



**High-Performance 8-Bit Microcontrollers**

## **Z8 Encore! XP<sup>®</sup> F1680 Series**

### **Product Specification**

PS025006-1007

PRELIMINARY



**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**

©2007 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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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](http://www.zilog.com/quality).

# Revision History

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

Date	Revision Level	Description	Page No
October 2007	06	Updated <a href="#">Table 146</a> , <a href="#">Trim Bit Address 0007H</a> , <a href="#">Trim Bit Address 0008H</a> , <a href="#">Table 191</a> , <a href="#">Table 192</a> , <a href="#">Figure 73</a> , <a href="#">Table 196</a> , <a href="#">Table 198</a> , <a href="#">Table 199</a> , <a href="#">Table 201</a> , <a href="#">Table 202</a> , and <a href="#">Table 203</a> . Added <a href="#">Figure 72</a> .	<a href="#">272</a> , <a href="#">277</a> , <a href="#">278</a> , <a href="#">338</a> , <a href="#">340</a> , <a href="#">344</a> , <a href="#">346</a> , <a href="#">347</a> , <a href="#">348</a> , <a href="#">349</a> , <a href="#">350</a> , <a href="#">351</a> , <a href="#">343</a>
September 2007	05	Updated <a href="#">Table 192</a> .	<a href="#">340</a>
August 2007	04	Changed description of <a href="#">Z8F16800144ZCOG</a> to <a href="#">Z8 Encore! XP Dual 44-pin F1680 Series Development Kit</a> . Updated electrical characteristics in <a href="#">Table 191</a> , <a href="#">Table 192</a> , <a href="#">Table 194</a> , <a href="#">Table 195</a> , <a href="#">Table 197</a> , <a href="#">Table 198</a> . Removed <a href="#">VBO_Trim</a> section and table.	<a href="#">369</a> <a href="#">338</a> , <a href="#">340</a> , <a href="#">345</a> , <a href="#">346</a> , <a href="#">346</a> , <a href="#">347</a> , <a href="#">274</a>
May 2007	03	Updated <a href="#">Voltage Brownout Reset</a> section, <a href="#">Table 193</a> , <a href="#">Table 194</a> , and <a href="#">Table 204</a>	<a href="#">37</a> , <a href="#">344</a> , <a href="#">345</a> , <a href="#">351</a>
April 2007	02	Updated thoroughly	All
October 2005	01	Original Issue	All

# Table of Contents

<b>Overview</b>	<b>1</b>
Features	1
Part Selection Guide	2
Block Diagram	3
eZ8 CPU and Peripheral Overview	4
eZ8 CPU Features	4
General Purpose Input/Output	4
Flash Controller	4
Non Volatile Data Storage	5
Internal Precision Oscillator	5
Crystal Oscillator	5
Secondary Oscillator	5
10-Bit Analog-to-Digital Converter	5
Low-Power Operational Amplifier	5
Analog Comparator	5
Temperature Sensor	5
Low-Voltage Detector	6
Enhanced SPI	6
UART with LIN	6
Master/Slave I2C	6
Timers	6
Multi-Channel Timer	7
Interrupt Controller	7
Reset Controller	7
On-Chip Debugger	7
Direct LED Drive	7
Acronyms and Expansions	8
<b>Pin Description</b>	<b>11</b>
Overview	11
Available Packages	11
Pin Configurations	11
Signal Descriptions	14
Pin Characteristics	17
<b>Address Space</b>	<b>21</b>
Overview	21

Register File .....	21
Program Memory .....	22
Data Memory .....	23
Flash Information Area .....	23
<b>Register Map .....</b>	<b>25</b>
<b>Reset, Stop Mode Recovery, and</b>	
<b>Low-Voltage Detection .....</b>	<b>33</b>
Overview .....	33
Reset Types .....	33
Reset Sources .....	35
Power-On Reset .....	35
Voltage Brownout Reset .....	37
Watchdog Timer Reset .....	38
External Reset Input .....	38
External Reset Indicator .....	39
On-Chip Debugger Initiated Reset .....	39
Stop Mode Recovery .....	39
Stop Mode Recovery Using Watchdog Timer Timeout .....	40
Stop Mode Recovery Using Timer Interrupt .....	40
Stop Mode Recovery Using Comparator Interrupt .....	41
Stop Mode Recovery Using GPIO Port Pin Transition .....	41
Stop Mode Recovery Using External RESET Pin .....	41
Low-Voltage Detection .....	41
Reset Register Definitions .....	42
<b>Low-Power Modes .....</b>	<b>45</b>
Overview .....	45
STOP Mode .....	45
HALT Mode .....	46
Peripheral-Level Power Control .....	46
Power Control Register Definitions .....	47
<b>General-Purpose Input/Output .....</b>	<b>49</b>
Overview .....	49
GPIO Port Availability by Device .....	49
Architecture .....	50
GPIO Alternate Functions .....	50
Direct LED Drive .....	51
Shared Reset Pin .....	51

Crystal Oscillator Override . . . . .	51
32 kHz Secondary Oscillator Override . . . . .	51
5 V Tolerance . . . . .	51
External Clock Setup . . . . .	52
GPIO Interrupts . . . . .	61
GPIO Control Register Definitions . . . . .	61
Port A–E Address Registers . . . . .	62
Port A–E Control Registers . . . . .	63
Port A–E Data Direction Subregisters . . . . .	63
Port A–E Alternate Function Subregisters . . . . .	64
Port A–E Output Control Subregisters . . . . .	65
Port A–E High Drive Enable Subregisters . . . . .	65
Port A–E Stop Mode Recovery Source Enable Subregisters . . . . .	66
Port A–E Pull-up Enable Subregisters . . . . .	66
Port A–E Alternate Function Set 1 Subregisters . . . . .	67
Port A–E Alternate Function Set 2 Subregisters . . . . .	67
Port A–E Input Data Registers . . . . .	68
Port A–E Output Data Register . . . . .	69
LED Drive Enable Register . . . . .	69
LED Drive Level Register . . . . .	69
<b>Interrupt Controller . . . . .</b>	<b>71</b>
Overview . . . . .	71
Interrupt Vector Listing . . . . .	71
Architecture . . . . .	73
Operation . . . . .	73
Master Interrupt Enable . . . . .	73
Interrupt Vectors and Priority . . . . .	74
Interrupt Assertion . . . . .	74
Software Interrupt Assertion . . . . .	75
Interrupt Control Register Definitions . . . . .	75
Interrupt Request 0 Register . . . . .	75
Interrupt Request 1 Register . . . . .	76
Interrupt Request 2 Register . . . . .	77
IRQ0 Enable High and Low Bit Registers . . . . .	78
IRQ1 Enable High and Low Bit Registers . . . . .	79
IRQ2 Enable High and Low Bit Registers . . . . .	80
Interrupt Edge Select Register . . . . .	82
Shared Interrupt Select Register . . . . .	82
Interrupt Control Register . . . . .	83



<b>Timers</b>	<b>85</b>
Overview	85
Architecture	85
Operation	86
Timer Clock Source	86
Low-Power Modes	87
Timer Operating Modes	88
Reading the Timer Count Values	104
Timer Output Signal Operation	104
Timer Noise Filter	104
Architecture	105
Timer Control Register Definitions	107
Timer 0–2 High and Low Byte Registers	107
Timer Reload High and Low Byte Registers	108
Timer 0–2 PWM0 High and Low Byte Registers	108
Timer 0–2 PWM1 High and Low Byte Registers	109
Timer 0–2 Control Registers	110
Timer 0–2 Status Registers	115
Timer 0–2 Noise Filter Control Register	116
<b>Multi-Channel Timer</b>	<b>119</b>
Overview	119
Architecture	119
Timer Operation	120
Multi-Channel Timer Counter	120
Clock Source	120
Multi-Channel Timer Clock Prescaler	120
Multi-Channel Timer Start	120
Multi-Channel Timer Mode Control	120
Count Modulo Mode	121
Count Up/Down Mode	121
Capture/Compare Channel Operation	122
One-Shot Compare Operation	122
Continuous Compare Operation	123
PWM Output Operation	123
Capture Operation	123
Multi-Channel Timer Interrupts	123
Timer Interrupt	123
Capture/Compare Channel Interrupt	124
Low-Power Modes	124

Operation in HALT Mode . . . . .	124
Operation in STOP Mode . . . . .	124
Power Reduction During Operation . . . . .	124
Multi-Channel Timer Applications Examples . . . . .	124
PWM Programmable Deadband Generation . . . . .	124
Multiple Timer Intervals Generation . . . . .	124
Multi-Channel Timer Control Register Definitions . . . . .	126
Multi-Channel Timer Address Map . . . . .	126
Multi-Channel Timer High and Low Byte Registers . . . . .	127
MCT Reload High and Low Byte Registers . . . . .	128
MCT Sub-Address Register . . . . .	129
MCT SubRegister x (0, 1, or 2) . . . . .	129
Multi-Channel Timer Control 0, Control 1 Registers . . . . .	130
Multi-Channel Timer Channel Status 0 and Status 1 Registers . . . . .	132
Multi-Channel Timer Channel-y Control Registers . . . . .	133
One-Shot Operation . . . . .	133
Continuous Compare Operation . . . . .	133
PWM Output Operation . . . . .	133
Capture Operation . . . . .	134
Multi-Channel Timer Channel-y High and Low Byte Registers . . . . .	134
<b>Watchdog Timer . . . . .</b>	<b>137</b>
Operation . . . . .	137
Watchdog Timer Refresh . . . . .	138
Watchdog Timer Timeout Response . . . . .	138
Watchdog Timer Reload Unlock Sequence . . . . .	139
Watchdog Timer Register Definitions . . . . .	139
Watchdog Timer Reload High and Low Byte Registers . . . . .	139
<b>LIN-UART . . . . .</b>	<b>141</b>
Architecture . . . . .	141
Data Format for Standard UART Modes . . . . .	142
Transmitting Data Using Polled Method . . . . .	143
Transmitting Data Using Interrupt-Driven Method . . . . .	144
Receiving Data Using Polled Method . . . . .	145
Receiving Data Using the Interrupt-Driven Method . . . . .	146
Clear To Send Operation . . . . .	147
External Driver Enable . . . . .	147
LIN-UART Special Modes . . . . .	148
MULTIPROCESSOR Mode . . . . .	148
LIN Protocol Mode . . . . .	150

LIN-UART Interrupts	153
LIN-UART Baud Rate Generator	156
Noise Filter	157
Architecture	157
Operation	157
LIN-UART Control Register Definitions	159
LIN-UART Transmit Data Register	159
LIN-UART Receive Data Register	159
LIN-UART Status 0 Register	160
LIN-UART Mode Select and Status Register	163
LIN-UART Control 0 Register	165
LIN-UART Control 1 Registers	166
Noise Filter Control Register	169
LIN Control Register	170
LIN-UART Address Compare Register	171
LIN-UART Baud Rate High and Low Byte Registers	172
<b>Infrared Encoder/Decoder</b>	<b>177</b>
Overview	177
Architecture	177
Operation	177
Transmitting IrDA Data	178
Receiving IrDA Data	179
Infrared Encoder/Decoder Control Register Definitions	180
<b>Analog-to-Digital Converter</b>	<b>181</b>
Architecture	181
Operation	181
ADC Timing	182
ADC Interrupt	182
Reference Buffer	183
Internal Voltage Reference Generator	184
Calibration and Compensation	184
ADC Control Register Definitions	184
ADC Control Register 0	184
ADC Raw Data High Byte Register	185
ADC Data High Byte Register	186
ADC Data Low Bit Register	186
Sample Settling Time Register	187
Sample Time Register	187
ADC Clock Prescale Register	188

<b>Low-Power Operational Amplifier</b>	<b>189</b>
<b>Enhanced Serial Peripheral Interface</b>	<b>191</b>
Overview	191
Architecture	191
ESPI Signals	193
Master-In/Slave-Out	193
Master-Out/Slave-In	193
Serial Clock	193
Slave Select	194
ESPI Register Overview	194
Operation	194
Throughput	195
ESPI Clock Phase and Polarity Control	195
Slave Select Modes of Operation	197
SPI Protocol Configuration	200
Error Detection	203
ESPI Interrupts	204
ESPI Baud Rate Generator	205
ESPI Control Register Definitions	205
ESPI Data Register	205
ESPI Transmit Data Command and Receive Data	
Buffer Control Register	206
ESPI Control Register	207
ESPI Mode Register	209
ESPI Status Register	210
ESPI State Register	211
ESPI Baud Rate High and Low Byte Registers	213
<b>I2C Master/Slave Controller</b>	<b>215</b>
Architecture	215
I2C Master/Slave Controller Registers	216
Operation	217
SDA and SCL Signals	217
I <sup>2</sup> C Interrupts	218
Start and Stop Conditions	220
Software Control of I2C Transactions	220
Master Transactions	221
Slave Transactions	228
I <sup>2</sup> C Control Register Definitions	235
I2C Data Register	235

