA}$; $V_{DD} = 3.0\text{ V}$ High Output Drive disabled.
V_{OL2}	Low Level Output Voltage High Drive	—	—	0.6	V	$I_{OL} = 20\text{ mA}$; $V_{DD} = 3.3\text{ V}$ High Output Drive enabled $T_A = -40\text{ }^{\circ}\text{C to }+70\text{ }^{\circ}\text{C}$
V_{OH2}	High Level Output Voltage High Drive	2.4	—	—	V	$I_{OH} = -20\text{ mA}$; $V_{DD} = 3.3\text{ V}$ High Output Drive enabled; $T_A = -40\text{ }^{\circ}\text{C to }+70\text{ }^{\circ}\text{C}$
V_{OL3}	Low Level Output Voltage High Drive	—	—	0.6	V	$I_{OL} = 15\text{ mA}$; $V_{DD} = 3.3\text{ V}$ High Output Drive enabled; $T_A = +70\text{ }^{\circ}\text{C to }+105\text{ }^{\circ}\text{C}$
V_{OH3}	High Level Output Voltage High Drive	2.4	—	—	V	$I_{OH} = 15\text{ mA}$; $V_{DD} = 3.3\text{ V}$ High Output Drive enabled; $T_A = +70\text{ }^{\circ}\text{C to }+105\text{ }^{\circ}\text{C}$

Table 97. DC Characteristics (Continued)

Symbol	Parameter	T _A = -40 °C to 105 °C			Units	Conditions
		Minimum	Typical	Maximum		
V _{RAM}	RAM Data Retention	0.7	–	–	V	
I _{IL}	Input Leakage Current	-5	–	+5	μA	V _{DD} = 3.6 V; V _{IN} = VDD or VSS ¹
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}	XIN Pad Capacitance	–	8.0 ²	–	pF	
C _{XOUT}	XOUT Pad Capacitance	–	9.5 ²	–	pF	
I _{PU1}	Weak Pull-up Current	9	20	50	μA	VDD = 2.7–3.6 V. T _A = 0 °C to +70 °C
I _{PU2}	Weak Pull-up Current	7	20	75	μA	VDD = 2.7–3.6 V. T _A = -40 °C to +105 °C

¹ This condition excludes all pins that have on-chip pull-ups, when driven Low.

² These values are provided for design guidance only and are not tested in production.

Figure 41 on page 189 displays the typical active mode current consumption while operating at 25 °C, 3.3 V, versus the system clock frequency. All GPIO pins are configured as outputs and driven High.

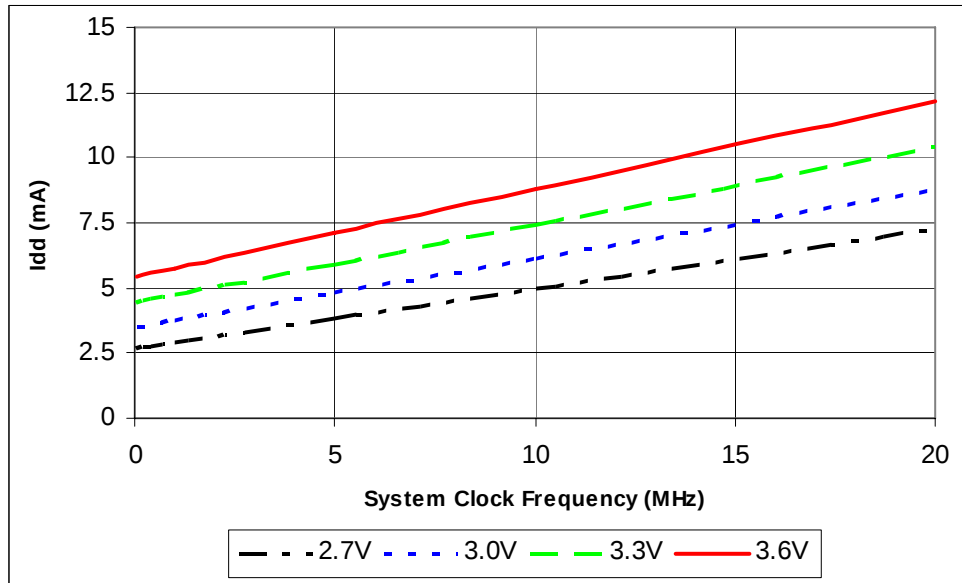


Figure 41. Typical Active Mode I_{DD} Versus System Clock Frequency

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

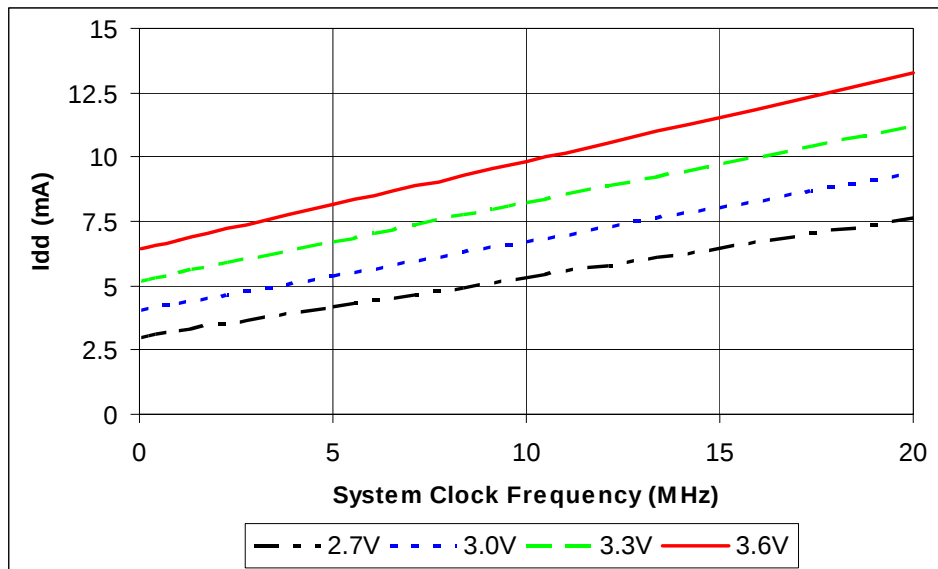


Figure 42. Maximum Active Mode I_{DD} Versus System Clock Frequency

Figure 43 displays the typical current consumption in HALT mode while operating at 25 °C versus the system clock frequency. All GPIO pins are configured as outputs and driven High.

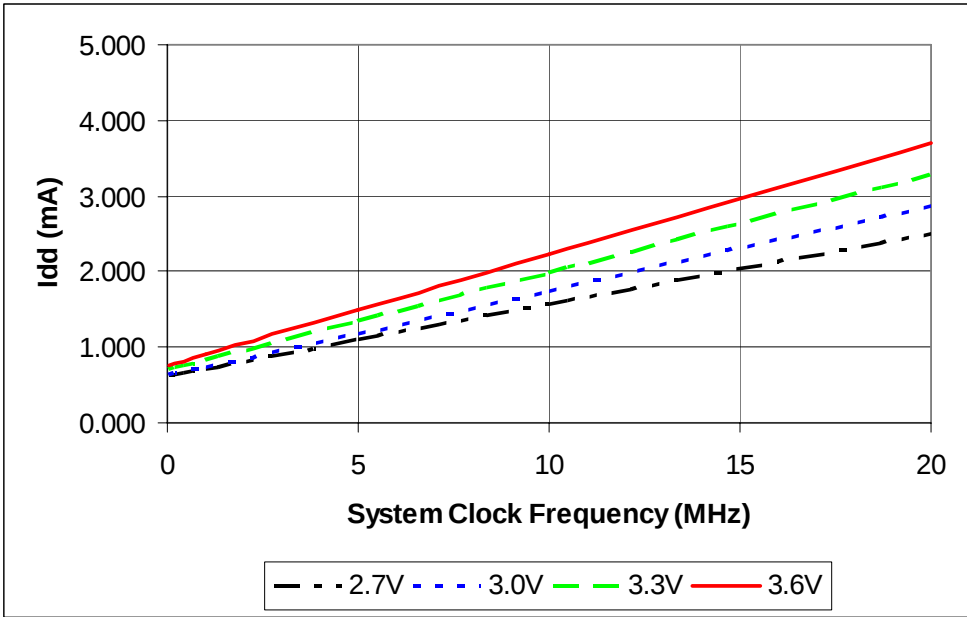


Figure 43. Typical HALT Mode I_{DD} Versus System Clock Frequency

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

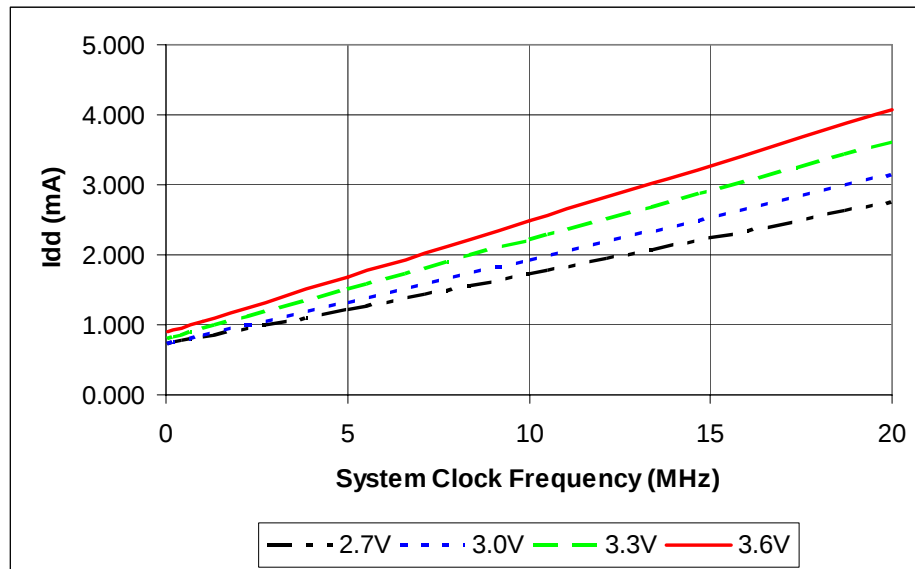


Figure 44. Maximum HALT Mode I_{CC} Versus System Clock Frequency

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

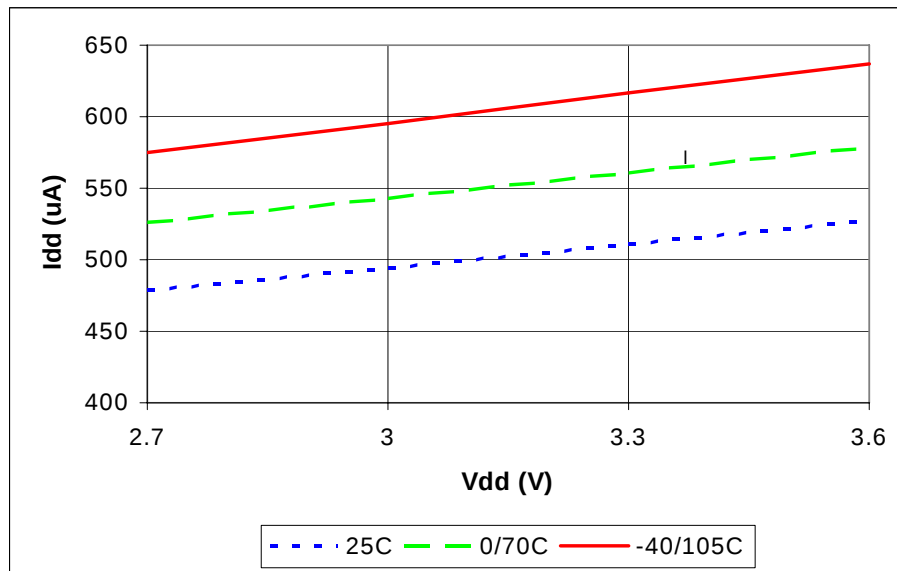


Figure 45. Maximum STOP Mode I_{DD} with VBO Enabled versus Power Supply Voltage

Figure 46 on page 193 displays the maximum current consumption in STOP mode with the VBO disabled and Watchdog Timer enabled versus 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 indistinguishable on the scale of Figure 46 on page 193.

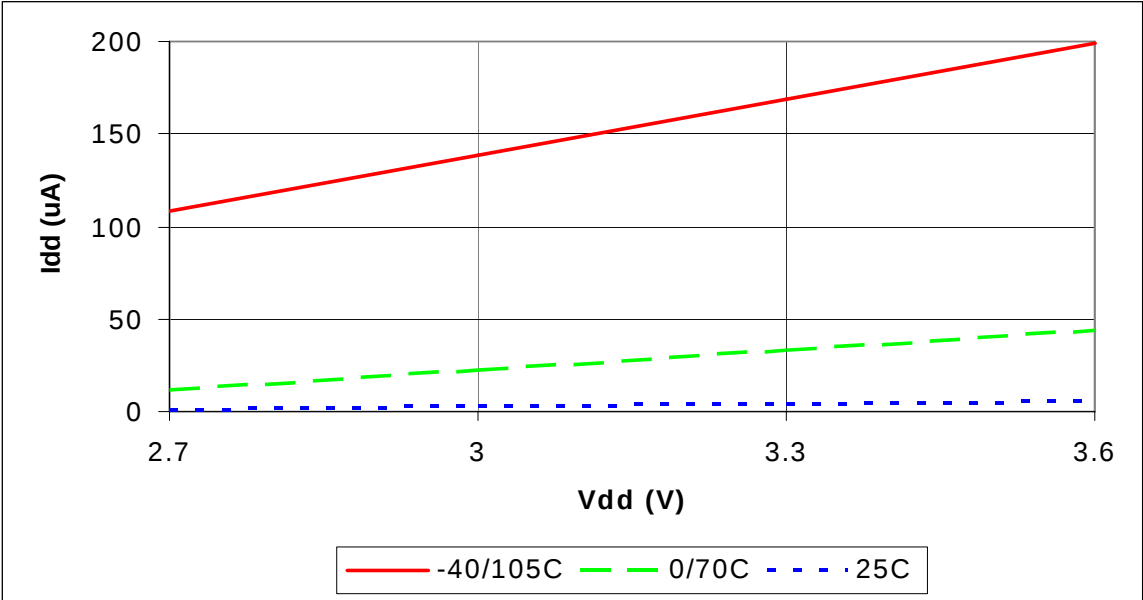


Figure 46. Maximum STOP Mode I_{DD} with VBO Disabled versus Power Supply Voltage

AC Characteristics

Table 98 provides information on the AC characteristics and timing. All AC timing information assumes a standard load of 50 pF on all outputs.

Table 98. AC Characteristics

Symbol	Parameter	V _{DD} = 2.7 - 3.6 V T _A = -40 °C to 105 °C		Units	Conditions
		Minimum	Maximum		
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 the 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} = 50 ns
T _{XINL}	System Clock Low Time	20	30	ns	T _{CLK} = 50 ns

On-Chip Peripheral AC and DC Electrical Characteristics

Table 99 provides information on the Power-On Reset and Voltage Brownout electrical characteristics.

Table 99. Power-On Reset and Voltage Brownout Electrical Characteristics and Timing

Symbol	Parameter	T _A = -40 °C to 105 °C			Units	Conditions
		Minimum	Typical ¹	Maximum		
V _{POR}	Power-On Reset Voltage Threshold	2.15	2.40	2.60	V	V _{DD} = V _{POR}
V _{VBO}	Voltage Brownout 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 (10 kHz) + 16 System Clock cycles (20 MHz)
T _{VBO}	Voltage Brownout 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	

¹ 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 100 provides information on the external RC oscillator electrical characteristics and timing, and Table 101 provides information on the Flash memory electrical characteristics and timing.

Table 100. External RC Oscillator Electrical Characteristics and Timing

Symbol	Parameter	T _A = -40 °C to 105 °C			Units	Conditions
		Minimum	Typical ¹	Maximum		
V _{DD}	Operating Voltage Range	2.70 ¹	—	—	V	
R _{EXT}	External Resistance from XIN to VDD	40	45	200	kΩ	
C _{EXT}	External Capacitance from XIN to VSS	0	20	1000	pF	
F _{OSC}	External RC Oscillation Frequency	—	—	4	MHz	
¹ When using the external RC oscillator mode, the oscillator can stop oscillating if the power supply drops below 2.7 V, but before the power supply drops to the voltage brown-out threshold. The oscillator will resume oscillation as soon as the supply voltage exceeds 2.7 V.						

Table 101. Flash Memory Electrical Characteristics and Timing

Parameter	V _{DD} = 2.7 - 3.6V T _A = -40 °C to 105 °C			Units	Notes
	Minimum	Typical	Maximum		
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.

Table 101. Flash Memory Electrical Characteristics and Timing (Continued)

Parameter	V _{DD} = 2.7 - 3.6V T _A = -40 °C to 105 °C			Units	Notes
	Minimum	Typical	Maximum		
Data Retention	100	–	–	years	25 °C
Endurance	10,000	–	–	cycles	Program / erase cycles

Table 102 provides information on the Reset and Stop Mode Recovery pin timing and Table 103 provides information on the Watchdog Timer Electrical Characteristics and Timing.

Table 102. Reset and Stop Mode Recovery Pin Timing

Symbol	Parameter	T _A = -40 °C to 105 °C			Units	Conditions
		Minimum	Typical	Maximum		
T _{RESET}	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	RESET, DBG and GPIO pins configured as SMR sources.

¹ When using the external RC oscillator mode, the oscillator can stop oscillating if the power supply drops below 2.7 V, but before the power supply drops to the voltage brown-out threshold. The oscillator will resume oscillation as soon as the supply voltage exceeds 2.7 V.

Table 103. Watchdog Timer Electrical Characteristics and Timing

Symbol	Parameter	V _{DD} = 2.7–3.6 V T _A = -40 °C to 105 °C			Units	Conditions
		Minimum	Typical	Maximum		
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.

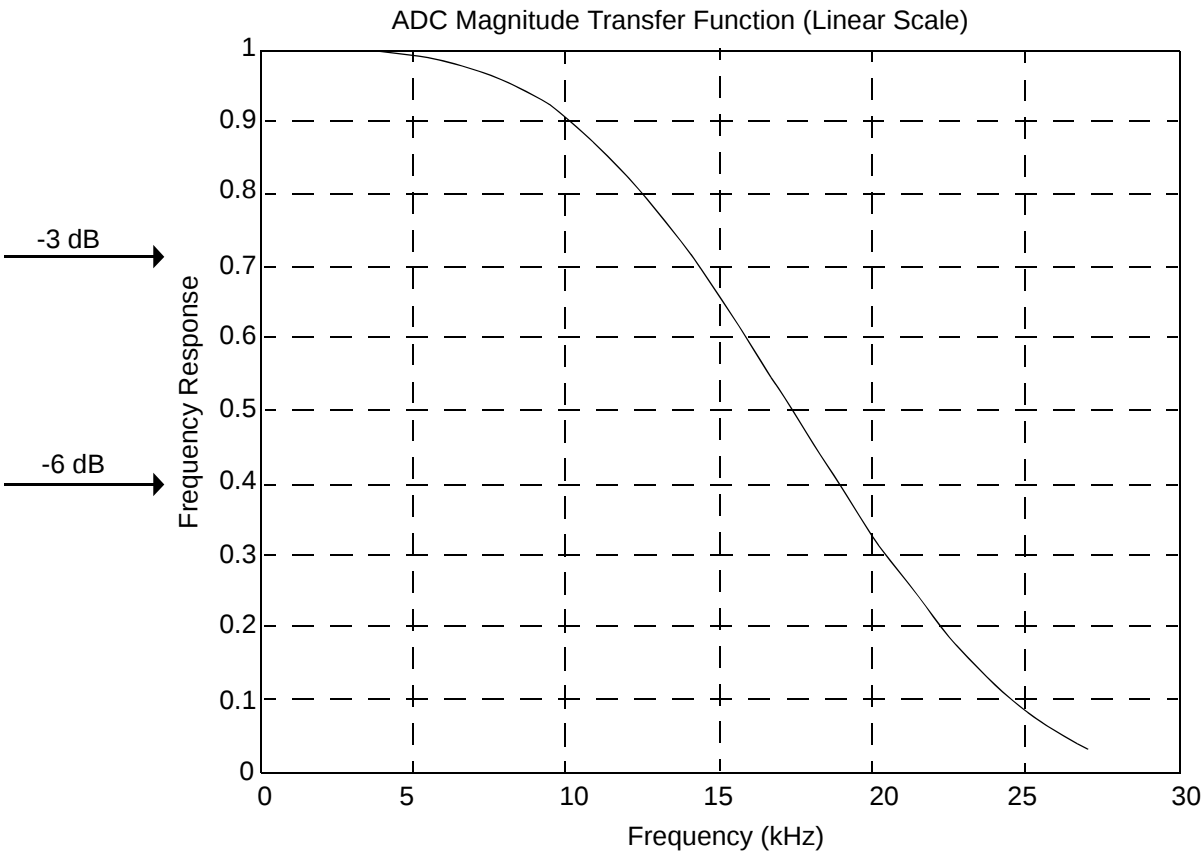


Figure 47. Analog-to-Digital Converter Frequency Response

Table 104. Analog-to-Digital Converter Electrical Characteristics and Timing

Symbol	Parameter	V _{DD} = 3.0–3.6 V T _A = -40 °C to 105 °C			Units	Conditions
		Minimum	Typical	Maximum		
	Resolution	10	–	–	bits	External V _{REF} = 3.0 V;
	Differential Nonlinearity (DNL)	-0.25	–	0.25	lsb	Guaranteed by design
	Integral Nonlinearity (INL)	-2.0	–	2.0	lsb	External V _{REF} = 3.0 V
	DC Offset Error	-35	–	25	mV	
V _{REF}	Internal Reference Voltage	1.9	2.0	2.4	V	V _{DD} = 3.0 - 3.6 V 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 AVDD.
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	W	Recommended
Z _{in}	Input Impedance		150		KΩ	
V _{REF}	External Reference Voltage			AVDD	V	AVDD ≤ VDD. When using an external reference voltage, decoupling capacitance should be placed from VREF to AVSS.
I _{REF}	Current draw into VREF 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 105 lists the GPIO port input timing.