I2C Interrupt Status Register . . . . .	236
I2C Control Register . . . . .	237
I2C Baud Rate High and Low Byte Registers . . . . .	239
I2C State Register . . . . .	240
I2C Mode Register . . . . .	243
I2C Slave Address Register . . . . .	244
<b>Comparator . . . . .</b>	<b>247</b>
Overview . . . . .	247
Operation . . . . .	247
Comparator Control Register Definitions . . . . .	248
Comparator 0 Control Register . . . . .	248
Comparator 1 Control Register . . . . .	249
<b>Temperature Sensor . . . . .</b>	<b>251</b>
Overview . . . . .	251
Operation . . . . .	251
<b>Flash Memory . . . . .</b>	<b>253</b>
Overview . . . . .	253
Flash Information Area . . . . .	253
Operation . . . . .	256
Flash Operation Timing Using Flash Frequency Registers . . . . .	258
Flash Code Protection Against External Access . . . . .	258
Flash Code Protection Against Accidental Program and Erasure . . . . .	258
Byte Programming . . . . .	259
Page Erase . . . . .	260
Mass Erase . . . . .	260
Flash Controller Bypass . . . . .	260
Flash Controller Behavior in Debug Mode . . . . .	261
Flash Control Register Definitions . . . . .	261
Flash Control Register . . . . .	261
Flash Status Register . . . . .	262
Flash Page Select Register . . . . .	263
Flash Sector Protect Register . . . . .	263
Flash Frequency High and Low Byte Registers . . . . .	264
<b>Flash Option Bits . . . . .</b>	<b>267</b>
Overview . . . . .	267
Operation . . . . .	267
Option Bit Configuration by Reset . . . . .	267
Option Bit Types . . . . .	267



Flash Option Bit Control Register Definitions .....	269
Trim Bit Address Register .....	269
Trim Bit Data Register .....	269
Flash Option Bit Address Space .....	270
Flash Program Memory Address 0000H .....	270
Flash Program Memory Address 0001H .....	271
Trim Bit Address Space .....	272
Trim Bit Address 0000H .....	273
Trim Bit Address 0001H .....	273
Trim Bit Address 0002H .....	274
Trim Bit Address 0003H .....	274
Trim Bit Address 0004H .....	276
Trim Bit Address 0005H .....	276
Trim Bit Address 0006H .....	277
Trim Bit Address 0007H .....	277
Trim Bit Address 0008H .....	278
Zilog Calibration Bits .....	278
Temperature Sensor Calibration Bits .....	278
<b>Non-Volatile Data Storage .....</b>	<b>281</b>
Overview .....	281
Operation .....	281
NVDS Code Interface .....	281
Byte Write .....	282
Byte Read .....	283
Power Failure Protection .....	284
Optimizing NVDS Memory Usage for Execution Speed .....	284
<b>On-Chip Debugger .....</b>	<b>285</b>
Overview .....	285
Architecture .....	285
Operation .....	286
On-Chip Debugger Interface .....	286
DEBUG Mode .....	287
OCD Data Format .....	287
OCD Auto-Baud Detector/Generator .....	288
High Speed Synchronous .....	289
OCD Serial Errors .....	289
Automatic Reset .....	290
Transmit Flow Control .....	290
Breakpoints .....	291

OCD CNTR Register . . . . .	291
On-Chip Debugger Commands . . . . .	292
On-Chip Debugger Control Register Definitions . . . . .	299
OCD Control Register . . . . .	299
OCD Status Register . . . . .	300
Line Control Register . . . . .	301
Baud Reload Register . . . . .	302
<b>Oscillator Control . . . . .</b>	<b>303</b>
Overview . . . . .	303
Operation . . . . .	303
System Clock Selection . . . . .	303
Clock Failure Detection and Recovery . . . . .	305
Peripheral Clock . . . . .	306
Oscillator Control Register Definitions . . . . .	306
Oscillator Control0 Register . . . . .	306
Oscillator Control1 Register . . . . .	307
<b>Crystal Oscillator . . . . .</b>	<b>309</b>
Overview . . . . .	309
Operating Modes . . . . .	309
Main Crystal Oscillator Operation . . . . .	310
Main Oscillator Operation with External RC Network . . . . .	311
Secondary Crystal Oscillator Operation . . . . .	313
<b>Internal Precision Oscillator . . . . .</b>	<b>315</b>
Overview . . . . .	315
Operation . . . . .	315
<b>eZ8 CPU Instruction Set . . . . .</b>	<b>317</b>
Assembly Language Programming Introduction . . . . .	317
Assembly Language Syntax . . . . .	318
eZ8 CPU Instruction Notation . . . . .	318
eZ8 CPU Instruction Classes . . . . .	320
eZ8 CPU Instruction Summary . . . . .	325
<b>Opcode Maps . . . . .</b>	<b>334</b>
<b>Electrical Characteristics . . . . .</b>	<b>337</b>
Absolute Maximum Ratings . . . . .	337
DC Characteristics . . . . .	338
AC Characteristics . . . . .	344



On-Chip Peripheral AC and DC Electrical Characteristics ..... 345

    General Purpose I/O Port Input Data Sample Timing ..... 352

    General Purpose I/O Port Output Timing ..... 354

    On-Chip Debugger Timing ..... 355

    UART Timing ..... 356

**Packaging ..... 359**

**Ordering Information ..... 367**

**Index ..... 373**

**Customer Support ..... 383**

# Overview

Zilog's Z8 Encore! XP<sup>®</sup> F1680 Series MCU family is based on Zilog's advanced 8-bit eZ8 CPU core. This microcontroller is optimized for low-power applications and supports 1.8 V to 3.6 V wide low-voltage operation with extremely Low ACTIVE, HALT, and STOP mode currents. It is an assortment of speed and low-power options. In addition, the feature-rich analog and digital peripherals of the Z8 Encore! XP F1680 Series makes it suitable for a variety of applications including safety and security, utility metering, digital power supervisory, handheld electronic devices, and general motor control.

## Features

Key features of Z8 Encore! XP F1680 Series MCU include:

- 20 MHz eZ8 CPU core
- 8 KB, 16 KB, or 24 KB Flash memory with in-circuit programming capability
- 1 KB or 2 KB Register RAM
- 1 KB Program RAM for program code shadowing and data storage (optional)
- 128 B or 256 B Non-Volatile Data Storage (NVDS)
- Up to 8-Channel, 10-bit Analog-to-Digital Converter (ADC)
- On-chip Temperature Sensor
- Up to two on-chip analog comparators (20-pin and 28-pin packages contain only one)
- On-chip Low-Power Operational Amplifier (LPO)
- Two full-duplex 9-bit UART ports with the support of Local Interconnect Network (LIN) protocol (20-pin and 28-pin packages contain only one)
- Infrared Data Association (IrDA)-compliant infrared encoders/decoders, integrated with UARTs
- Enhanced Serial Peripheral Interface (SPI) controller (except 20-pin packages)
- I<sup>2</sup>C controller which supports Master/Slave modes
- Three enhanced 16-bit Timers with Capture, Compare, and PWM capability
- Additional two basic 16-bit timers with interrupt (shared as UART Baud Rate Generator)
- Optional 16-bit Multi-Channel Timer which supports four Capture/Compare/PWM modules (44-pin packages only)

- Watchdog Timer (WDT) with dedicated internal RC oscillator
- 17 to 37 General-Purpose Input/Output (GPIO) pins depending upon package
- Up to 8 direct LED drives with programmable drive current capability
- Up to 31 interrupt sources with up to 24 interrupt vectors
- On-Chip Debugger (OCD)
- Power-On Reset (POR) and Voltage Brownout (VBO) protection
- Built-in Low-Voltage Detection (LVD) with programmable voltage threshold
- 32 kHz secondary oscillator for Timers
- Internal Precision Oscillator (IPO) with output frequency in the range of 43.2 kHz to 11.0592 MHz
- Crystal oscillator with three power settings and external RC network option
- Wide operation voltage range: 1.8–3.6 V
- 20-, 28-, 40-, and 44-pin packages
- 0 °C to +70 °C (standard) and –40 °C to +105 °C (extended) operating temperature ranges

## Part Selection Guide

[Table 1](#) displays basic features and package styles available for each device within the Z8 Encore! XP F1680 Series product line.

**Table 1. Z8 Encore! XP F1680 Series Family Part Selection Guide**

Part Number	Flash (KB)	RAM (B)	Program RAM (B)	NVDS (B)	I/O	ADC Inputs	SPI	I <sup>2</sup> C	UARTs	Packages
Z8F2480	24	2048	1024	—	17–37	7–8	0–1	1	1–2	20-, 28-, 40-, and 44-pin
Z8F1680	16	2048	1024	256	17–37	7–8	0–1	1	1–2	20-, 28-, 40-, and 44-pin
Z8F0880	8	1024	1024	128	17–37	7–8	0–1	1	1–2	20-, 28-, 40-, and 44-pin

## Block Diagram

Figure 1 illustrates the block diagram of the architecture of Z8 Encore! XP F1680 Series devices.

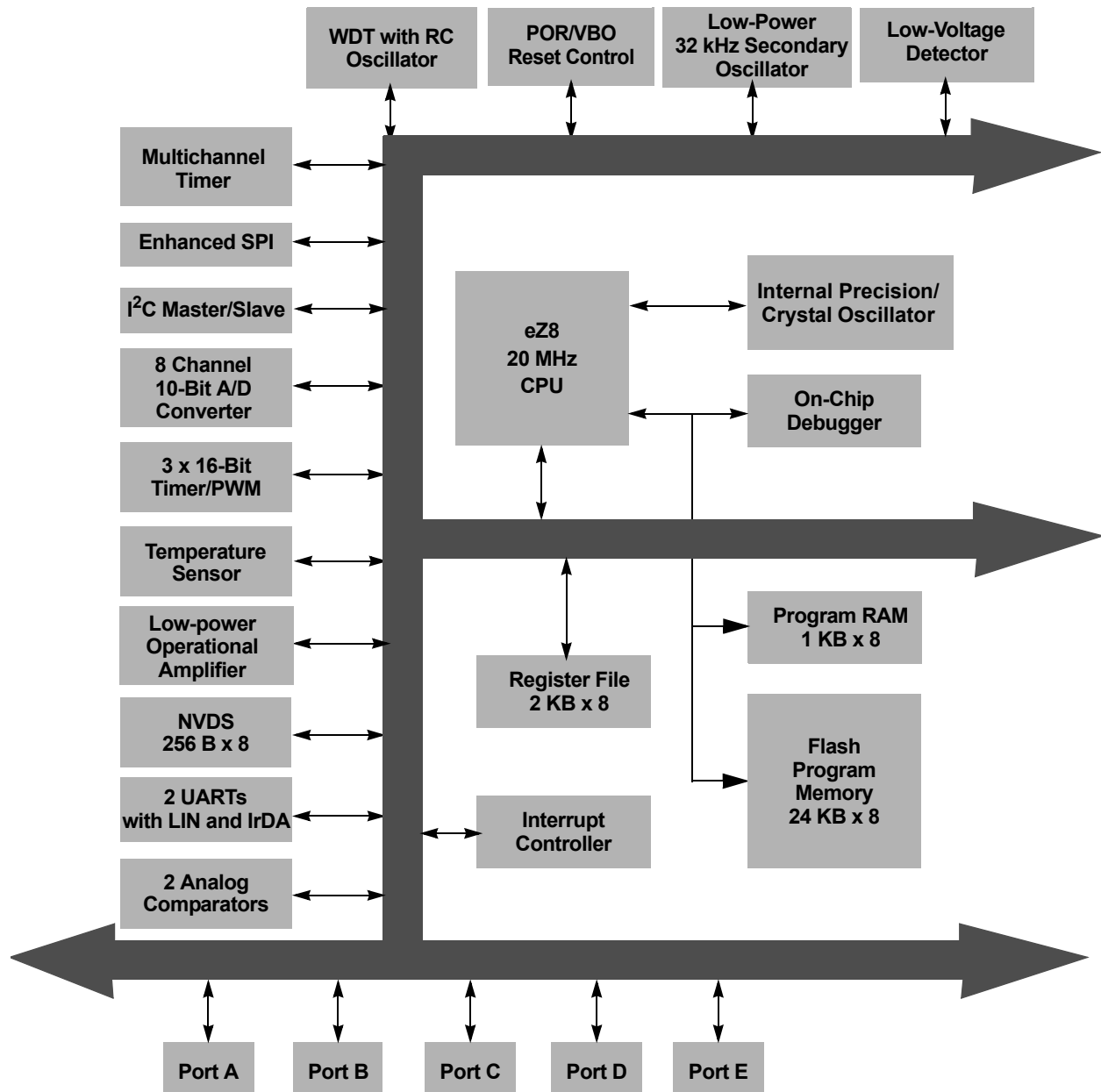


Figure 1. Z8 Encore! XP F1680 Series Block Diagram

## eZ8 CPU and Peripheral Overview

### eZ8 CPU Features

Zilog's eZ8 CPU, latest 8-bit CPU meets the continuing demand for faster and more code-efficient microcontrollers. It executes a superset of the original Z8<sup>®</sup> 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 greater depth in sub-routine calls and interrupts more than hardware stacks.
- Compatible with existing Z8 code.
- Expanded internal Register File allows access 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.
- 2–9 clock cycles per instruction.