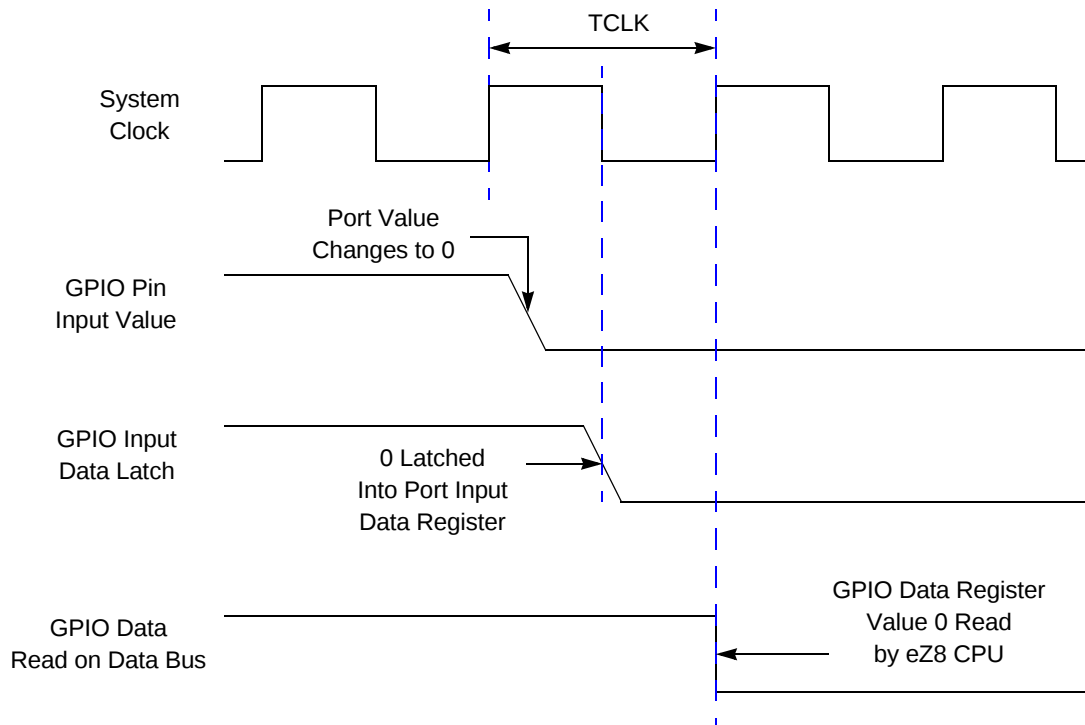


Figure 48. Port Input Sample Timing

Table 105. GPIO Port Input Timing

Parameter	Abbreviation	Delay (ns)	
		Minimum	Maximum
T _{S_PORT}	Port Input Transition to XIN Fall Setup Time (Not pictured)	5	–
T _{H_PORT}	XIN 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 106 provide timing information for GPIO Port pins.

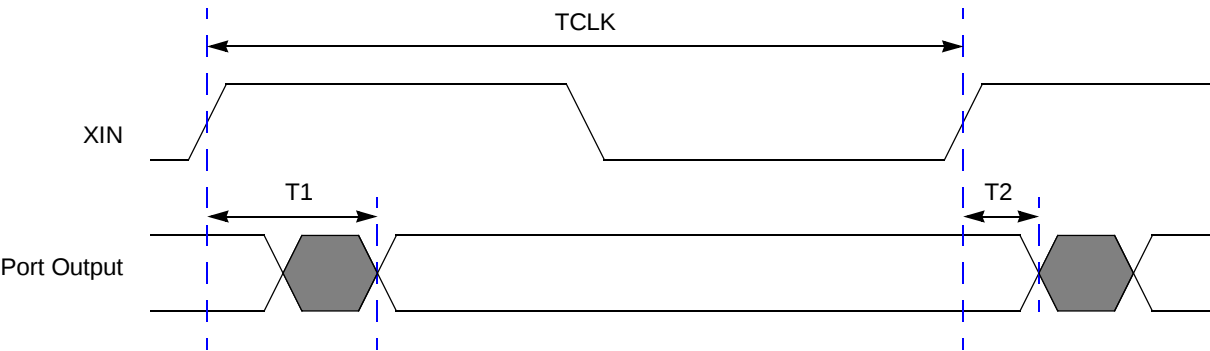


Figure 49. GPIO Port Output Timing

Table 106. GPIO Port Output Timing

Parameter Abbreviation		Delay (ns)	
		Minimum	Maximum
GPIO Port pins			
T ₁	XIN Rise to Port Output Valid Delay	–	15
T ₂	XIN Rise to Port Output Hold Time	2	–

On-Chip Debugger Timing

Figure 50 and Table 107 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4 μ s maximum rise and fall time.

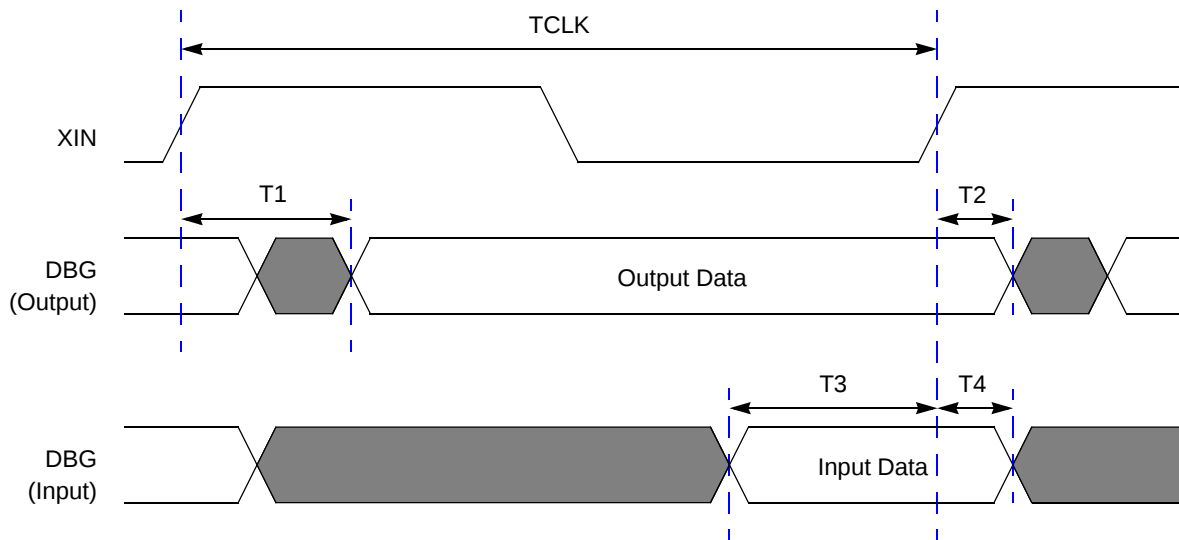


Figure 50. On-Chip Debugger Timing

Table 107. On-Chip Debugger Timing

Parameter Abbreviation		Delay (ns)	
		Minimum	Maximum
DBG			
T ₁	XIN Rise to DBG Valid Delay	–	15
T ₂	XIN Rise to DBG Output Hold Time	2	–
T ₃	DBG to XIN Rise Input Setup Time	10	–
T ₄	DBG to XIN Rise Input Hold Time	5	–
	DBG frequency		System Clock/4

SPI MASTER Mode Timing

Figure 51 and Table 108 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.

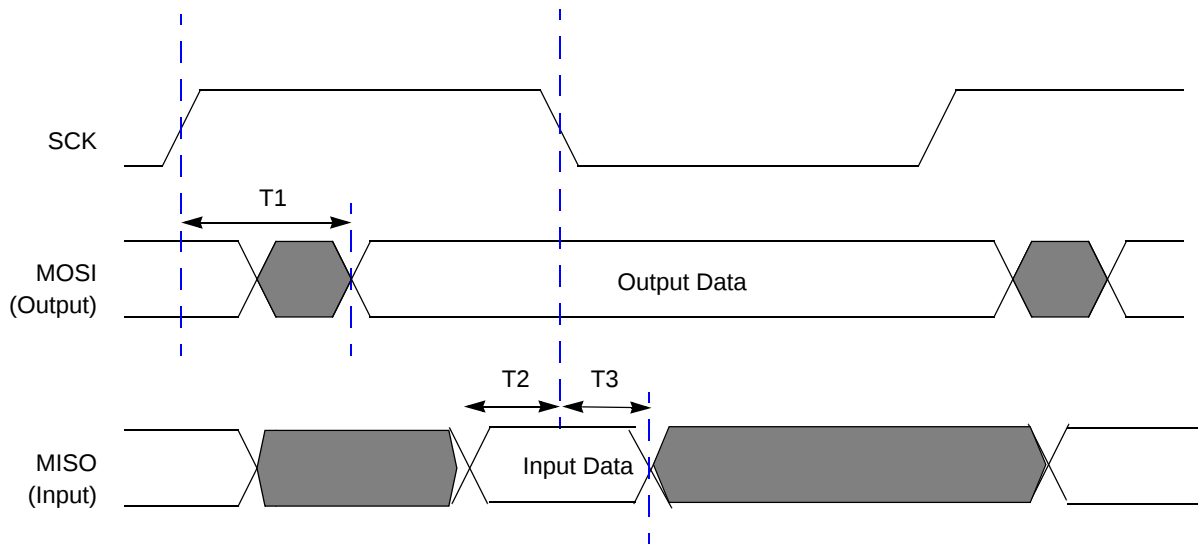


Figure 51. SPI MASTER Mode Timing

Table 108. SPI MASTER Mode Timing

Parameter	Abbreviation	Delay (ns)	
		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 109 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.

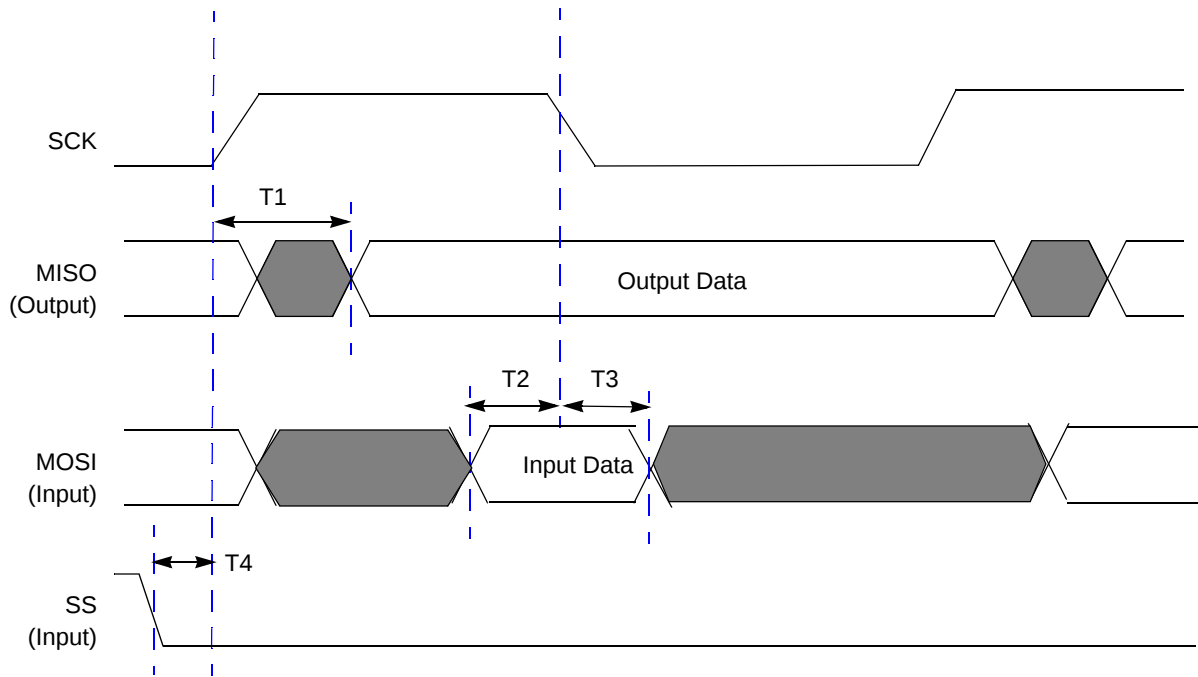


Figure 52. SPI SLAVE Mode Timing

Table 109. SPI SLAVE Mode Timing

Parameter	Abbreviation	Delay (ns)	
		Minimum	Maximum
SPI SLAVE			
T ₁	SCK (transmit edge) to MISO output Valid Delay	2 * Xin period	3 * Xin period + 20 nsec
T ₂	MOSI input to SCK (receive edge) Setup Time	0	
T ₃	MOSI input to SCK (receive edge) Hold Time	3 * Xin period	
T ₄	SS input assertion to SCK setup	1 * Xin period	

I²C Timing

Figure 53 and Table 110 provide timing information for I²C pins.

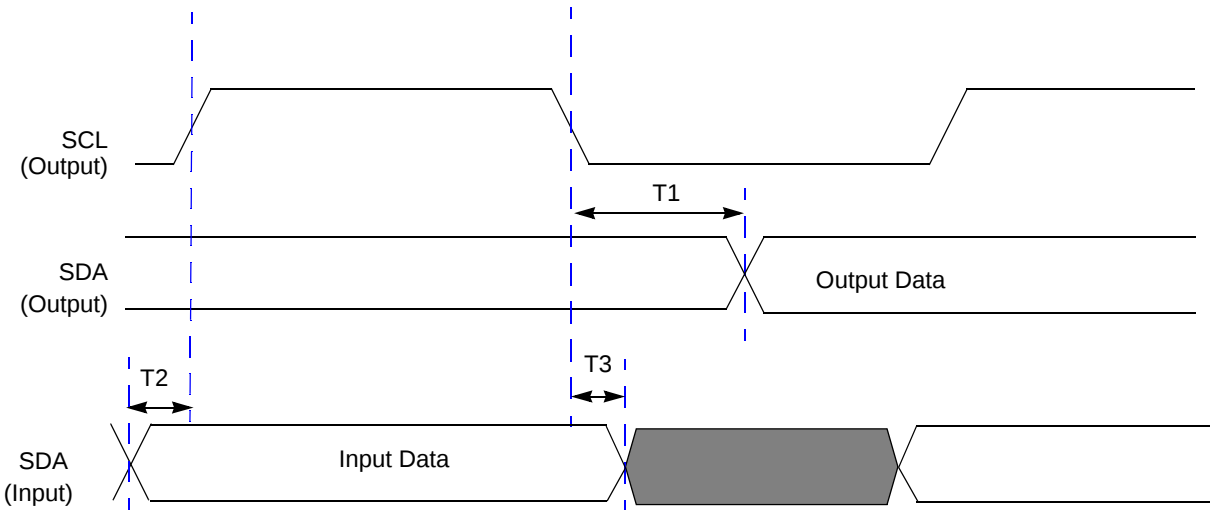


Figure 53. I²C Timing

Table 110. I²C Timing

Parameter	Abbreviation	Delay (ns)	
		Minimum	Maximum
I ² C			
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 111 provide timing information for UART pins for the case where the Clear To Send input pin ($\overline{\text{CTS}}$) 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 $\overline{\text{DE}}$. The $\overline{\text{CTS}}$ to $\overline{\text{DE}}$ assertion delay (T_1) assumes the UART Transmit Data Register has been loaded with data prior to $\overline{\text{CTS}}$ assertion.

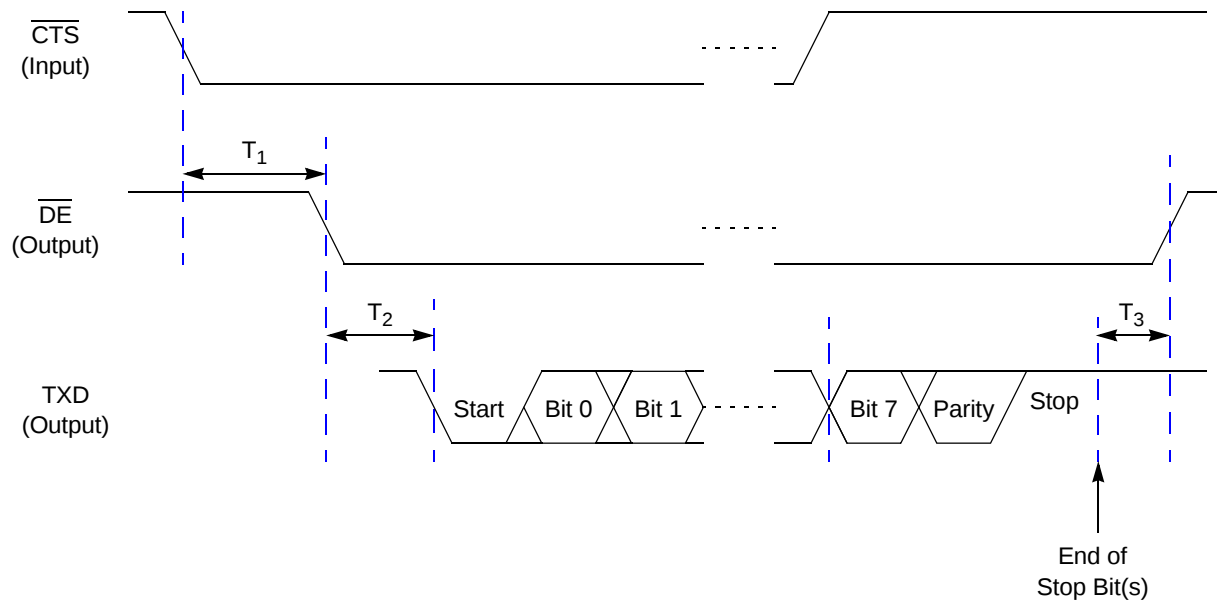


Figure 54. UART Timing with $\overline{\text{CTS}}$

Table 111. UART Timing with $\overline{\text{CTS}}$

Parameter	Abbreviation	Delay (ns)	
		Minimum	Maximum
T_1	$\overline{\text{CTS}}$ Fall to $\overline{\text{DE}}$ Assertion Delay	2 * XIN period	2 * XIN period + 1 Bit period
T_2	$\overline{\text{DE}}$ Assertion to TXD Falling Edge (Start) Delay	1 Bit period	1 Bit period + 1 * XIN period
T_3	End of Stop Bit(s) to $\overline{\text{DE}}$ Deassertion Delay	1 * XIN period	2 * XIN period

Figure 55 and Table 112 provide timing information for UART pins for the case where the Clear To Send input signal ($\overline{\text{CTS}}$) 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 represented here by $\overline{\text{DE}}$. $\overline{\text{DE}}$ asserts after the UART Transmit Data Register has been written. $\overline{\text{DE}}$ remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

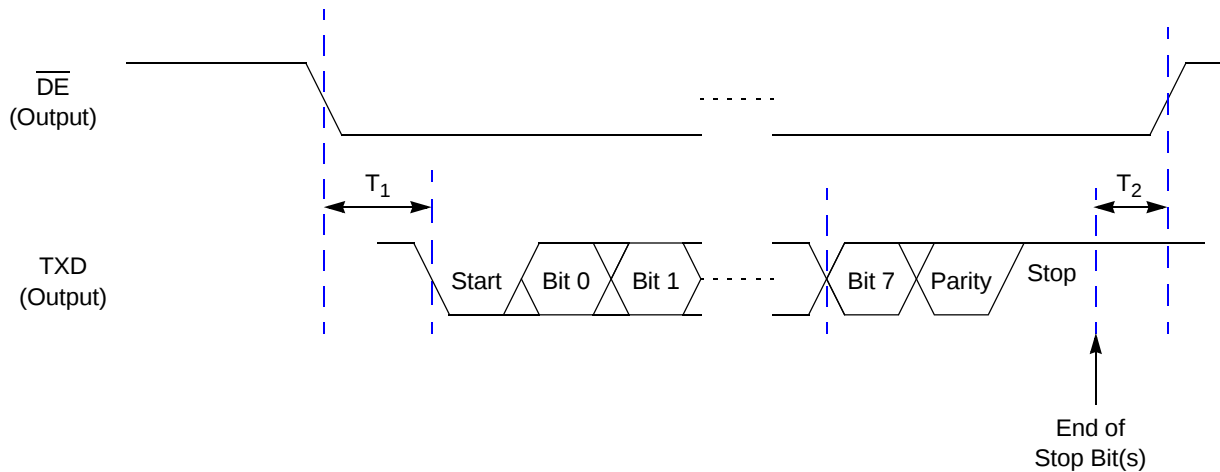


Figure 55. UART Timing without $\overline{\text{CTS}}$

Table 112. UART Timing without $\overline{\text{CTS}}$

Parameter	Abbreviation	Delay (ns)	
		Minimum	Maximum
T_1	$\overline{\text{DE}}$ Assertion to TXD Falling Edge (Start) Delay	1 Bit period	1 Bit period + 1 * XIN period
T_2	End of Stop Bit(s) to $\overline{\text{DE}}$ Deassertion Delay	1 * XIN period	2 * XIN period



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 (opcodes and operands) to represent the instructions themselves. The opcodes 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 opcode-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 is:

Table 113. Assembly Language Syntax Example 1

Assembly Language Code	ADD	43H	08H	(ADD dst, src)
Object Code	04	08	43	(OPC src, dst)

Example 2: In general, when an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or, using Escaped Mode Addressing, a Working Register R0–R15. If the contents of Register 43H and Working Register R8 are added and the result is stored in 43H, the assembly syntax and resulting object code is:

Table 114. Assembly Language Syntax Example 2

Assembly Language Code	ADD	43H,	R8	(ADD dst, src)
Object Code	04	E8	43	(OPC src, dst)

See the device-specific Product Specification to determine the exact register file range available. The register file size varies, depending on the device type.

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 115](#) on page 211.

Table 115. Notational Shorthand

Notation	Description	Operand	Range
b	Bit	b	b represents a value from 0 to 7 (000B to 111B).
cc	Condition Code	—	See Condition Codes overview in the eZ8 CPU User Manual.
DA	Direct Address	Addr	Addr. 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
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 116 contains additional symbols that are used throughout the Instruction Summary and Instruction Set Description sections.

Table 116. 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. For example,

$$\text{dst} \leftarrow \text{dst} + \text{src}$$

indicates the source data is added to the destination data and 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 117](#). 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 117. Condition Codes

Binary	Hex	Assembly Mnemonic	Definition	Flag Test Operation
0000	0	F	Always False	–
0001	1	LT	Less Than	$(S \text{ XOR } V) = 1$
0010	2	LE	Less Than or Equal	$(Z \text{ OR } (S \text{ XOR } V)) = 1$
0011	3	ULE	Unsigned Less Than or Equal	$(C \text{ OR } Z) = 1$
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	$(S \text{ XOR } V) = 0$
1010	A	GT	Greater Than	$(Z \text{ OR } (S \text{ XOR } V)) = 0$
1011	B	UGT	Unsigned Greater Than	$(C = 0 \text{ AND } Z = 0) = 1$
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 118 through Table 125 on page 218 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 118. 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
MULT	dst	Multiply

Table 118. Arithmetic Instructions (Continued)

Mnemonic	Operands	Instruction
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 119. 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 120. 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 121. 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 122. 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
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 123. 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 124. 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 125. 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 126](#) 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 126. eZ8 CPU Instruction Summary

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
ADC dst, src	$\text{dst} \leftarrow \text{dst} + \text{src} + \text{C}$	r	r	12	*	*	*	*	0	*	2	3
		r	lr	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	$\text{dst} \leftarrow \text{dst} + \text{src} + \text{C}$	ER	ER	18	*	*	*	*	0	*	4	3
		ER	IM	19							4	3

Table 126. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
ADD dst, src	$\text{dst} \leftarrow \text{dst} + \text{src}$	r	r	02	*	*	*	*	0	*	2	3
		r	lr	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	$\text{dst} \leftarrow \text{dst} + \text{src}$	ER	ER	08	*	*	*	*	0	*	4	3
		ER	IM	09							4	3
AND dst, src	$\text{dst} \leftarrow \text{dst} \text{ AND } \text{src}$	r	r	52	-	*	*	0	-	-	2	3
		r	lr	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	$\text{dst} \leftarrow \text{dst} \text{ AND } \text{src}$	ER	ER	58	-	*	*	0	-	-	4	3
		ER	IM	59							4	3
BCLR bit, dst	$\text{dst}[\text{bit}] \leftarrow 0$	r		E2	-	-	-	-	-	-	2	2
BIT p, bit, dst	$\text{dst}[\text{bit}] \leftarrow \text{p}$	r		E2	-	-	-	-	-	-	2	2
BRK	Debugger Break			00	-	-	-	-	-	-	1	1
BSET bit, dst	$\text{dst}[\text{bit}] \leftarrow 1$	r		E2	-	-	-	-	-	-	2	2
BSWAP dst	$\text{dst}[7:0] \leftarrow \text{dst}[0:7]$	R		D5	X	*	*	0	-	-	2	2
BTJ p, bit, src, dst	if $\text{src}[\text{bit}] = \text{p}$ $\text{PC} \leftarrow \text{PC} + \text{X}$		r	F6	-	-	-	-	-	-	3	3
			lr	F7							3	4
BTJNZ bit, src, dst	if $\text{src}[\text{bit}] = 1$ $\text{PC} \leftarrow \text{PC} + \text{X}$		r	F6	-	-	-	-	-	-	3	3
			lr	F7							3	4
BTJZ bit, src, dst	if $\text{src}[\text{bit}] = 0$ $\text{PC} \leftarrow \text{PC} + \text{X}$		r	F6	-	-	-	-	-	-	3	3
			lr	F7							3	4