For more details on eZ8 CPU, refer to *eZ8 CPU User Manual (UM0077)* available at [www.zilog.com](http://www.zilog.com).

### General Purpose Input/Output

The Z8 Encore! XP F1680 Series features 17 to 37 port pins (Ports A–E) for general purpose input/output (GPIO) pins. The number of GPIO pins available is a function of package. Each pin is individually programmable.

### Flash Controller

The Flash Controller is used to program and erase Flash memory. The Flash Controller supports protection against accidental program and erasure.

## **Non Volatile Data Storage**

The Non Volatile Data Storage (NVDS) uses a hybrid hardware/software scheme to implement a byte-programmable data memory and is capable of over 100,000 write cycles.

## **Internal Precision Oscillator**

The internal precision oscillator (IPO) is a trimmable clock source which requires no external components. You can select IPO frequency from one of eight frequencies (43.2 kHz to 11.0592 MHz) and is available with factory-trimmed calibration data.

## **Crystal Oscillator**

The crystal oscillator circuit provides highly accurate clock frequencies using an external crystal, ceramic resonator, or RC network.

## **Secondary Oscillator**

The secondary oscillator is a low-power oscillator, which is optimized for use with a 32 kHz watch crystal. It can be used as timer/counter clock source in any mode.

## **10-Bit Analog-to-Digital Converter**

The Analog-to-Digital Converter (ADC) converts an analog input signal to a 10-bit binary number. The ADC supports up to eight analog input sources multiplexed with GPIO ports.

## **Low-Power Operational Amplifier**

The low-power operational amplifier (LPO) is a general-purpose operational amplifier primarily targeted for current sense applications. The LPO output can be internally routed to the ADC or externally to a pin.

## **Analog Comparator**

The analog comparator compares the signal at an input pin with either an internal programmable voltage reference or a second-input pin. The comparator output is used to either drive an output pin or to generate an interrupt.

## **Temperature Sensor**

The temperature sensor produces an analog output proportional to the device temperature. This signal is sent either to the ADC or to the analog comparator.

## **Low-Voltage Detector**

The low-voltage detector generates an interrupt when the supply voltage drops below a user-programmable level.

## **Enhanced SPI**

The enhanced SPI is a full-duplex, buffered, synchronous character-oriented channel which supports a four-wire interface.

## **UART with LIN**

A full-duplex 9-bit UART provides serial, asynchronous communication, and supports the local interconnect network (LIN) serial communications protocol. The UART supports 8-bit and 9-bit data modes, selectable parity, and an efficient bus transceiver Driver Enable signal for controlling a multi-transceiver bus, such as RS-485. The LIN bus is a cost-efficient, single-master, multiple-slave organization which supports speed up to 20 Kbits.

## **Master/Slave I<sup>2</sup>C**

The inter-integrated circuit (I<sup>2</sup>C) controller makes the Z8 Encore! XP F1680 Series products compatible with the I<sup>2</sup>C protocol. The I<sup>2</sup>C controller consists of two bi-directional bus lines:

1. Serial data (SDA) line
2. Serial clock (SCL) line

It also supports Master, Slave, and Multi-Master Operations.

## **Timers**

Three enhanced 16-bit reloadable timers are used for timing/counting events or motor control operations. These timers provide a 16-bit programmable reload counter and operate in ONE-SHOT, CONTINUOUS, GATED, CAPTURE, CAPTURE RESTART, COMPARE, CAPTURE and COMPARE, PWM SINGLE OUTPUT, PWM DUAL OUTPUT, TRIGGERED ONE-SHOT, and DEMODULATION modes. In addition to these three enhanced 16-bit timers, there are two basic 16-bit timers with interrupt function. The two timers are used as Baud Rate Generator (BRG) when UART is enabled and configured as basic 16-bit timers when UART is disabled.

## **Multi-Channel Timer**

The multi-channel timer has a 16-bit up/down counter and a 4-channel Capture/Compare/PWM channel array. This timer enables the support of multiple synchronous Capture/Compare/PWM channels based on a single timer.

## **Interrupt Controller**

The Z8 Encore! XP F1680 Series products support up to thirty-one interrupt sources with twenty-four interrupt vectors. These interrupts consist of up to fifteen internal peripheral interrupts and up to sixteen GPIO pin interrupts. The interrupts have three levels of programmable-interrupt priority.

## **Reset Controller**

The Z8 Encore! XP F1680 Series products are reset using the RESET pin, POR, WDT timeout, STOP mode exit, or VBO warning signal. The RESET pin is bidirectional, that is, it functions as reset source as well as a reset indicator.

## **On-Chip Debugger**

The Z8 Encore! XP F1680 Series products feature an integrated OCD. The OCD provides a rich-set of debugging capabilities, such as reading and writing registers, programming Flash memory, setting breakpoints, and executing code. The OCD uses one single-pin interface for communication with an external host.

## **Direct LED Drive**

The Port C pins also provide a current synchronized output capable of driving an LED without requiring any external resistor. Up to eight LEDs are driven with individually programmable drive current level from 3 mA to 20 mA.

## Acronyms and Expansions

This document uses the following acronyms and expansions.

Abbreviations/ Acronyms	Expansions
ADC	Analog-to-Digital Converter
NVDS	Non Volatile Data Storage
LPO	Low-Power Operational Amplifier
LIN	Local Interconnect Network
SPI	Serial Peripheral Interface
ESPI	Enhanced 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
IPO	Internal Precision Oscillator
UART	Universal Asynchronous Receiver/Transmitter
IrDA	Infrared Data Association
I <sup>2</sup> C	Inter-integrated circuit
PDIP	Plastic Dual Inline Package
SOIC	Small Outline Integrated Circuit
SSOP	Small Shrink Outline Package
QFN	Quad Flat No Lead
LQFP	Low-Profile Quad Flat Package
PRAM	Program RAM
PC	Program counter
IRQ	Interrupt request
ISR	Interrupt service routine
MSB	Most-significant byte

<b>Abbreviations/ Acronyms</b>	<b>Expansions</b>
LSB	Least-significant byte
PWM	Pulse Width Modulation
CI	Channel Interrupt
TI	Timer Interrupt
Endec	Encoder/Decoder
I <sup>2</sup> S	Inter IC Sound
TDM	Time division multiplexing
TTL	Transistor-Transistor Logic
SAR	Successive Approximation Register



# Pin Description

## Overview

The Z8 Encore! XP F1680 Series products are available in a variety of package styles and pin configurations. This chapter describes the signals and available pin configurations for each of the package styles. For information on the physical package specifications, see [Packaging](#) on page 359.

## Available Packages

[Table 2](#) lists the package styles available for each device in the Z8 Encore! XP F1680 Series product line.

**Table 2. Z8 Encore! XP F1680 Series Package Options**

Part Number	ADC	20-pin PDIP	20-pin SOIC	20-pin SSOP	28-pin PDIP	28-pin SOIC	28-pin SSOP	40-pin PDIP	44-pin QFN	44-pin LQFP
Z8F2480	Yes	X	X	X	X	X	X	X	X	X
Z8F1680	Yes	X	X	X	X	X	X	X	X	X
Z8F0880	Yes	X	X	X	X	X	X	X	X	X

## Pin Configurations

[Figure 2](#) on page 12 through [Figure 5](#) on page 14 illustrate the pin configurations of all the packages available in the Z8 Encore! XP F1680 Series. For description of the signals, see [Table 3](#) on page 15.

At reset, all port pins default to an input state. In addition, any alternate functionality is not enabled, so the pins function as general-purpose input ports until programmed otherwise. At power up, the Port D0 pin defaults to the RESET alternate function.

The pin configurations listed are preliminary and subject to change based on manufacturing limitations.

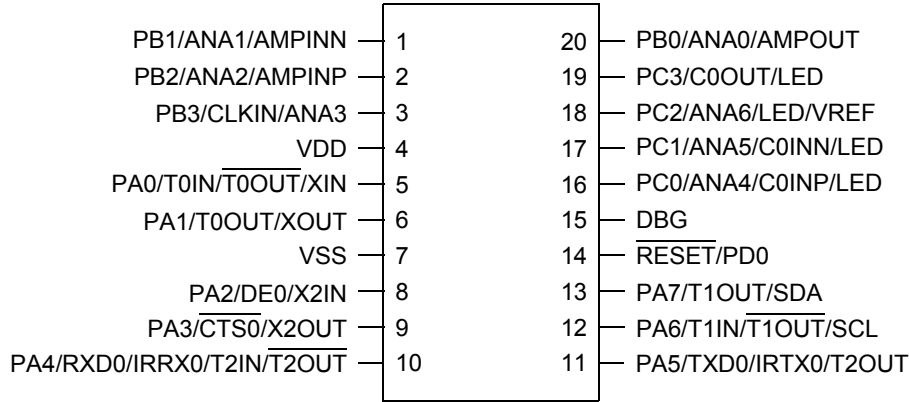


Figure 2. Z8F2480, Z8F1680 and Z8F0880 in 20-Pin SOIC, SSOP or PDIP Packages

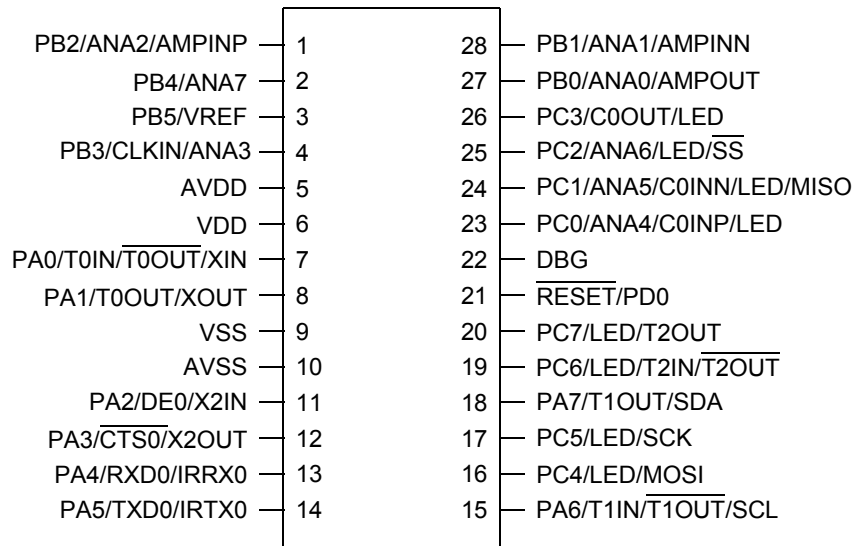
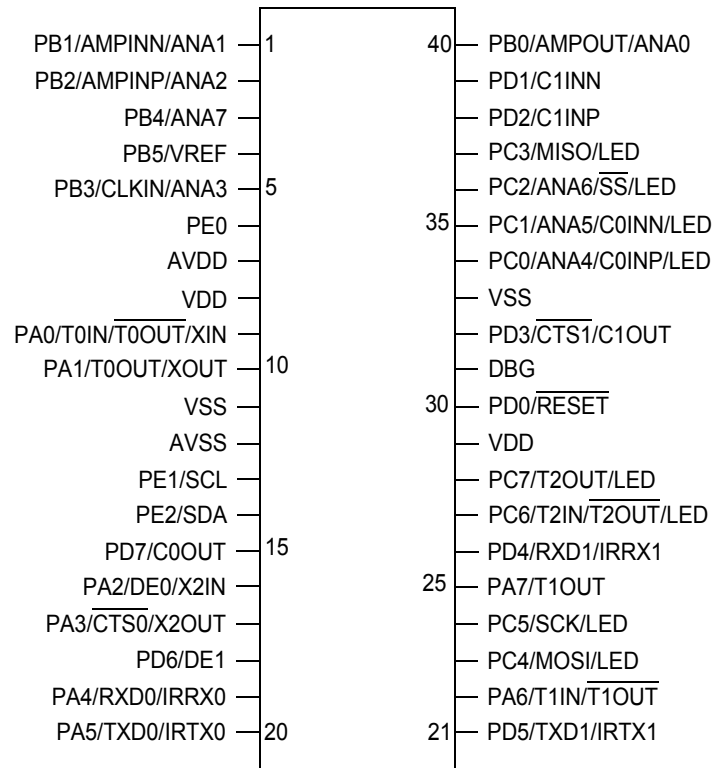