Table 126. eZ8 CPU Instruction Summary (Continued)

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

Table 126. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
DECW dst	$\text{dst} \leftarrow \text{dst} - 1$	RR		80	-	*	*	*	-	-	2	5
		IRR		81							2	6
DI	$\text{IRQCTL}[7] \leftarrow 0$			8F	-	-	-	-	-	-	1	2
DJNZ dst, RA	$\text{dst} \leftarrow \text{dst} - 1$ if $\text{dst} \neq 0$ $\text{PC} \leftarrow \text{PC} + \text{X}$	r		0A-FA	-	-	-	-	-	-	2	3
EI	$\text{IRQCTL}[7] \leftarrow 1$			9F	-	-	-	-	-	-	1	2
HALT	HALT Mode			7F	-	-	-	-	-	-	1	2
INC dst	$\text{dst} \leftarrow \text{dst} + 1$	R		20	-	*	*	*	-	-	2	2
		IR		21							2	3
		r		0E-FE							1	2
INCW dst	$\text{dst} \leftarrow \text{dst} + 1$	RR		A0	-	*	*	*	-	-	2	5
		IRR		A1							2	6
IRET	$\text{FLAGS} \leftarrow @\text{SP}$ $\text{SP} \leftarrow \text{SP} + 1$ $\text{PC} \leftarrow @\text{SP}$ $\text{SP} \leftarrow \text{SP} + 2$ $\text{IRQCTL}[7] \leftarrow 1$			BF	*	*	*	*	*	*	1	5
JP dst	$\text{PC} \leftarrow \text{dst}$	DA		8D	-	-	-	-	-	-	3	2
		IRR		C4							2	3
JP cc, dst	if cc is true $\text{PC} \leftarrow \text{dst}$	DA		0D-FD	-	-	-	-	-	-	3	2
JR dst	$\text{PC} \leftarrow \text{PC} + \text{X}$	DA		8B	-	-	-	-	-	-	2	2
JR cc, dst	if cc is true $\text{PC} \leftarrow \text{PC} + \text{X}$	DA		0B-FB	-	-	-	-	-	-	2	2



Table 126. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
LD dst, rc	$\text{dst} \leftarrow \text{src}$	r	IM	0C-FC	-	-	-	-	-	-	2	2
		r	X(r)	C7							3	3
		X(r)	r	D7							3	4
		r	lr	E3							2	3
		R	R	E4							3	2
		R	IR	E5							3	4
		R	IM	E6							3	2
		IR	IM	E7							3	3
		lr	r	F3							2	3
		IR	R	F5							3	3
LDC dst, src	$\text{dst} \leftarrow \text{src}$	r	lrr	C2	-	-	-	-	-	-	2	5
		lr	lrr	C5							2	9
		lrr	r	D2							2	5
LDCI dst, src	$\text{dst} \leftarrow \text{src}$ $r \leftarrow r + 1$ $rr \leftarrow rr + 1$	lr	lrr	C3	-	-	-	-	-	-	2	9
		lrr	lr	D3							2	9
LDE dst, src	$\text{dst} \leftarrow \text{src}$	r	lrr	82	-	-	-	-	-	-	2	5
		lrr	r	92							2	5
LDEI dst, src	$\text{dst} \leftarrow \text{src}$ $r \leftarrow r + 1$ $rr \leftarrow rr + 1$	lr	lrr	83	-	-	-	-	-	-	2	9
		lrr	lr	93							2	9

Table 126. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
LDX dst, src	$\text{dst} \leftarrow \text{src}$	r	ER	84	-	-	-	-	-	-	3	2
		lr	ER	85							3	3
		R	IRR	86							3	4
		IR	IRR	87							3	5
		r	X(rr)	88							3	4
		X(rr)	r	89							3	4
		ER	r	94							3	2
		ER	lr	95							3	3
		IRR	R	96							3	4
		IRR	IR	97							3	5
		ER	ER	E8							4	2
		ER	IM	E9							4	2
LEA dst, X(src)	$\text{dst} \leftarrow \text{src} + \text{X}$	r	X(r)	98	-	-	-	-	-	-	3	3
		rr	X(rr)	99							3	5
MULT dst	$\text{dst}[15:0] \leftarrow \text{dst}[15:8] * \text{dst}[7:0]$	RR		F4	-	-	-	-	-	-	2	8
NOP	No operation			0F	-	-	-	-	-	-	1	2
OR dst, src	$\text{dst} \leftarrow \text{dst OR src}$	r	r	42	-	*	*	0	-	-	2	3
		r	lr	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	$\text{dst} \leftarrow \text{dst OR src}$	ER	ER	48	-	*	*	0	-	-	4	3
		ER	IM	49							4	3
POP dst	$\text{dst} \leftarrow @\text{SP}$ $\text{SP} \leftarrow \text{SP} + 1$	R		50	-	-	-	-	-	-	2	2
		IR		51							2	3

Table 126. eZ8 CPU Instruction Summary (Continued)

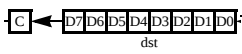
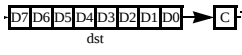
Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
POPX dst	dst ← @SP SP ← SP + 1	ER		D8	-	-	-	-	-	-	3	2
PUSH src	SP ← SP - 1 @SP ← src	R		70	-	-	-	-	-	-	2	2
		IR		71							2	3
PUSHX src	SP ← SP - 1 @SP ← src	ER		C8	-	-	-	-	-	-	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	lr	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

Table 126. eZ8 CPU Instruction Summary (Continued)

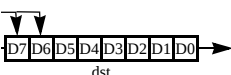
Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
SCF	$C \leftarrow 1$			DF	1	-	-	-	-	-	1	2
SRA dst		R		D0	*	*	*	0	-	-	2	2
		IR		D1							2	3
SRL dst		R		1F C0	*	*	0	*	-	-	3	2
		IR		1F C1							3	3
SRP src	$RP \leftarrow src$		IM	01	-	-	-	-	-	-	2	2
STOP	STOP Mode			6F	-	-	-	-	-	-	1	2
SUB dst, src	$dst \leftarrow dst - src$	r	r	22	*	*	*	*	1	*	2	3
		r	lr	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	$dst \leftarrow dst - src$	ER	ER	28	*	*	*	*	1	*	4	3
		ER	IM	29							4	3
SWAP dst	$dst[7:4] \leftrightarrow dst[3:0]$	R		F0	X	*	*	X	-	-	2	2
		IR		F1							2	3
TCM dst, src	(NOT dst) AND src	r	r	62	-	*	*	0	-	-	2	3
		r	lr	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

Table 126. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
TM dst, src	dst AND src	r	r	72	-	*	*	0	-	-	2	3
		r	lr	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	lr	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
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.

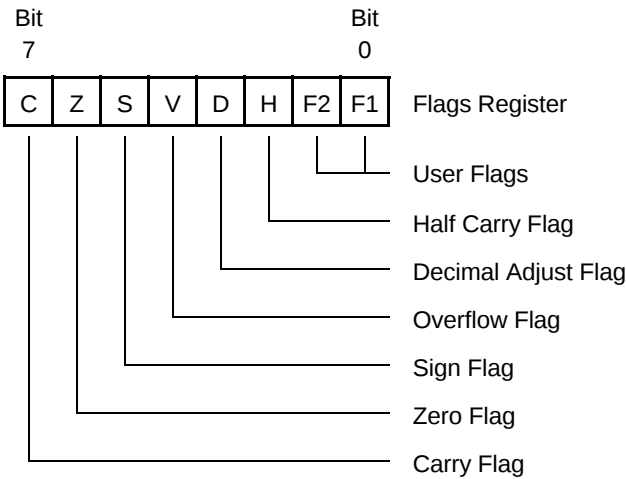


Figure 56. Flags Register

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.



Opcode Maps

A description of the opcode map data and the abbreviations are provided in [Figure 57](#) and [Table 127](#) on page 230. [Figure 58](#) on page 231 and [Figure 59](#) on page 232 provide information on each of the eZ8 CPU instructions.

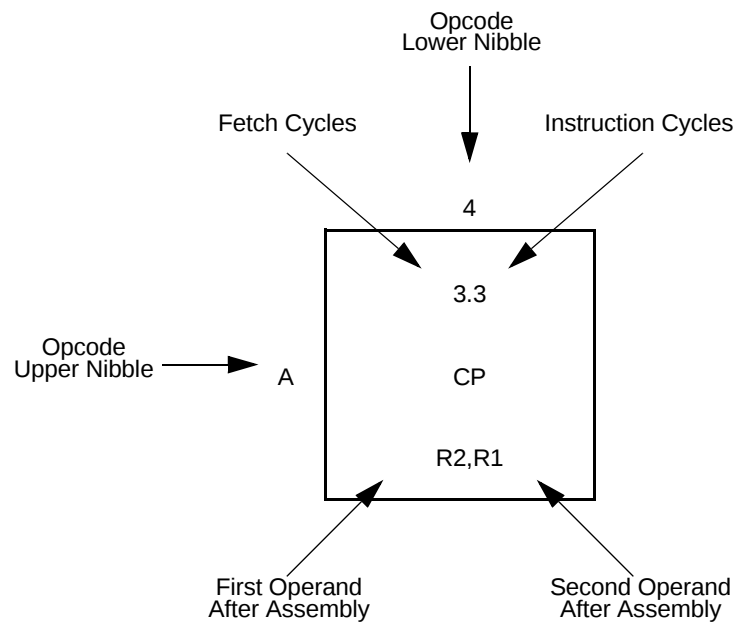


Figure 57. Opcode Map Cell Description

Table 127. Opcode 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

		Lower Nibble (Hex)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Upper Nibble (Hex)	0	1.2 BRK	2.2 SRP	2.3 ADD	2.4 ADD	3.3 ADD	3.4 ADD	3.3 ADD	3.4 ADD	4.3 ADDX	4.3 ADDX	2.3 DJNZ	2.2 JR	2.2 LD	3.2 JP	1.2 INC	1.2 NOP
	1	2.2 RLC	2.3 RLC	2.3 ADC	2.4 ADC	3.3 ADC	3.4 ADC	3.3 ADC	3.4 ADC	4.3 ADCX	4.3 ADCX						See 2nd Opcode Map
	2	2.2 INC	2.3 INC	2.3 SUB	2.4 SUB	3.3 SUB	3.4 SUB	3.3 SUB	3.4 SUB	4.3 SUBX	4.3 SUBX						
	3	2.2 DEC	2.3 DEC	2.3 SBC	2.4 SBC	3.3 SBC	3.4 SBC	3.3 SBC	3.4 SBC	4.3 SBCX	4.3 SBCX						
	4	2.2 DA	2.3 DA	2.3 OR	2.4 OR	3.3 OR	3.4 OR	3.3 OR	3.4 OR	4.3 ORX	4.3 ORX						
	5	2.2 POP	2.3 POP	2.3 AND	2.4 AND	3.3 AND	3.4 AND	3.3 AND	3.4 AND	4.3 ANDX	4.3 ANDX						1.2 WDT
	6	2.2 COM	2.3 COM	2.3 TCM	2.4 TCM	3.3 TCM	3.4 TCM	3.3 TCM	3.4 TCM	4.3 TCMX	4.3 TCMX						1.2 STOP
	7	2.2 PUSH	2.3 PUSH	2.3 TM	2.4 TM	3.3 TM	3.4 TM	3.3 TM	3.4 TM	4.3 TMX	4.3 TMX						1.2 HALT
	8	2.5 DECW	2.6 DECW	2.5 LDE	2.9 LDEI	3.2 LDX	3.3 LDX	3.4 LDX	3.5 LDX	3.4 LDX	3.4 LDX						1.2 DI
	9	2.2 RL	2.3 RL	2.5 LDE	2.9 LDEI	3.2 LDX	3.3 LDX	3.4 LDX	3.5 LDX	3.3 LEA	3.5 LEA						1.2 EI
	A	2.5 INCW	2.6 INCW	2.3 CP	2.4 CP	3.3 CP	3.4 CP	3.3 CP	3.4 CP	4.3 CPX	4.3 CPX						1.4 RET
	B	2.2 CLR	2.3 CLR	2.3 XOR	2.4 XOR	3.3 XOR	3.4 XOR	3.3 XOR	3.4 XOR	4.3 XORX	4.3 XORX						1.5 IRET
	C	2.2 RRC	2.3 RRC	2.5 LDC	2.9 LDCI	2.3 JP	2.9 LDC		3.4 LD	3.2 PUSHX							1.2 RCF
	D	2.2 SRA	2.3 SRA	2.5 LDC	2.9 LDCI	2.6 CALL	2.2 BSWAP	3.3 CALL	3.4 LD	3.2 POPX							1.2 SCF
	E	2.2 RR	2.3 RR	2.2 BIT	2.3 LD	3.2 LD	3.3 LD	3.2 LD	3.3 LD	4.2 LDX	4.2 LDX						1.2 CCF
	F	2.2 SWAP	2.3 SWAP	2.6 TRAP	2.3 LD	2.8 MULT	3.3 LD	3.3 BTJ	3.4 BTJ								

Figure 58. First Opcode Map



		Lower Nibble (Hex)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Upper Nibble (Hex)	0																
	1																
	2																
	3																
	4																
	5																
	6																
	7																
	8																
	9																
	A			3.3 CPC r1,r2	3.4 CPC r1,lr2	4.3 CPC R2,R1	4.4 CPC IR2,R1	4.3 CPC R1,IM	4.4 CPC IR1,IM	5.3 CPCX ER2,ER1	5.3 CPCX IM,ER1						
	B																
	C	3.2 SRL R1	3.3 SRL IR1														
	D																
	E																
	F																

Figure 59. Second Opcode Map after 1FH

Packaging

Figure 60 displays the 20-pin SSOP package available for Z8 Encore! XP[®] F0822 Series devices.

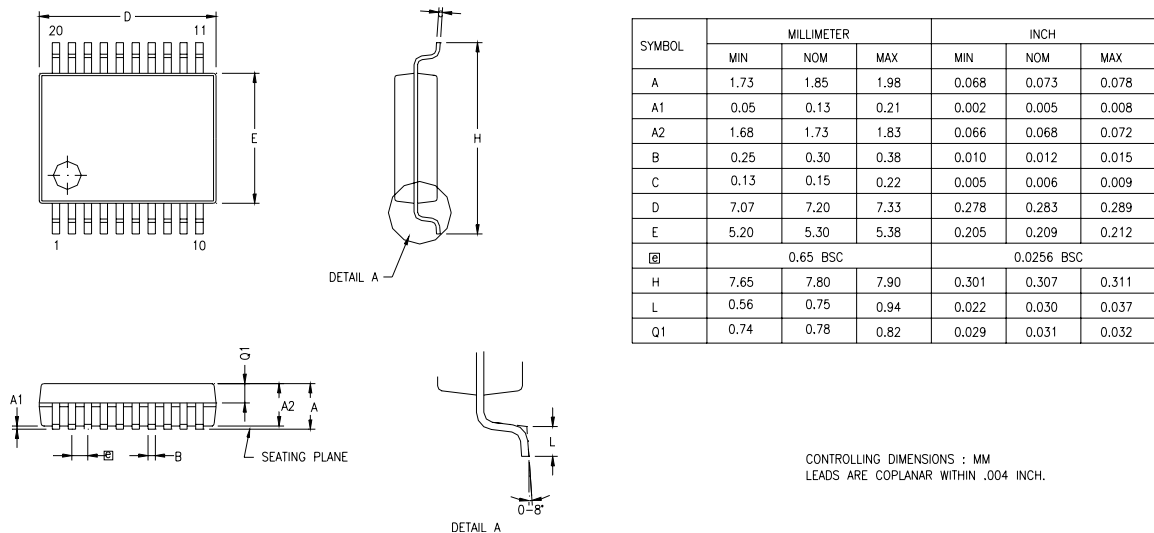


Figure 60. 20-Pin Small Shrink Outline Package (SSOP)

Figure 61 displays the 20-pin PDIP package available for Z8 Encore! XP F0822 Series devices.

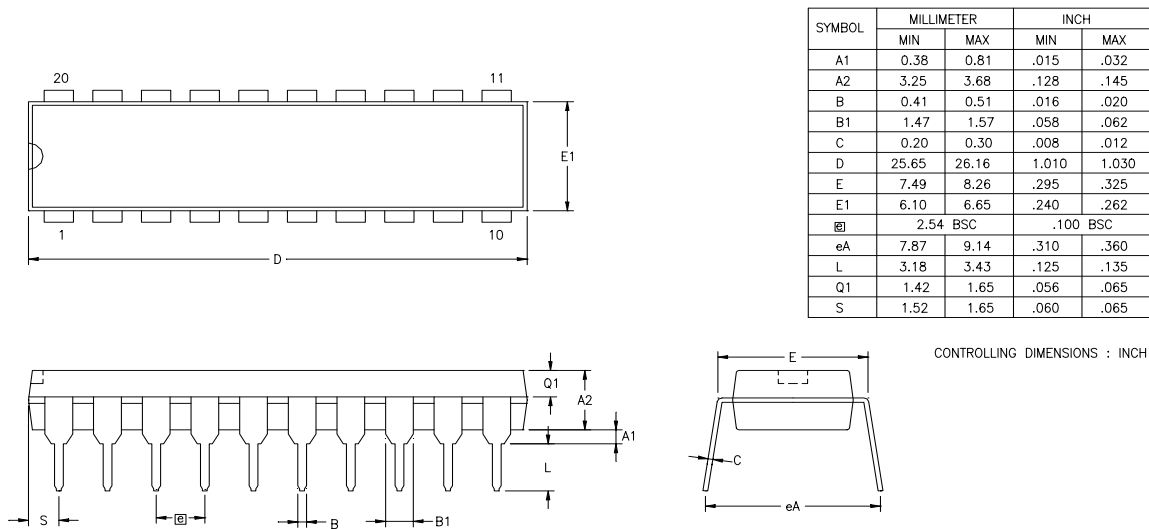


Figure 61. 20-Pin Plastic Dual-Inline Package (PDIP)

Figure 62 displays the 28-pin SOIC package available for Z8 Encore! XP F0822 Series devices.

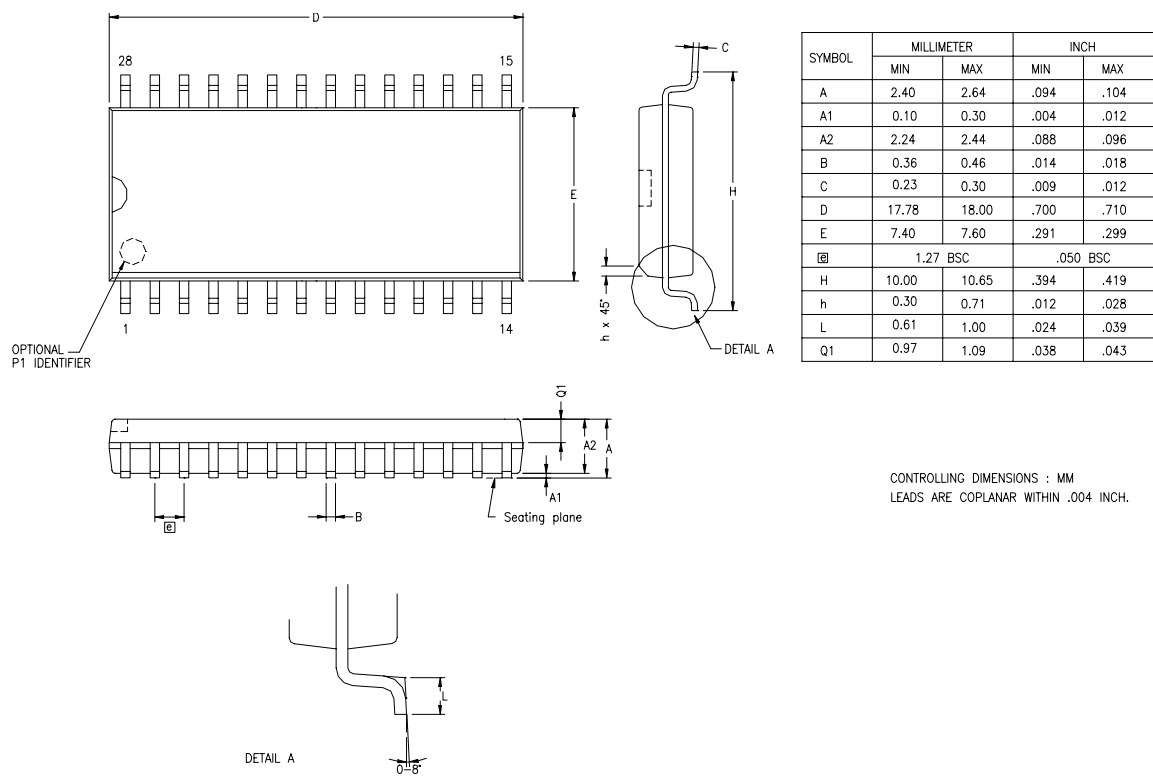


Figure 62. 28-Pin Small Outline Integrated Circuit Package (SOIC)

Figure 63 displays the 28-pin PDIP package available for Z8 Encore! XP F0822 Series devices.

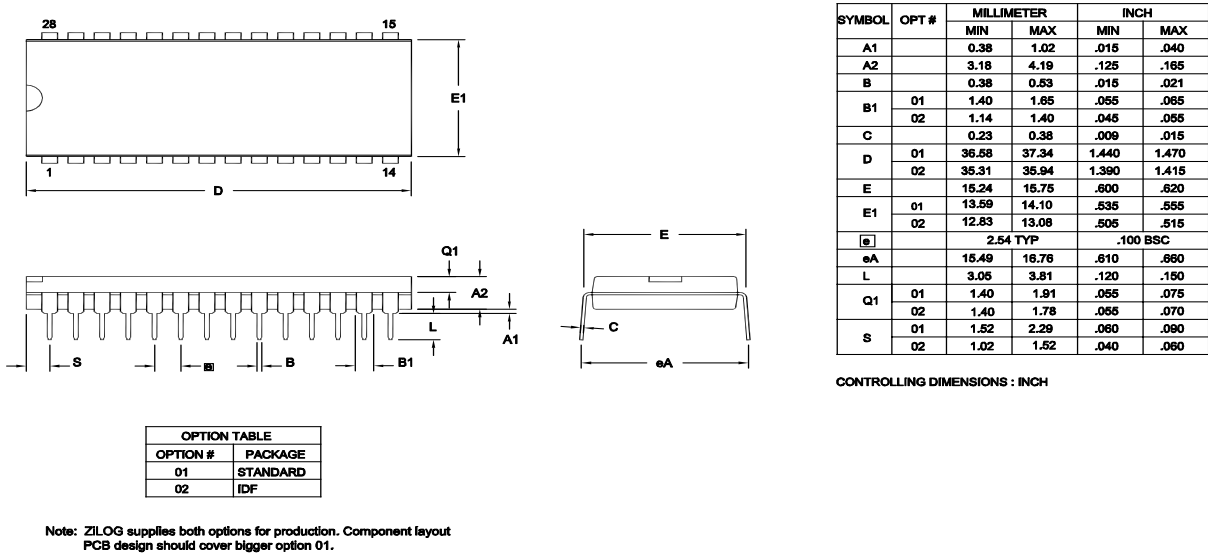


Figure 63. 28-Pin Plastic Dual-Inline Package (PDIP)



Ordering Information

Order Z8 Encore! XP F0822 Series from Zilog[®], using the following part numbers. For more information regarding ordering, consult your local Zilog sales office. Zilog website at www.zilog.com lists all regional offices and provides additional Z8 Encore! XP product information.