Figure 3. Z8F2480, Z8F1680 and Z8F0880 in 28-Pin SOIC, SSOP or PDIP Packages



**Figure 4. Z8F2480, Z8F1680 and Z8F0880 in 40-Pin Dual Inline Package (PDIP)**

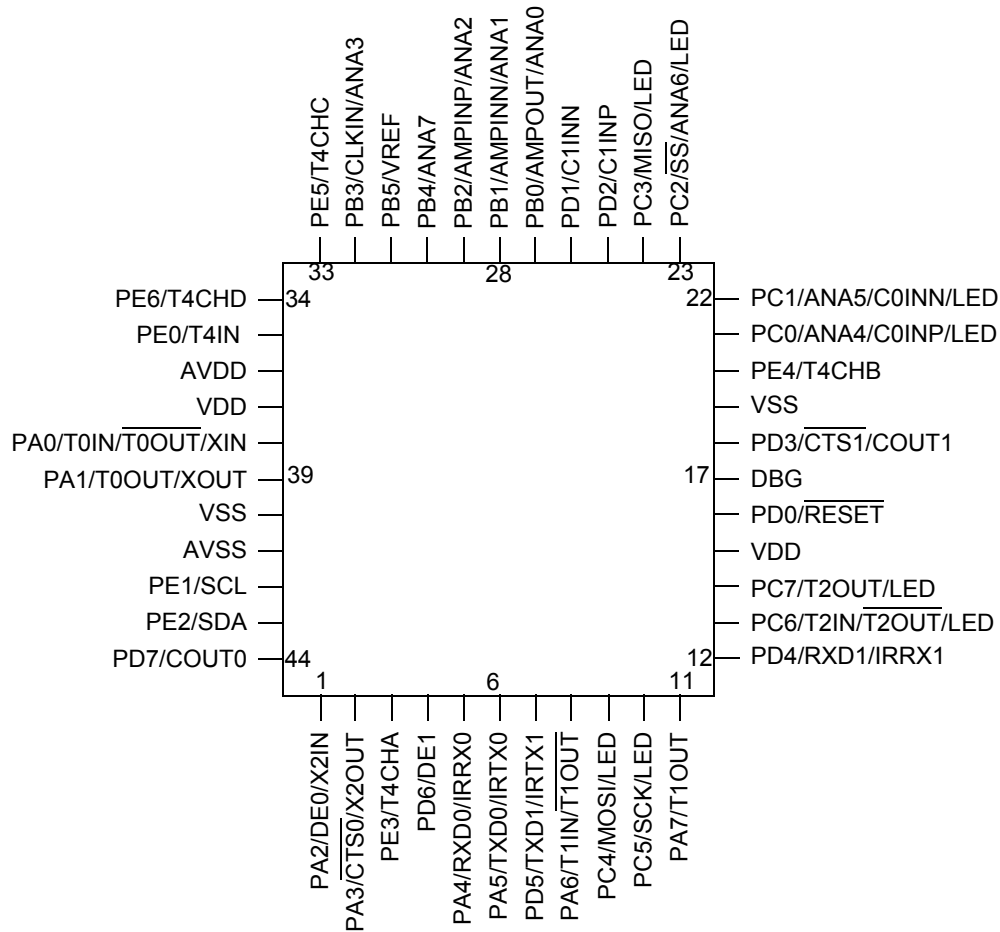


Figure 5. Z8F2480, Z8F1680 and Z8F0880 in 44-Pin Low-Profile Quad Flat Package (LQFP) or Quad Flat No Lead (QFN)

## Signal Descriptions

Table 3 on page 15 describes the Z8 Encore! XP F1680 Series signals. To determine the signals available for the specific package styles, see [Pin Configurations](#) on page 11.


**Table 3. Signal Descriptions**

Signal Mnemonic	I/O	Description
<b>General-Purpose I/O Ports A–E</b>		
PA[7:0]	I/O	Port A: These pins are used for general-purpose I/O.
PB[5:0]	I/O	Port B: These pins are used for GPIO.
PC[7:0]	I/O	Port C: These pins are used for GPIO.
PD[7:0]	I/O	Port D: These pins are used for GPIO. PD0 is output only.
PE[6:0]	I/O	Port E: These pins are used for GPIO.
<b>LIN-UART Controllers</b>		
TXD0/TXD1	O	Transmit Data 0-1: These signals are the transmit output from the UART0/1 and IrDA0/1.
RXD0/RXD1	I	Receive Data 0-1: These signals are the receive input for the UART0/1 and IrDA0/1.
CTS0/CTS1	I	Clear To Send 0-1: These signals are the flow control input for the UART0/1.
DE0/DE1	O	Driver Enable 0-1: These signals allow automatic control of external RS-485 drivers. These signals are approximately the inverse of the TXE (Transmit Empty) bit in the UART Status 0/1 register. The DE0/1 signal can be used to ensure the external RS-485 driver is enabled when data is transmitted by the UART0/1.
<b>I<sup>2</sup>C Controller</b>		
SCL	I/O	I <sup>2</sup> C Serial Clock: The I <sup>2</sup> C Master supplies this signal. If the Z8 Encore! XP F1680 Series is the I <sup>2</sup> C Master, this pin is an output and if it is I <sup>2</sup> C slave, this pin is an input. 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 <sup>2</sup> C and an external I <sup>2</sup> C Master/Slave. When the GPIO pin is configured for alternate function to enable the SDA function, this pin is open-drain.
<b>ESPI Controller</b>		
SS	I/O	Slave Select: This signal can be an output or an input. If the Z8 Encore! XP F1680 Series is the SPI master, this pin may be configured as the Slave Select output, and if it is the SPI slave, this pin is the input slave select.
SCK	I/O	SPI Serial Clock: The SPI master supplies this signal. If the Z8 Encore! XP F1680 Series is the SPI master, this pin is output and if it is the SPI slave, this pin is an input.
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.
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.

Table 3. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
<b>Timers</b>		
T0OUT/T1OUT/T2OUT	O	Timer Output 0–2: These signals are output from the timers.
T0OUT/T1OUT/T2OUT	O	Timer Complement Output 0–2: These signals are output from the timers in PWM Dual Output mode.
T0IN/T1IN/T2IN	I	Timer Input 0–2: These signals are used as the capture, gating, and counter inputs. The T0IN/T1IN/T2IN signal is multiplexed with T0OUT/T1OUT/T2OUT signals.
<b>Multi-Channel Timers</b>		
TACHA, TACHB, TACHC, TACHD	I/O	Multi-channel timer Input/Output: These signals function as Capture input or Compare output for channels CHA, CHB, CHC, and CHD.
T4IN	I	Multi-channel Timer clock input: This signal allows external input to serve as the clock source for the Multi-channel timer.
<b>Comparators</b>		
C0INP/C0INN, C1INP/C1INN	I	Comparator Inputs: These signals are positive and negative inputs to the comparator 0 and comparator 1.
C0OUT/C1OUT	O	Comparator Outputs: These are the output from the comparator 0 and the comparator 1.
<b>Analog</b>		
ANA[7:0]	I	Analog Port: These signals are used as inputs to the ADC. The ANA0, ANA1, and ANA2 pins can also access the inputs and outputs of the integrated Low-Power Operational Amplifier.
VREF	I/O	ADC reference voltage input.
<b>Low-Power Operational Amplifier</b>		
AMPINP/AMPINN	I	Low-Power Operational Amplifier Inputs: If enabled, these pins drive the positive and negative amplifier inputs respectively.
AMPOUT	O	Low-Power Operational Amplifier Output: If enabled, this pin is driven by the on-chip low-power operational amplifier.
<b>Oscillators</b>		
XIN	I	External Crystal Input: This is the input pin to the crystal oscillator. A crystal can be connected between the pin and the <b>XOUT</b> pin to form the oscillator. In addition, this pin is used with external RC networks or external clock drivers to provide the system clock.
XOUT	O	External Crystal Output: This pin is the output of the crystal oscillator. A crystal can be connected between it and the <b>XIN</b> pin to form the oscillator.

**Table 3. Signal Descriptions (Continued)**

Signal Mnemonic	I/O	Description
X2IN	I	Watch Crystal Input: This is the input pin to the low-power 32 kHz oscillator. A watch crystal can be connected between the X2IN and the X2OUT pin to form the oscillator.
X2OUT	O	Watch Crystal Output: This pin is the output from the low power 32 kHz oscillator. A watch crystal can be connected between the X2IN and the X2OUT pin to form the oscillator.
<b>Clock Input</b>		
CLKIN	I	Clock Input Signal: This pin may be used to input a TTL-level signal to be used as the system clock.
<b>LED Drivers</b>		
LED	O	Direct LED Drive Capability: All Port C pins have the capability to drive an LED without any other external components. These pins have programmable drive strengths set by the GPIO block.
<b>On-Chip Debugger</b>		
DBG	I/O	Debug: This signal is the control and data input and output of the On-Chip Debugger.
	<b>Caution:</b>	The DBG pin is open-drain and requires an external pull-up resistor to ensure proper operation.
<b>Reset</b>		
RESET	I/O	RESET: Generates a Reset when asserted (driven Low). Also serves as a Reset indicator; the Z8 Encore! XP forces this pin Low when in Reset. This pin is open-drain and features an enabled internal pull-up resistor.
<b>Power Supply</b>		
V <sub>DD</sub>	I	Digital Power Supply.
AV <sub>DD</sub>	I	Analog Power Supply.
V <sub>SS</sub>	I	Digital Ground.
AV <sub>SS</sub>	I	Analog Ground.
Note: The AV <sub>DD</sub> and AV <sub>SS</sub> signals are available only in 28-pin, 40-pin, and 44-pin packages.		

## Pin Characteristics

Table 4 on page 18 provides detailed information on the characteristics of each pin available on the Z8 Encore! XP F1680 Series 20-, 28-, 40-, and 44-pin devices. Data provided in Table 4 on page 18 is sorted alphabetically by the pin symbol mnemonic.

**Table 4. Pin Characteristics (20-, 28-, 40- and 44-pin Devices)**

<b>Symbol Mnemonic</b>	<b>Direction</b>	<b>Reset Direction</b>	<b>Active Low or Active High</b>	<b>Tristate Output</b>	<b>Internal Pull-up or Pull-down</b>	<b>Schmitt Trigger Input</b>	<b>Open Drain Output</b>	<b>5 V Tolerance</b>
AVDD	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AVSS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	NA
DBG	I/O	I	N/A	Yes	Yes	Yes	Yes	No
PA[7:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, Programmable	Yes, 5 V tolerant inputs unless pullups are enabled
PB[5:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, Programmable	Yes, 5 V tolerant inputs unless pullups are enabled
PC[7:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, Programmable	Yes, 5 V tolerant inputs unless pullups are enabled
PD[7:1]	I/O	I	N/A	Yes	Programmable Pull-up	Yes	Yes, Programmable	Yes, 5 V tolerant inputs unless pullups are enabled
PE[6:0]	I/O	I	N/A	Yes	Programmable Pull-up	Yes	Yes, Programmable	Yes, 5 V tolerant inputs unless pullups are enabled

Table 4. Pin Characteristics (20-, 28-, 40- and 44-pin Devices) (Continued)

Symbol Mnemonic	Direction	Reset Direction	Active Low or Active High	Tristate Output	Internal Pull-up or Pull-down	Schmitt Trigger Input	Open Drain Output	5 V Tolerance
RESET/PD0	I/O	I/O (defaults to $\overline{\text{RESET}}$ )	Low (in RESET mode)	Yes (PD0 only)	Programmable for PD0; always On for $\overline{\text{RESET}}$	Yes	Programmable for PD0; always On for $\overline{\text{RESET}}$	Yes, 5 V tolerant inputs unless pullups are enabled
VDD	N/A	N/A	N/A	N/A			N/A	N/A
VSS	N/A	N/A	N/A	N/A			N/A	N/A



# Address Space

## Overview

The eZ8 CPU can access the following three distinct address spaces:

- The [Register File](#) contains addresses for general-purpose registers, eZ8 CPU, peripherals, 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 contain data only.