Part Number	Flash	RAM	I/O Lines	Interrupts	16-Bit Timers w/PWM	10-Bit A/D Channels	I ² C	SPI	UARTs with IrDA	Description
Z8F08xx with 8 KB Flash, 10-Bit Analog-to-Digital Converter										
Standard Temperature: 0 °C to 70 °C										
Z8F0821HH020SC	8 KB	1 KB	11	16	2	2	1	0	1	SSOP 20-pin package
Z8F0821PH020SC	8 KB	1 KB	11	16	2	2	1	0	1	PDIP 20-pin package
Z8F0822SJ020SC	8 KB	1 KB	19	19	2	5	1	1	1	SOIC 28-pin package
Z8F0822PJ020SC	8 KB	1 KB	19	19	2	5	1	1	1	PDIP 28-pin package
Extended Temperature: -40° to +105°C										
Z8F0821HH020EC	8 KB	1 KB	11	16	2	2	1	0	1	SSOP 20-pin package
Z8F0821PH020EC	8 KB	1 KB	11	16	2	2	1	0	1	PDIP 20-pin package
Z8F0822SJ020EC	8 KB	1 KB	19	19	2	5	1	1	1	SOIC 28-pin package
Z8F0822PJ020EC	8 KB	1 KB	19	19	2	5	1	1	1	PDIP 28-pin package

Part Number	Flash	RAM	I/O Lines	Interrupts	16-Bit Timers w/PWM	10-Bit A/D Channels	I ² C	SPI	UARTs with IrDA	Description
Z8F08xx with 8 KB Flash										
Standard Temperature: 0 °C to 70 °C										
Z8F0811HH020SC	8 KB	1 KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0811PH020SC	8 KB	1 KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0812SJ020SC	8 KB	1 KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0812PJ020SC	8 KB	1 KB	19	19	2	0	1	1	1	PDIP 28-pin package
Extended Temperature: -40 °C to +105 °C										
Z8F0811HH020EC	8 KB	1 KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0811PH020EC	8 KB	1 KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0812SJ020EC	8 KB	1 KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0812PJ020EC	8 KB	1 KB	19	19	2	0	1	1	1	PDIP 28-pin package



Part Number	Flash	RAM	I/O Lines	Interrupts	16-Bit Timers w/PWM	10-Bit A/D Channels	I ² C	SPI	UARTs with IrDA	Description
Z8F04xx with 4 KB Flash, 10-Bit Analog-to-Digital Converter										
Standard Temperature: 0 °C to 70 °C										
Z8F0421HH020SC	4 KB	1 KB	11	16	2	2	1	0	1	SSOP 20-pin package
Z8F0421PH020SC	4 KB	1 KB	11	16	2	2	1	0	1	PDIP 20-pin package
Z8F0422SJ020SC	4 KB	1 KB	19	19	2	5	1	1	1	SOIC 28-pin package
Z8F0422PJ020SC	4 KB	1 KB	19	19	2	5	1	1	1	PDIP 28-pin package
Extended Temperature: -40 °C to 105 °C										
Z8F0421HH020EC	4 KB	1 KB	11	16	2	2	1	0	1	SSOP 20-pin package
Z8F0421PH020EC	4 KB	1 KB	11	16	2	2	1	0	1	PDIP 20-pin package
Z8F0422SJ020EC	4 KB	1 KB	19	19	2	5	1	1	1	SOIC 28-pin package
Z8F0422PJ020EC	4 KB	1 KB	19	19	2	5	1	1	1	PDIP 28-pin package

Part Number	Flash	RAM	I/O Lines	Interrupts	16-Bit Timers w/PWM	10-Bit A/D Channels	I ² C	SPI	UARTs with IrDA	Description
Z8F04xx with 4 KB Flash										
Standard Temperature: 0 °C to 70 °C										
Z8F0411HH020SC	4 KB	1 KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0411PH020SC	4 KB	1 KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0412SJ020SC	4 KB	1 KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0412PJ020SC	4 KB	1 KB	19	19	2	0	1	1	1	PDIP 28-pin package
Extended Temperature: -40 °C to 105 °C										
Z8F0411HH020EC	4 KB	1 KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0411PH020EC	4 KB	1 KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0412SJ020EC	4 KB	1 KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0412PJ020EC	4 KB	1 KB	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
Note: Replace C with G for lead-free packaging.										

Part Number Suffix Designations

Z8	F	08	21	H	H	020	S	C	
									Environmental Flow
									C = Plastic Standard
									G = Lead-Free Package
									Temperature Range (°C)
									S = Standard, 0 to 70
									E = Extended, -40 to +105
									Speed
									020 = 20 MHz
									Pin Count
									H = 20
									J = 28
									Package
									H = SSOP
									P = PDIP
									S = SOIC
									Device Type
									22 = 19 I/O lines, 5 ADC channels, one SPI
									21 = 11 I/O lines, 2 ADC channels, no SPI
									12 = 19 I/O lines, no ADC channels, one SPI
									11 = 11 I/O lines, no ADC channels, no SPI
									Memory Size
									08 = 8 KB Flash, 1 KB RAM
									04 = 4 KB Flash, 1 KB RAM
									Memory Type
									F = Flash
									Device Family
									Z8 = Zilog's 8-Bit Microcontroller

For example, part number Z8F0821HH020SC is a Z8 Encore! XP Flash 8 KB microcontroller in a 20-pin SSOP package, operating with a maximum 20 MHz external clock frequency over a 0 °C to +70 °C temperature range and built using the Plastic-Standard environmental flow.

Index

Symbols

212
% 212
@ 212

Numerics

10-bit ADC 4
40-lead plastic dual-inline package 234

A

absolute maximum ratings 185
AC characteristics 194
ADC 214
 architecture 147
 automatic power-down 148
 block diagram 147
 continuous conversion 148
 control register 150
 control register definitions 150
 data high byte register 151
 data low bits register 151
 electrical characteristics and timing 199
 operation 148
 single-shot conversion 148
ADCCTL register 150
ADCDH register 151
ADCDL register 151
ADCX 214
ADD 214
additional symbols 212
address space 13
ADDX 214
analog signals 10
analog-to-digital converter (ADC) 147
AND 217
ANDX 217
arithmetic instructions 214
assembly language programming 209

assembly language syntax 210

B

B 212
b 211
baud rate generator, UART 99
BCLR 215
binary number suffix 212
BIT 215
bit 211
 clear 215
 manipulation instructions 215
 set 215
 set or clear 215
 swap 215, 218
 test and jump 217
 test and jump if non-zero 217
 test and jump if zero 217
block diagram 3
block transfer instructions 215
BRK 217
BSET 215
BSWAP 215, 218
BTJ 217
BTJNZ 217
BTJZ 217

C

CALL procedure 217
capture mode 81
capture/compare mode 82
cc 211
CCF 216
characteristics, electrical 185
clear 216
clock phase (SPI) 116
CLR 216
COM 217

- compare 82
- compare - extended addressing 214
- compare mode 82
- compare with carry 214
- compare with carry - extended addressing 214
- complement 217
- complement carry flag 215, 216
- condition code 211
- continuous conversion (ADC) 148
- continuous mode 81
- control register definition, UART 100
- control register, I2C 141
- counter modes 81
- CP 214
- CPC 214
- CPCX 214
- CPU and peripheral overview 3
- CPU control instructions 216
- CPX 214
- Customer Feedback Form 251
- customer feedback form 240
- Customer Information 251

D

- DA 211, 214
- data register, I2C 139
- DC characteristics 187
- debugger, on-chip 171
- DEC 214
- decimal adjust 214
- decrement 214
 - and jump non-zero 217
 - word 214
- DECW 214
- destination operand 212
- device, port availability 47
- DI 216
- direct address 211
- disable interrupts 216
- DJNZ 217
- DMA controller 5
- dst 212

E

- EI 216
- electrical characteristics 185
 - ADC 199
 - flash memory and timing 196
 - GPIO input data sample timing 200
 - watch-dog timer 197
- enable interrupt 216
- ER 211
- extended addressing register 211
- external pin reset 43
- external RC oscillator 196
- eZ8
 - features 3
- eZ8 CPU features 3
- eZ8 CPU instruction classes 214
- eZ8 CPU instruction notation 210
- eZ8 CPU instruction set 209
- eZ8 CPU instruction summary 218

F

- FCTL register 159
- features, Z8 Encore! 1
- first opcode map 231
- FLAGS 212
- flags register 212
- flash
 - controller 4
 - option bit address space 163
 - option bit configuration - reset 163
 - program memory address 0001H 165
- flash memory
 - arrangement 154
 - byte programming 157
 - code protection 156
 - control register definitions 159
 - controller bypass 158
 - electrical characteristics and timing 196
 - flash control register 159
 - flash status register 160
 - frequency high and low byte registers 161
 - mass erase 158
 - operation 155

- operation timing 155
- page erase 158
- page select register 160
- FPS register 160
- FSTAT register 160

G

- gated mode 82
- general-purpose I/O 47
- GPIO 4, 47
 - alternate functions 47
 - architecture 47
 - control register definitions 49
 - input data sample timing 200
 - interrupts 49
 - port A-C pull-up enable sub-registers 54
 - port A-H address registers 50
 - port A-H alternate function sub-registers 51
 - port A-H control registers 51
 - port A-H data direction sub-registers 51
 - port A-H high drive enable sub-registers 53
 - port A-H input data registers 54
 - port A-H output control sub-registers 52
 - port A-H output data registers 55
 - port A-H stop mode recovery sub-registers 53
 - port availability by device 47
 - port input timing 200
 - port output timing 201

H

- H 212
- HALT 216
- halt mode 45, 216
- hexadecimal number prefix/suffix 212

I

- I2C 4
 - 10-bit address read transaction 137
 - 10-bit address transaction 134
 - 10-bit addressed slave data transfer format 134
 - 10-bit receive data format 137

- 7-bit address transaction 132
- 7-bit address, reading a transaction 136
- 7-bit addressed slave data transfer format 131, 132, 133
- 7-bit receive data transfer format 136
- baud high and low byte registers 143, 145
- C status register 140
- control register definitions 139
- controller 127
- controller signals 9
- interrupts 128
- operation 128
- SDA and SCL signals 128
- stop and start conditions 130
- I2CBRH register 143, 144, 145
- I2CBRL register 143
- I2CCTL register 141
- I2CDATA register 139
- I2CSTAT register 140
- IM 211
- immediate data 211
- immediate operand prefix 212
- INC 214
- increment 214
- increment word 214
- INCW 214
- indexed 211
- indirect address prefix 212
- indirect register 211
- indirect register pair 211
- indirect working register 211
- indirect working register pair 211
- infrared encoder/decoder (IrDA) 109
- instruction set, ez8 CPU 209
- instructions
 - ADC 214
 - ADCX 214
 - ADD 214
 - ADDX 214
 - AND 217
 - ANDX 217
 - arithmetic 214
 - BCLR 215
 - BIT 215

- bit manipulation 215
- block transfer 215
- BRK 217
- BSET 215
- BSWAP 215, 218
- BTJ 217
- BTJNZ 217
- BTJZ 217
- CALL 217
- CCF 215, 216
- CLR 216
- COM 217
- CP 214
- CPC 214
- CPCX 214
- CPU control 216
- CPX 214
- DA 214
- DEC 214
- DECW 214
- DI 216
- DJNZ 217
- EI 216
- HALT 216
- INC 214
- INCW 214
- IRET 217
- JP 217
- LD 216
- LDC 216
- LDCI 215, 216
- LDE 216
- LDEI 215
- LDX 216
- LEA 216
- load 216
- logical 217
- MULT 214
- NOP 216
- OR 217
- ORX 217
- POP 216
- POPX 216
- program control 217
- PUSH 216
- PUSHX 216
- RCF 215, 216
- RET 217
- RL 218
- RLC 218
- rotate and shift 218
- RR 218
- RRC 218
- SBC 215
- SCF 215, 216
- SRA 218
- SRL 218
- SRP 216
- STOP 216
- SUB 215
- SUBX 215
- SWAP 218
- TCM 215
- TCMX 215
- TM 215
- TMX 215
- TRAP 217
- watch-dog timer refresh 216
- XOR 217
- XORX 217
- instructions, eZ8 classes of 214
- interrupt control register 67
- interrupt controller 5, 57
 - architecture 57
 - interrupt assertion types 60
 - interrupt vectors and priority 60
 - operation 59
 - register definitions 61
 - software interrupt assertion 60
- interrupt edge select register 67
- interrupt request 0 register 61
- interrupt request 1 register 62
- interrupt request 2 register 63
- interrupt return 217
- interrupt vector listing 57
- interrupts
 - not acknowledge 128
 - receive 128

- SPI 119
 - transmit 128
- UART 97
- introduction 1
- IR 211
- Ir 211
- IrDA
 - architecture 109
 - block diagram 109
 - control register definitions 112
 - operation 109
 - receiving data 111
 - transmitting data 110
- IRET 217
- IRQ0 enable high and low bit registers 63
- IRQ1 enable high and low bit registers 64
- IRQ2 enable high and low bit registers 65
- IRR 211
- Irr 211

J

- JP 217
- jump, conditional, relative, and relative conditional 217

L

- LD 216
- LDC 216
- LDCI 215, 216
- LDE 216
- LDEI 215, 216
- LDX 216
- LEA 216
- load 216
- load constant 215
- load constant to/from program memory 216
- load constant with auto-increment addresses 216
- load effective address 216
- load external data 216
- load external data to/from data memory and auto-increment addresses 215
- load external to/from data memory and auto-incre-

- ment addresses 216
- load instructions 216
- load using extended addressing 216
- logical AND 217
- logical AND/extended addressing 217
- logical exclusive OR 217
- logical exclusive OR/extended addressing 217
- logical instructions 217
- logical OR 217
- logical OR/extended addressing 217
- low power modes 45

M

- master interrupt enable 59
- master-in, slave-out and-in 115
- memory
 - program 13
- MISO 115
- mode
 - capture 81
 - capture/compare 82
 - continuous 81
 - counter 81
 - gated 82
 - one-shot 81
 - PWM 81
- modes 82
- MOSI 115
- MULT 214
- multiply 214
- multiprocessor mode, UART 95

N

- NOP (no operation) 216
- not acknowledge interrupt 128
- notation
 - b 211
 - cc 211
 - DA 211
 - ER 211
 - IM 211
 - IR 211

- Ir 211
- IRR 211
- Irr 211
- p 211
- R 211
- r 211
- RA 211
- RR 211
- rr 211
- vector 211
- X 211
- notational shorthand 211

O

OCD

- architecture 171
- auto-baud detector/generator 174
- baud rate limits 174
- block diagram 171
- breakpoints 175
- commands 176
- control register 181
- data format 173
- DBG pin to RS-232 Interface 172
- debug mode 173
- debugger break 217
- interface 171
- serial errors 174
- status register 183
- timing 202

OCD commands

- execute instruction (12H) 181
- read data memory (0DH) 180
- read OCD control register (05H) 179
- read OCD revision (00H) 178
- read OCD status register (02H) 178
- read program counter (07H) 179
- read program memory (0BH) 180
- read program memory CRC (0EH) 181
- read register (09H) 179
- read runtime counter (03H) 178
- step instruction (10H) 181
- stuff instruction (11H) 181

- write data memory (0CH) 180
- write OCD control register (04H) 178
- write program counter (06H) 179
- write program memory (0AH) 180
- write register (08H) 179
- on-chip debugger 5
- on-chip debugger (OCD) 171
- on-chip debugger signals 11
- on-chip oscillator 167
- one-shot mode 81
- opcode map
 - abbreviations 230
 - cell description 229
 - first 231
 - second after 1FH 232
- Operational Description 89
- OR 217
- ordering information 236
- ORX 217
- oscillator signals 11

P

- p 211
- packaging
 - PDIP 234
- part selection guide 2
- PC 212
- PDIP 234
- peripheral AC and DC electrical characteristics 195
- PHASE=0 timing (SPI) 117
- PHASE=1 timing (SPI) 118
- pin characteristics 12
- polarity 211
- POP 216
- pop using extended addressing 216
- POPX 216
- port availability, device 47
- port input timing (GPIO) 200
- port output timing, GPIO 201
- power supply signals 11
- power-down, automatic (ADC) 148
- power-on and voltage brown-out 195
- power-on reset (POR) 41

program control instructions 217
program counter 212
program flash
 configurations 153
program memory 13, 153
PUSH 216
push using extended addressing 216
PUSHX 216
PWM mode 81
PxADDR register 50
PxCTL register 51

R

R 211
r 211
RCF 215, 216
receive
 10-bit data format (I2C) 137
 7-bit data transfer format (I2C) 136
 IrDA data 111
receive interrupt 128
receiving UART data-interrupt-driven method 94
receiving UART data-pollled method 93
register 124, 211
 ADC control (ADCCTL) 150
 ADC data high byte (ADCDH) 151
 ADC data low bits (ADC DL) 151
 baud low and high byte (I2C) 143, 145
 baud rate high and low byte (SPI) 125
 control (SPI) 122
 control, I2C 141
 data, SPI 121
 flash control (FCTL) 159
 flash high and low byte (FFREQH and FRE-EQL) 161
 flash page select (FPS) 160
 flash status (FSTAT) 160
 GPIO port A-H address (PxADDR) 50
 GPIO port A-H alternate function sub-registers 52
 GPIO port A-H control address (PxCTL) 51
 GPIO port A-H data direction sub-registers 51
 I2C baud rate high (I2CBRH) 143, 144, 145

I2C control (I2CCTL) 141
I2C data (I2CDATA) 139
I2C status 140
I2C status (I2CSTAT) 140
I2Cbaud rate low (I2CBRL) 143
mode, SPI 124
OCD control 181
OCD status 183
SPI baud rate high byte (SPIBRH) 125
SPI baud rate low byte (SPIBRL) 126
SPI control (SPICTL) 122
SPI data (SPIDATA) 121
SPI status (SPISTAT) 123
status, I2C 140
status, SPI 123
UARTx baud rate high byte (UxBRH) 106
UARTx baud rate low byte (UxBRL) 106
UARTx Control 0 (UxCTL0) 103, 106
UARTx control 1 (UxCTL1) 104
UARTx receive data (UxRXD) 101
UARTx status 0 (UxSTAT0) 101
UARTx status 1 (UxSTAT1) 102
UARTx transmit data (UxTXD) 100
watch-dog timer control (WDTCTL) 86
watch-dog timer reload high byte (WDTH) 88
watch-dog timer reload low byte (WDTL) 88
watch-dog timer reload upper byte (WDTU) 87
register address (RA) 211
register file 13
register file address map 15
register pair 211
register pointer 212
reset
 and stop mode characteristics 39
 and stop mode recovery 39
 carry flag 215
 controller 5
 sources 40
RET 217
return 217
RL 218
RLC 218
rotate and shift instructions 218
rotate left 218

rotate left through carry 218
rotate right 218
rotate right through carry 218
RP 212
RR 211, 218
rr 211
RRC 218

S

SBC 215
SCF 215, 216
SCK 115
SDA and SCL (IrDA) signals 128
second opcode map after 1FH 232
serial clock 115
serial peripheral interface (SPI) 113
set carry flag 215, 216
set register pointer 216
shift right arithmetic 218
shift right logical 218
signal descriptions 9
single-shot conversion (ADC) 148
SIO 5
slave data transfer formats (I2C) 134
slave select 116
software trap 217
source operand 212
SP 212
SPI
 architecture 113
 baud rate generator 120
 baud rate high and low byte register 125
 clock phase 116
 configured as slave 114
 control register 122
 control register definitions 121
 data register 121
 error detection 119
 interrupts 119
 mode fault error 119
 mode register 124
 multi-master operation 118
 operation 114

 overrun error 119
 signals 115
 single master, multiple slave system 114
 single master, single slave system 113
 status register 123
 timing, PHASE = 0 117
 timing, PHASE=1 118
SPI controller signals 10
SPI mode (SPIMODE) 124
SPIBRH register 125
SPIBRL register 126
SPICTL register 122
SPIDATA register 121
SPIMODE register 124
SPISTAT register 123
SRA 218
src 212
SRL 218
SRP 216
SS, SPI signal 115
stack pointer 212
status register, I2C 140
STOP 216
stop mode 45, 216
stop mode recovery
 sources 43
 using a GPIO port pin transition 44
 using watch-dog timer time-out 44
SUB 215
subtract 215
subtract - extended addressing 215
subtract with carry 215
subtract with carry - extended addressing 215
SUBX 215
SWAP 218
swap nibbles 218
symbols, additional 212
system and core resets 40

T

TCM 215
TCMX 215
test complement under mask 215

test complement under mask - extended addressing 215
test under mask 215
test under mask - extended addressing 215
timer signals 10
timers 5, 69
 architecture 69
 block diagram 70
 capture mode 74, 81
 capture/compare mode 77, 82
 compare mode 75, 82
 continuous mode 71, 81
 counter mode 72
 counter modes 81
 gated mode 76, 82
 one-shot mode 70, 81
 operating mode 70
 PWM mode 73, 81
 reading the timer count values 77
 reload high and low byte registers 79
 timer control register definitions 78
 timer output signal operation 78
timers 0-3
 control 0 registers 80
 control registers 81
 high and low byte registers 78, 79
TM 215
TMX 215
transmit
 IrDA data 110
transmit interrupt 128
transmitting UART data-interrupt-driven method 92
transmitting UART data-polled method 91
TRAP 217

U

UART 4
 architecture 89
 baud rate generator 99
 baud rates table 107
 control register definitions 100
 controller signals 10

data format 90
interrupts 97
multiprocessor mode 95
receiving data using interrupt-driven method 94
receiving data using the polled method 93
transmitting data using the interrupt-driven method 92
transmitting data using the polled method 91
x baud rate high and low registers 106
x control 0 and control 1 registers 103
x status 0 and status 1 registers 101, 102

UxBRH register 106
UxBRL register 106
UxCTL0 register 103, 106
UxCTL1 register 104
UxRXD register 101
UxSTAT0 register 101
UxSTAT1 register 102
UxTXD register 100

V

vector 211
voltage brown-out reset (VBR) 41

W

watch-dog timer
 approximate time-out delay 83
 CNTL 42
 control register 86
 electrical characteristics and timing 197
 interrupt in normal operation 84
 refresh 84, 216
 reload unlock sequence 85
 reload upper, high and low registers 87
 reset 42
 reset in normal operation 85
 reset in STOP mode 84, 85
 time-out response 84
WDTCTL register 86
WDTH register 88
WDTL register 88
working register 211

working register pair 211
WTDU register 87

X

X 211
XOR 217
XORX 217

Z

Z8 Encore!
 block diagram 3
 features 1
 introduction 1
 part selection guide 2



Customer Support

For answers to technical questions about the product, documentation, or any other issues with Zilog's offerings, please visit Zilog's Knowledge Base at <http://www.zilog.com/kb>.

For any comments, detail technical questions, or reporting problems, please visit Zilog's Technical Support at <http://support.zilog.com>.