These three address spaces are covered briefly in the following sections. For more details on the eZ8 CPU and its address space, refer to *eZ8 CPU User Manual (UM0077)* available on [www.zilog.com](http://www.zilog.com).

## Register File

The Register File address space in the Z8 Encore!<sup>®</sup> MCU is 4 KB (4096 bytes). The Register File is composed of two sections: Control Registers and general-purpose registers. When instructions are executed, registers defined as sources are read and registers defined as destinations are written. 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 4 KB Register File address space are reserved for control of the eZ8 CPU, on-chip peripherals, and the input/output ports. These registers are located at addresses `F00H` to `FFFH`. Some of the addresses within the 256 B control register sections are reserved (that is, unavailable). Reading from a reserved Register File address returns an undefined value. Writing to the reserved Register File addresses is not recommended and can produce unpredictable results.

The on-chip Register RAM always begins at address `000H` in the Register File address space. The Z8 Encore! XP F1680 Series devices contain 1 KB or 2 KB of on-chip Register 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.

In addition, Z8 Encore! XP F1680 Series devices contain 1 KB of on-chip Program RAM. Normally it is used as Program RAM and is present in the Program Memory address space (see [Program Memory](#)). However, it can also be used as additional Register RAM present in the Register File address space `800H–BFFH` (1 KB Program RAM, 2 KB Register RAM), or `400H–7FFH` (1 KB Program RAM, 1 KB Register RAM), if you do not need to

use this on-chip Program RAM to shadow Interrupt Service Routines (ISR). For details, see [PRAM\\_M](#) user option bit on page 270.

## Program Memory

The eZ8 CPU supports 64 KB of Program Memory address space. The Z8 Encore! XP F1680 Series devices contain 8 KB to 24 KB of on-chip Flash memory in the Program Memory address space, depending on the device.

In addition, Z8 Encore! XP F1680 Series devices contain up to 1 KB of on-chip Program RAM. The Program RAM is mapped in the Program Memory address space beyond the on-chip Flash memory. The Program RAM is entirely under user control and is meant to store interrupt service routines of high-frequency interrupts. Since interrupts bring the CPU out of low-power mode, it is important to ensure that interrupts that occur very often use as low current as possible. For battery operated systems, Program RAM based handling of high frequency interrupts provides power savings by keeping the Flash block disabled. Program RAM (PRAM) is optimized for low-current operation and can be easily boot-strapped with interrupt code at power up.

Reading from Program Memory addresses present outside the available Flash memory and PRAM addresses returns `FFH`. Writing to these unimplemented Program Memory addresses produces no effect. [Table 5](#) describes the Program Memory maps for the Z8 Encore! XP F1680 Series products.

**Table 5. Z8 Encore! XP F1680 Series Program Memory Maps**

Program Memory Address (Hex)	Function
<b>Z8F2480 Product</b>	
0000–0001	Flash Option Bits
0002–0003	Reset Vector
0004–0005	WDT Interrupt Vector
0006–0007	Illegal Instruction Trap
0008–0037	Interrupt Vectors*
0038–003D	Oscillator Fail Traps*
003E–5FFF	Program Flash
E000–E3FF	1 KB PRAM

**Table 5. Z8 Encore! XP F1680 Series Program Memory Maps (Continued)**

Program Memory Address (Hex)	Function
<b>Z8F1680 Product</b>	
0000–0001	Flash Option Bits
0002–0003	Reset Vector
0004–0005	WDT Interrupt Vector
0006–0007	Illegal Instruction Trap
0008–0037	Interrupt Vectors*
0038–003D	Oscillator Fail Traps*
003E–3FFF	Program Flash
E000–E3FF	1 KB PRAM
<b>Z8F0880 Product</b>	
0000–0001	Flash Option Bits
0002–0003	Reset Vector
0004–0005	WDT Interrupt Vector
0006–0007	Illegal Instruction Trap
0008–0037	Interrupt Vectors*
0038–003D	Oscillator Fail Traps*
003E–1FFF	Program Flash
E000–E3FF	1 KB PRAM
Note: * See <a href="#">Table 34</a> on page 72 for a list of interrupt vectors and traps.	

## Data Memory

The Z8 Encore! XP F1680 Series does not use the eZ8 CPU's 64 KB Data Memory address space.

## Flash Information Area

[Table 6](#) on page 24 describes the Z8 Encore! XP F1680 Series Flash Information Area. This 128 B Information Area is accessed by setting bit 7 of the Flash Page Select Register to 1. When access is enabled, the Flash Information Area is mapped into the Program Memory and overlays the 128 bytes at addresses FE00H to FF7FH. 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 Flash Information Area is read-only.

**Table 6. Z8 Encore! XP F1680 Series Flash Memory Information Area Map**

<b>Program Memory Address (Hex)</b>	<b>Function</b>
FE00–FE3F	Zilog Option Bits
FE40–FE53	Part Number: 20-character ASCII alphanumeric code Left justified and filled with FH
FE54–FE5F	Reserved
FE60–FE7F	Zilog Calibration Data (only use the first two bytes FE60 and FE61)
FE80–FFFF	Reserved

# Register Map

[Table 7](#) provides the address map for the Register File of Z8 Encore! XP F1680 Series devices. Not all devices and package styles in the Z8 Encore! XP F1680 Series support the ADC 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
<b>General Purpose RAM</b>				
<b>Z8F2480 Device</b>				
000–7FF	General-Purpose Register File RAM	—	XX	
800–EFF	Reserved *	—	XX	
<b>Z8F1680 Device</b>				
000–7FF	General-Purpose Register File RAM	—	XX	
800–EFF	Reserved *	—	XX	
<b>Z8F0880 Device</b>				
000–3FF	General-Purpose Register File RAM	—	XX	
400–EFF	Reserved *	—	XX	
Note: * = The Reserved space can be configured as General-Purpose Register File RAM depending on the <a href="#">User Option Bits</a> on page 267 and the on-chip PRAM size (see <a href="#">Ordering Information</a> on page 367). If the PRAM is programmed as General-Purpose Register File RAM on Reserved space, the starting address always begins immediately after the end of General-Purpose Register File RAM.				
<b>Special Purpose Registers</b>				
<b>Timer 0</b>				
F00	Timer 0 High Byte	T0H	00	107
F01	Timer 0 Low Byte	T0L	01	107
F02	Timer 0 Reload High Byte	T0RH	FF	108
F03	Timer 0 Reload Low Byte	T0RL	FF	108
F04	Timer 0 PWM0 High Byte	T0PWM0H	00	109
F05	Timer 0 PWM0 Low Byte	T0PWM0L	00	109
F06	Timer 0 Control 0	T0CTL0	00	110
F07	Timer 0 Control 1	T0CTL1	00	111
F20	Timer 0 PWM1 High Byte	T0PWM1H	00	109

**Table 7. Register File Address Map (Continued)**

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
F21	Timer 0 PWM1 Low Byte	T0PWM1L	00	110
F22	Timer 0 Control 2	T0CTL2	00	114
F23	Timer 0 Status	T0STA	00	115
F2C	Timer 0 Noise Filter Control	T0NFC	00	116
<b>Timer 1</b>				
F08	Timer 1 High Byte	T1H	00	107
F09	Timer 1 Low Byte	T1L	01	107
F0A	Timer 1 Reload High Byte	T1RH	FF	108
F0B	Timer 1 Reload Low Byte	T1RL	FF	108
F0C	Timer 1 PWM0 High Byte	T1PWM0H	00	109
F0D	Timer 1 PWM0 Low Byte	T1PWM0L	00	109
F0E	Timer 1 Control 0	T1CTL0	00	110
F0F	Timer 1 Control 1	T1CTL1	00	111
F24	Timer 1 PWM1 High Byte	T1PWM1H	00	109
F25	Timer 1 PWM1 Low Byte	T1PWM1L	00	110
F26	Timer 1 Control 2	T1CTL2	00	114
F27	Timer 1 Status	T1STA	00	115
F2D	Timer 1 Noise Filter Control	T1NFC	00	116
<b>Timer 2</b>				
F10	Timer 2 High Byte	T2H	00	107
F11	Timer 2 Low Byte	T2L	01	108
F12	Timer 2 Reload High Byte	T2RH	FF	108
F13	Timer 2 Reload Low Byte	T2RL	FF	108
F14	Timer 2 PWM0 High Byte	T2PWM0H	00	109
F15	Timer 2 PWM0 Low Byte	T2PWM0L	00	109
F16	Timer 2 Control 0	T2CTL0	00	110
F17	Timer 2 Control 1	T2CTL1	00	111
F28	Timer 2 PWM1 High Byte	T2PWM1H	00	109
F29	Timer 2 PWM1 Low Byte	T2PWM1L	00	110
F2A	Timer 2 Control 2	T2CTL2	00	114
F2B	Timer 2 Status	T2STA	00	115

**Table 7. Register File Address Map (Continued)**

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
F2E	Timer 2 Noise Filter Control	T2NFC	00	116
F2F–F3F	Reserved	—	XX	
<b>LIN UART 0</b>				
F40	LIN UART0 Transmit Data	U0TXD	XX	159
	LIN UART0 Receive Data	U0RXD	XX	159
F41	LIN UART0 Status 0—Standard UART Mode	U0STAT0	0000011Xb	160
	LIN UART0 Status 0—LIN Mode	U0STAT0	00000110b	161
F42	LIN UART0 Control 0	U0CTL0	00	165
F43	LIN UART0 Control 1—Multiprocessor Control	U0CTL1	00	166
	LIN UART0 Control 1—Noise Filter Control	U0CTL1	00	169
	LIN UART0 Control 1—LIN Control	U0CTL1	00	170
F44	LIN UART0 Mode Select and Status	U0MDSTAT	00	163
F45	UART0 Address Compare	U0ADDR	00	171
F46	UART0 Baud Rate High Byte	U0BRH	FF	172
F47	UART0 Baud Rate Low Byte	U0BRL	FF	172
<b>LIN UART 1</b>				
F48	LIN UART1 Transmit Data	U1TXD	XX	159
	LIN UART1 Receive Data	U1RXD	XX	159
F49	LIN UART1 Status 0—Standard UART Mode	U1STAT0	0000011Xb	160
	LIN UART1 Status 0—LIN Mode	U1STAT0	00000110b	161
F4A	LIN UART1 Control 0	U1CTL0	00	165
F4B	LIN UART1 Control 1—Multiprocessor Control	U1CTL1	00	166
	LIN UART1 Control 1—Noise Filter Control	U1CTL1	00	169
	LIN UART1 Control 1—LIN Control	U1CTL1	00	170
F4C	LIN UART1 Mode Select and Status	U1MDSTAT	00	163
F4D	UART1 Address Compare	U1ADDR	00	171
F4E	UART1 Baud Rate High Byte	U1BRH	FF	172
F4F	UART1 Baud Rate Low Byte	U1BRL	FF	172
<b>I<sup>2</sup>C</b>				
F50	I <sup>2</sup> C Data	I2CDATA	00	236
F51	I <sup>2</sup> C Interrupt Status	I2CISTAT	80	236

**Table 7. Register File Address Map (Continued)**

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
F52	I <sup>2</sup> C Control	I2CCTL	00	238
F53	I <sup>2</sup> C Baud Rate High Byte	I2CBRH	FF	239
F54	I <sup>2</sup> C Baud Rate Low Byte	I2CBRL	FF	240
F55	I <sup>2</sup> C State	I2CSTATE	02	241
F56	I <sup>2</sup> C Mode	I2CMODE	00	242
F57	I <sup>2</sup> C Slave Address	I2CSLVAR	00	245
F58-F5F	Reserved	—	XX	
<b>Enhanced Serial Peripheral Interface (ESPI)</b>				
F60	ESPI Data	ESPIDATA	XX	206
F61	ESPI Transmit Data Command	ESPIIDCR	00	206
F62	ESPI Control	ESPICTL	00	207
F63	ESPI Mode	ESPIIMODE	00	209
F64	ESPI Status	ESPISTAT	01	210
F65	ESPI State	ESPISTATE	00	211
F66	ESPI Baud Rate High Byte	ESPIBRH	FF	211
F67	ESPI Baud Rate Low Byte	ESPIBRL	FF	211
F68-F6F	Reserved	—	XX	
<b>Analog-to-Digital Converter (ADC)</b>				
F70	ADC Control 0	ADCCTL0	00	184
F71	ADC Raw Data High Byte	ADCRD_H	80	185
F72	ADC Data High Byte	ADCD_H	XX	186
F73	ADC Data Low Bits	ADCD_L	XX	186
F74	ADC Sample Settling Time	ADCSST	FF	187
F75	Sample Time	ADCST	XX	187
F76	ADC Clock Prescale Register	ADCCP	00	188
F77-F7F	Reserved	—	XX	
<b>Low-Power Control</b>				
F80	Power Control 0	PWRCTL0	80	47
F81	Reserved	—	XX	
<b>LED Controller</b>				
F82	LED Drive Enable	LEDEN	00	69

**Table 7. Register File Address Map (Continued)**

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
F83	LED Drive Level High Bit	LEDLVLH	00	70
F84	LED Drive Level Low Bit	LEDLVLL	00	70
F85	Reserved	—	XX	
<b>Oscillator Control</b>				
F86	Oscillator Control 0	OSCCTL0	A0	306
F87	Oscillator Control 1	OSCCTL1	00	308
F88–F8F	Reserved			
<b>Comparator 0</b>				
F90	Comparator 0 Control	CMP0	14	248
<b>Comparator 1</b>				
F91	Comparator 1 Control	CMP1	14	249
F92–F9F	Reserved	—	XX	
<b>Multi-Channel Timer</b>				
FA0	MCT High Byte	MCTH	00	127
FA1	MCT Low Byte	MCTL	00	127
FA2	MCT Reload High Byte	MCTRH	FF	128
FA3	MCT Reload Low Byte	MCTRL	FF	128
FA4	MCT Sub-Address	MCTSA	XX	129
FA5	MCT SubRegister 0	MCTSR0	XX	129
FA6	MCT SubRegister 1	MCTSR1	XX	129
FA7	MCT SubRegister 2	MCTSR2	XX	129
FA8–FBF	Reserved	—	XX	
<b>Interrupt Controller</b>				
FC0	Interrupt Request 0	IRQ0	00	75
FC1	IRQ0 Enable High Bit	IRQ0ENH	00	78
FC2	IRQ0 Enable Low Bit	IRQ0ENL	00	79
FC3	Interrupt Request 1	IRQ1	00	76
FC4	IRQ1 Enable High Bit	IRQ1ENH	00	79
FC5	IRQ1 Enable Low Bit	IRQ1ENL	00	80
FC6	Interrupt Request 2	IRQ2	00	77
FC7	IRQ2 Enable High Bit	IRQ2ENH	00	81

**Table 7. Register File Address Map (Continued)**

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FC8	IRQ2 Enable Low Bit	IRQ2ENL	00	81
FC9–FCC	Reserved	—	XX	
FCD	Interrupt Edge Select	IRQES	00	82
FCE	Shared Interrupt Select	IRQSS	00	82
FCF	Interrupt Control	IRQCTL	00	83
<b>GPIO Port A</b>				
FD0	Port A Address	PAADDR	00	62
FD1	Port A Control	PACTL	00	63
FD2	Port A Input Data	PAIN	XX	64
FD3	Port A Output Data	PAOUT	00	64
<b>GPIO Port B</b>				
FD4	Port B Address	PBADDR	00	62
FD5	Port B Control	PBCTL	00	63
FD6	Port B Input Data	PBIN	XX	64
FD7	Port B Output Data	PBOUT	00	64
<b>GPIO Port C</b>				
FD8	Port C Address	PCADDR	00	62
FD9	Port C Control	PCCTL	00	63
FDA	Port C Input Data	PCIN	XX	64
FDB	Port C Output Data	PCOUT	00	64
<b>GPIO Port D</b>				
FDC	Port D Address	PDADDR	00	62
FDD	Port D Control	PDCTL	00	63
FDE	Port D Input Data	PDIN	XX	64
FDF	Port D Output Data	PDOUT	00	64
<b>GPIO Port E</b>				
FE0	Port E Address	PEADDR	00	62
FE1	Port E Control	PECTL	00	63
FE2	Port E Input Data	PEIN	XX	64
FE3	Port E Output Data	PEOUT	00	64
FE4–FEF	Reserved	—	XX	

**Table 7. Register File Address Map (Continued)**

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
Reset				
FF0	Reset Status	RSTSTAT	XX	42
FF1	Reserved	—	XX	
Watchdog Timer				
FF2	Watchdog Timer Reload High Byte	WDTH	FF	140
FF3	Watchdog Timer Reload Low Byte	WDTL	FF	140
FF4–FF5	Reserved	—	XX	
Trim Bit Control				
FF6	Trim Bit Address	TRMADR	00	269
FF7	Trim Data	TRMDR	XX	269
Flash Memory Controller				
FF8	Flash Control	FCTL	00	262
	Flash Status	FSTAT	00	262
FF9	Flash Page Select	FPS	00	263
	Flash Sector Protect	FPROT	00	264
FFA	Flash Programming Frequency High Byte	FFREQH	00	264
FFB	Flash Programming Frequency Low Byte	FFREQL	00	265
eZ8 CPU				
FFC	Flags	—	XX	Refer to eZ8 CPU User Manual (UM0077 )
FFD	Register Pointer	RP	XX	
FFE	Stack Pointer High Byte	SPH	XX	
FFF	Stack Pointer Low Byte	SPL	XX	
Note: XX=Undefined				



# Reset, Stop Mode Recovery, and Low-Voltage Detection

## Overview

The Reset Controller within the Z8 Encore! XP F1680 Series device controls Reset and Stop Mode Recovery operation and provides indication of low-voltage supply conditions. During the operation, the following events cause a Reset:

- Power-On Reset (POR)
- Voltage Brownout (VBO) protection
- Watchdog Timer (WDT) timeout (when configured by the WDT\_RES Flash Option Bit to initiate a Reset)
- External  $\overline{\text{RESET}}$  pin assertion (when the alternate RESET function is enabled by the GPIO register)
- On-Chip Debugger initiated Reset (OCDCTL[0] set to 1)

When the device is in STOP mode, a Stop Mode Recovery is initiated by each of the following:

- Watchdog Timer timeout
- GPIO Port input pin transition on an enabled Stop Mode Recovery source
- Interrupt from a timer or comparator enabled for STOP mode operation

The low-voltage detection circuitry on the device features the following:

- The low-voltage detection threshold level is user-defined
- It generates an interrupt when the supply voltage drops below a user-defined level

## Reset Types

The Z8 Encore! XP F1680 Series provides various types of Reset operation. Stop Mode Recovery is considered a form of Reset. [Table 8](#) on page 34 lists the types of Reset and their operating characteristics. The System Reset is longer, if the external crystal oscillator is enabled by the Flash option bits allowing additional time for oscillator start-up.

**Table 8. Reset and STOP Mode Recovery Characteristics and Latency**

Reset Type	Reset Characteristics and Latency		
	Control Registers	eZ8 CPU	Reset Latency (Delay)
System Reset (non-POR Reset)	Reset (as applicable)	Reset	68 Internal Precision Oscillator Cycles after IPO starts up
System Reset (POR Reset)	Reset (as applicable)	Reset	68 Internal Precision Oscillator Cycles + 50 ms Wait time
System Reset with Crystal Oscillator Enabled	Reset (as applicable)	Reset	568–10068 Internal Precision Oscillator Cycles after IPO starts up, see <a href="#">EXTLTMG</a> description in user option bit for details.
Stop Mode Recovery	Unaffected, except RSTSTAT and OSCCTL registers	Reset	4 Internal Precision Oscillator Cycles after IPO starts up

During a System Reset or Stop Mode Recovery, the Internal Precision Oscillator (IPO) requires 4  $\mu$ s to start up. When the reset type is a System Reset, the Z8 Encore! XP F1680 Series device is held in Reset for 68 IPO cycles. If the crystal oscillator is enabled in Flash option bits, the Reset period is increased to 568–10068 IPO cycles. For more details, see [EXTLTMG](#) description in user option bit. When the reset type is a Stop Mode Recovery, the Z8 Encore! XP F1680 Series device goes to normal mode immediately after 4 IPO cycles. The total Stop Mode Recovery delay is less than 6  $\mu$ s. When a Reset occurs due to a VBO condition, this delay is measured from the time the supply voltage first exceeds the VBO level (discussed later in this chapter). When a Reset occurs due to a POR condition, this delay is measured from the time that the supply voltage first exceeds the POR level. If the external pin reset remains asserted at the end of the Reset period, the device remains in reset until the pin is deasserted.

► **Note:** *After a Stop Mode Recovery, the external crystal oscillator is unstable. Use software to wait until it is stable before you can use it as main clock.*

At the beginning of Reset, all GPIO pins are configured as inputs with pull-up resistor disabled, except PD0 that is shared with the Reset pin. On Reset, the Port D0 pin is configured as a bidirectional open-drain Reset. The pin is internally driven Low during port reset after which the user code may reconfigure this pin as a general-purpose output.

During Reset, the eZ8 CPU and on-chip peripherals are idle; however, the on-chip crystal oscillator and WDT oscillator continue to function.

On Reset, control registers within the Register File that 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 not initialized and undefined following 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.

Because the control registers are reinitialized by a System Reset, the system clock after reset is always the 11 MHz IPO. User software must reconfigure the oscillator control block such that the correct system clock source is enabled and selected.

## Reset Sources

Table 9 lists the possible sources of a System Reset.

**Table 9. Reset Sources and Resulting Reset Type**

Operating Mode	Reset Source	Special Conditions
NORMAL or HALT mode	Power-On Reset	Reset delay begins after supply voltage exceeds POR level
	Voltage Brownout	Reset delay begins after supply voltage exceeds VBO level
	Watchdog Timer timeout when configured for Reset	None
	RESET pin assertion	All reset pulses less than three system clocks in width are ignored, see <a href="#">Electrical Characteristics</a> on page 337
	On-Chip Debugger initiated Reset (OCDCTL[0] set to 1)	System Reset, except the OCD is unaffected by reset
STOP mode	Power-On Reset	Reset delay begins after supply voltage exceeds POR level
	Voltage Brownout	Reset delay begins after supply voltage exceeds VBO level
	RESET pin assertion	All reset pulses less than the specified analog delay is ignored, see <a href="#">Electrical Characteristics</a> on page 337
	DBG pin driven Low	None

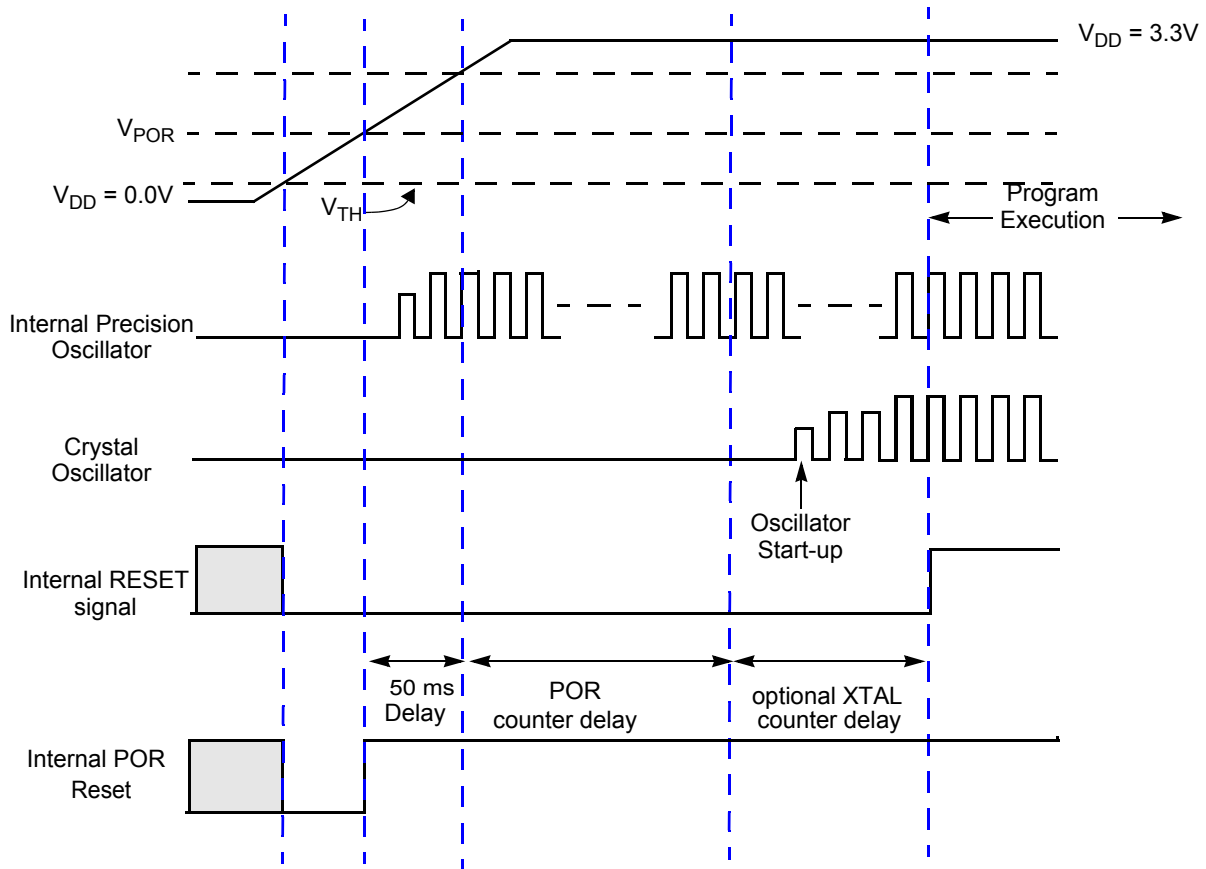
## Power-On Reset

Each device in the Z8 Encore! XP F1680 Series contains an internal Power-On Reset (POR) circuit. The POR circuit monitors the supply voltage and holds the whole device in the Reset state until the supply voltage reaches a safe circuit operating level when the device is powered on.

After power on, the POR circuit keeps idle until the supply voltage drops below  $V_{TH}$  voltage. [Figure 7](#) on page 37 illustrates this POR timing.


After the Z8 Encore! XP F1680 Series device exits the POR state, the eZ8 CPU fetches the Reset vector. Following this POR, the POR/VBO status bit in the Reset Status Register is set to 1.

For the POR threshold voltage ( $V_{POR}$ ) and POR start voltage  $V_{TH}$ , see [Electrical Characteristics](#) on page 337.



#### Notes

1. Not to Scale
2. Internal Reset and POR Reset are low active

 undefined

**Figure 6. Power-On Reset Operation**

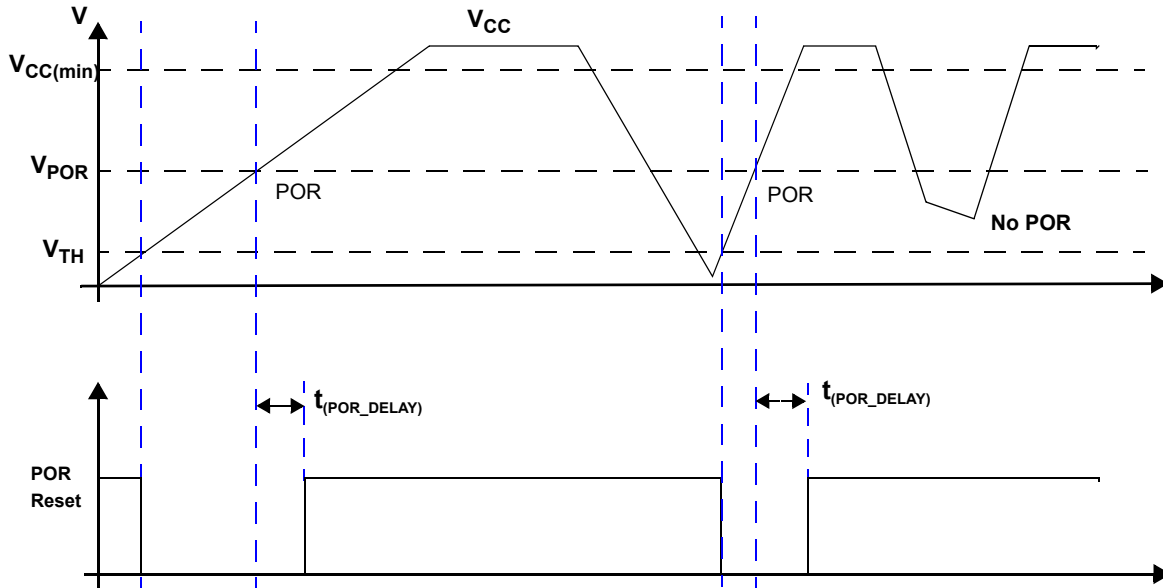


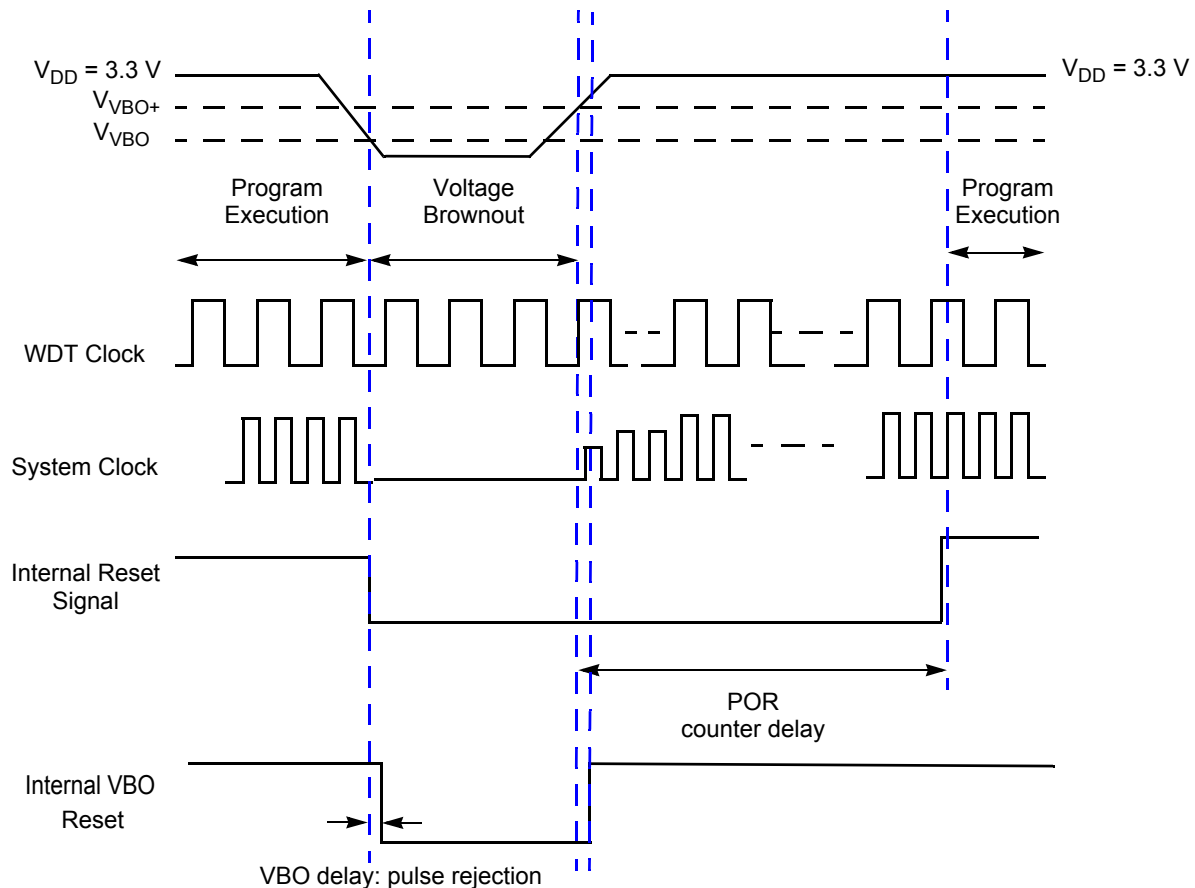
Figure 7. Power-On Reset Timing

## Voltage Brownout Reset

The Z8 Encore! XP F1680 Series provides a VBO Reset feature for low-voltage protection. The VBO circuit has a preset threshold voltage ( $V_{VBO}$ ) with a hysteresis of  $V_{hys}$ . The VBO circuit will monitor the power supply voltage if the VBO is enabled. When VBO Reset circuit detects the power supply voltage falls below the threshold voltage  $V_{VBO}$ , the VBO resets the device by pulling the POR Reset from 1 to 0. The VBO will hold the POR Reset until the power supply voltage goes above the  $V_{VBO} + V_{hys}$ , and then the VBO Reset is released. The device progresses through a full System Reset sequence, just like power-on case. Following this System Reset sequence, the POR/VBO status bit in the Reset Status (RSTSTAT) register is set to 1. [Figure 8](#) on page 38 illustrates VBO Reset operation.

For VBO threshold voltages ( $V_{VBO}$ ) and VBO hysteresis ( $V_{hys}$ ), see [Electrical Characteristics](#) on page 337.

The VBO circuit is either enabled or disabled during STOP mode. VBO circuit operation is controlled by both the VBO\_AO Flash Option Bit and the Power Control Register bit4. For more details, see [Flash Option Bits](#) on page 267 and [Power Control Register Definitions](#) on page 47.



Note: Not to Scale

Figure 8. Voltage Brownout Reset Operation

## Watchdog Timer Reset

If the device is in NORMAL or STOP mode, WDT initiates a System Reset at timeout if the `WDT_RES` Flash Option Bit is programmed to 1. This is the unprogrammed state of the `WDT_RES` Flash Option Bit. If the bit is programmed to 0, it configures the WDT to cause an interrupt, not a System Reset at timeout. The WDT status bit in the Reset Status Register is set to signify that the reset was initiated by the WDT.

## External Reset Input

The `RESET` pin has a Schmitt-triggered input and an internal pull-up resistor. When the `RESET` pin is asserted for a minimum of four system clock cycles, the device progresses through the System Reset sequence. Because of the possible asynchronicity of the system

clock and reset signals, the required reset duration may be as short as three clock periods and as long as four. A reset pulse three clock cycles in duration might trigger a Reset; a pulse four cycles in duration always triggers a Reset.

While the  $\overline{\text{RESET}}$  input pin is asserted Low, the Z8 Encore! XP F1680 Series devices remain in the Reset state. If the  $\overline{\text{RESET}}$  pin is held Low beyond the System Reset timeout, the device exits the Reset state on the system clock rising edge following  $\overline{\text{RESET}}$  pin deassertion. Following a System Reset initiated by the external  $\overline{\text{RESET}}$  pin, the  $\text{EXT}$  status bit in the RSTSTAT Register is set to 1.

## External Reset Indicator

During System Reset or when enabled by the GPIO logic (see [Port A–E Control Registers](#) on page 63), the  $\overline{\text{RESET}}$  pin functions as an open-drain (active Low) reset mode indicator in addition to the input functionality. This Reset output feature allows a Z8 Encore! XP F1680 Series device to reset other components to which it is connected, even if that reset is caused by internal sources such as POR, VBO, or WDT events.

After an internal Reset event occurs, the internal circuitry begins driving the  $\overline{\text{RESET}}$  pin Low. The  $\overline{\text{RESET}}$  pin is held Low by the internal circuitry until the appropriate delay listed in [Table 8](#) on page 34 has elapsed.

## On-Chip Debugger Initiated Reset

A POR can be initiated using the OCD by setting the  $\text{RST}$  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  $\text{RST}$  bit automatically clears during the system reset. Following the System Reset the  $\text{POR}$  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 STOP mode information, see [Low-Power Modes](#) on page 45. During Stop Mode Recovery, the CPU is held in reset for 4 IPO cycles.

Stop Mode Recovery does not affect On-chip registers other than the Reset Status (RSTSTAT) register and the Oscillator Control register (OSCCTL). After any Stop Mode Recovery, the IPO is enabled and selected as the system clock. If another system clock source is required or IPO disabling is required, the Stop Mode Recovery code must reconfigure the oscillator control block such that the correct system clock source is enabled and selected.

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  $\text{STOP}$  bit in the Reset Status 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	Watchdog Timer timeout when configured for Reset	Stop Mode Recovery
	Watchdog Timer timeout when configured for interrupt	Stop Mode Recovery followed by interrupt (if interrupts are enabled)
	Interrupt from Timer enabled for STOP mode operation	Stop Mode Recovery followed by interrupt (if interrupts are enabled)
	Interrupt from Comparator enabled for STOP mode operation	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
	Assertion of external <b>RESET</b> Pin	System Reset
	Debug Pin driven Low	System Reset

## Stop Mode Recovery Using Watchdog Timer Timeout

If the WDT times out during STOP mode, the device undergoes a Stop Mode Recovery sequence. In the Reset Status register, the **WDT** and **STOP** bits are set to 1.

If the WDT is configured to generate an interrupt on timeout and the Z8 Encore! XP F1680 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 Timer Interrupt

If a Timer with 32 K crystal enabled for STOP mode operation interrupts during STOP mode, the device undergoes a Stop Mode Recovery sequence. In the Reset Status Register, the **STOP** bit is set to 1. If the Z8 Encore! XP F1680 Series device is configured to respond to interrupts, the eZ8 CPU services the Timer interrupt request following the normal Stop Mode Recovery sequence.

## Stop Mode Recovery Using Comparator Interrupt

If Comparator enabled for STOP mode operation interrupts during STOP mode, the device undergoes a Stop Mode Recovery sequence. In the Reset Status Register, the STOP bit is set to 1. If the Z8 Encore! XP F1680 Series device is configured to respond to interrupts, the eZ8 CPU services the comparator interrupt request following the normal Stop Mode Recovery sequence.

## Stop Mode Recovery Using GPIO Port Pin Transition

Each of the GPIO Port pins may be configured as a Stop Mode Recovery input source. On any GPIO pin enabled as a Stop Mode Recovery source, a change in the input pin value (from high to low or from low to high) initiates Stop Mode Recovery. In the Reset Status 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 till the end of the Stop Mode Recovery delay. As a result, short pulses on the Port pin can initiate Stop Mode Recovery without being written to the Port Input Data register or without initiating an interrupt (if enabled for that pin).*

## Stop Mode Recovery Using External RESET Pin

When the Z8 Encore! XP F1680 Series device is in STOP mode and the external RESET pin is driven Low, a System Reset occurs. Because of a glitch filter operating on the RESET pin, the low pulse must be greater than the minimum width specified, or it is ignored. For details, see [Electrical Characteristics](#) on page 337.

## Low-Voltage Detection

In addition to the VBO Reset described earlier, it is also possible to generate an interrupt when the supply voltage drops below a user-selected value. For more details on the available Low-Voltage Detection (LVD) threshold levels, see [Trim Bit Address 0000H](#) on page 273.

When the supply voltage drops below the LVD threshold, the LVD bit of the RSTSTAT Register is set to 1. This bit remains 1 until the low-voltage condition elapses. Reading or writing this bit does not clear it. The LVD circuit can also generate an interrupt when enabled ([Interrupt Vectors and Priority](#) on page 74). The LVD is not latched, so enabling the interrupt is the only way to guarantee detection of a transient low-voltage event.

The LVD circuit is either enabled or disabled by the Power Control Register bit 4. For more details, see [Power Control Register Definitions](#) on page 47.

## Reset Register Definitions

### Reset Status Register

The Reset Status (RSTSTAT) register is a read-only register which indicates the source of the most recent Reset event, a Stop Mode Recovery event, and a WDT timeout. Reading this register resets the upper 4 bits to 0.

This register shares its address with the Reset Status register, which is write-only (see [Table 11](#)).

**Table 11. Reset Status Register (RSTSTAT)**

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

Reset or Stop Mode Recovery Event	POR	STOP	WDT	EXT
Power-On Reset or VBO Reset	1	0	0	0
Reset using RESET pin assertion	0	0	0	1
Reset using Watchdog Timer timeout	0	0	1	0
Reset using the On-Chip Debugger (OCTCTL[1] set to 1)	1	0	0	0
Reset from STOP mode using DBG Pin driven Low	1	0	0	0
Stop Mode Recovery using GPIO pin transition	0	1	0	0
Stop Mode Recovery using Watchdog Timer timeout	0	1	1	0

#### **POR/VBO—Power-On initiated VBO Reset or general VBO Reset Indicator**

If this bit is set to 1, a POR or VBO Reset event occurs. This bit is reset to 0, if a WDT timeout 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 occurs. If the STOP and WDT bits are both set to 1, the Stop Mode Recovery occurs because of a WDT timeout. If the STOP bit is 1 and the WDT bit is 0, the Stop Mode Recovery is not caused by a WDT timeout. This bit is reset by Power-On Reset or WDT timeout that occurred while not in STOP mode. Reading this register also resets this bit.

**WDT—Watchdog Timer Timeout Indicator**

If this bit is set to 1, a WDT timeout occurs. A POR resets this pin. A Stop Mode Recovery from a change in an input pin also resets this bit. Reading this register resets this bit. This Read must occur before clearing the WDT interrupt.

**EXT—External Reset Indicator**

If this bit is set to 1, a Reset initiated by the external  $\overline{\text{RESET}}$  pin occurs. A POR or a Stop Mode Recovery from a change in an input pin resets this bit. Reading this register resets this bit.

Reserved—Must be 0.

**LVD—Low-Voltage Detection Indicator**

If this bit is set to 1 the current state of the supply voltage is below the low-voltage detection threshold. This value is not latched but is a real-time indicator of the supply voltage level.



# Low-Power Modes

## Overview

The Z8 Encore! XP F1680 Series products have power-saving features. The highest level of power reduction is provided by the STOP mode. The next lower level of power reduction is provided by the HALT mode.

Further power savings can be implemented by disabling individual peripheral blocks while in NORMAL mode.

## STOP Mode

Executing the eZ8 CPU's Stop instruction places the device into STOP mode. In STOP mode, the operating characteristics are:

- Primary crystal oscillator and internal precision oscillator are stopped; XIN and XOUT (if previously enabled) are disabled and PA0/PA1 reverts to the states programmed by the GPIO registers.
- System clock is stopped.
- eZ8 CPU is stopped.
- Program counter (PC) stops incrementing.
- Watchdog Timer's internal RC oscillator continues operating if enabled by the Oscillator Control Register.
- If enabled, the Watchdog Timer (WDT) logic continues operating.
- If enabled, the 32 kHz secondary oscillator continues operating.
- If enabled for operation in STOP Mode, the Timer logic continues to operate with 32 kHz secondary oscillator as the Timer clock source.
- If enabled for operation in STOP mode by the associated Flash Option Bit, the VBO protection circuit continues operating. The low-voltage detection circuit continues to operate if being enabled by the Power Control Register to do so.
- Low-Power Operational Amplifier and comparator continue to operate if being enabled by the Power Control Register to do so.
- All other on-chip peripherals are idle.

To minimize current in STOP mode, all GPIO pins which are configured as digital inputs must be driven to one of the supply rails ( $V_{DD}$  or GND). The device is brought out of STOP mode using Stop Mode Recovery. For more details on Stop Mode Recovery, see [Reset, Stop Mode Recovery, and Low-Voltage Detection](#) on page 33.

## HALT Mode

Executing the eZ8 CPU's HALT instruction places the device into HALT mode. In HALT mode, the operating characteristics are:

- Primary oscillator is enabled and continues to operate
- System clock is enabled and continues to operate
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- WDT's internal RC oscillator continues to operate
- If enabled, the WDT continues to operate
- If enabled, the 32 kHz secondary oscillator for Timers 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
- Watchdog Timer timeout (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 ( $V_{DD}$  or GND).

## Peripheral-Level Power Control

In addition to the STOP and HALT modes, it is possible to disable each peripheral on each of the Z8 Encore! XP F1680 Series devices. Disabling a given peripheral minimizes its power consumption.

## Power Control Register Definitions

### Power Control Register 0

Each bit of the following registers disables a peripheral block, either by gating its system clock input or by removing power from the block.

The default state of the low-power operational amplifier is OFF. To use the low-power operational amplifier, clear the TRAM bit by turning it ON. Clearing this bit might interfere with normal ADC measurements on ANA0 (the LPO output). This bit enables the amplifier even in STOP mode. If the amplifier is not required in STOP mode, disable it. Failure to perform this results in STOP mode currents greater than specified.

► **Note:** *This register is only reset during a POR sequence. Other System Reset events do not affect it.*

**Table 12. Power Control Register 0 (PWRCTL0)**

BITS	7	6	5	4	3	2	1	0
FIELD	TRAM	Reserved		LVD/VBO	TEMP	Reserved	COMP0	COMP1
RESET	1	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F80H							

#### TRAM— Low-Power Operational Amplifier Disable

0 = Low-Power Operational Amplifier is enabled (this applies even in STOP mode).

1 = Low-Power Operational Amplifier is disabled.

Reserved—Must be 0.

#### LVD/VBO— Low-Voltage Detection/Voltage Brownout Detector Disable

0 = LVD/VBO Enabled

1 = LVD/VBO Disabled

The LVD and VBO circuits are enabled or disabled separately to minimize power consumption in low-power modes. The LVD is controlled by the LVD/VBO bit only in all modes. The VBO is set by the LVD/VBO bit and the VBO\_AO bit of Flash Option bit at Program Memory Address 0000H. [Table 13](#) on page 48 lists the setup condition for LVD and VBO circuits in different operation modes.

#### TEMP—Temperature Sensor Disable

0 = Temperature Sensor Enabled

1 = Temperature Sensor Disabled

**COMP0—Comparator 0 Disable**

0 = Comparator 0 is Enabled (this applies even in STOP Mode)

1 = Comparator 0 is Disabled

**COMP1—Comparator 1 Disable**

0 = Comparator 1 is Enabled (this applies even in STOP Mode)

1 = Comparator 1 is Disabled

**Table 13. Setup Condition for LVD and VBO Circuits in Different Operation Modes**

LVD/VBO Bit Setup <sup>1</sup>		LVD/VBO Bit = "0" or enabled		LVD/VBO Bit = "1" or disabled	
Operation Mode		ACTIVE/HALT Mode	STOP Mode	ACTIVE/HALT Mode	STOP Mode
VBO_AO Bit Setup	VBO_AO="1" or enabled <sup>2</sup>	LVD "ON"		LVD "OFF"	
		VBO "ON"		VBO "ON"	
	VBO_AO="0" or disabled <sup>3</sup>	LVD "ON"		LVD "OFF"	
		VBO "ON"	VBO "OFF"	VBO "OFF"	VBO "OFF"

- **Notes:**
1. The LVD can be turned ON or OFF by the LVD/VBO bit in any mode.
  2. When VBO\_AO Bit is enabled, VBO is always ON for all modes no matter the setting of LVD/VBO Bit.
  3. When VBO\_AO Bit is disabled, VBO circuit is always OFF in STOP mode no matter the setting of LVD/VBO Bit. And VBO can be turned On or OFF by the LVD/VBO Bit in ACTIVE and HALT modes.

# General-Purpose Input/Output

## Overview

The Z8 Encore! XP F1680 Series product supports a maximum of 37 port pins (Ports A–E) for general-purpose input/output (GPIO) operations. Each port contains control and data registers. The GPIO control registers 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. In addition, the Port C pins are capable of direct LED drive at programmable drive strengths.

## GPIO Port Availability by Device

[Table 14](#) lists the port pins available with each device and package type.

**Table 14. Port Availability by Device and Package Type**

Devices	Package	10-bit ADC	SPI	Port A	Port B	Port C	Port D	Port E	Total I/O
Z8F2480PH, Z8F2480HH; Z8F2480SH; Z8F1680PH, Z8F1680HH; Z8F1680SH; Z8F0880PH, Z8F0880HH; Z8F0880SH	20-pin PDIP SOIC SSOP	7	0	[7:0]	[3:0]	[3:0]	[0]	—	17
Z8F2480PJ, Z8F2480SJ; Z8F2480HJ; Z8F1680PJ, Z8F1680SJ; Z8F1680HJ; Z8F0880PJ, Z8F0880SJ; Z8F0880HJ	28-pin PDIP SOIC SSOP	8	1	[7:0]	[5:0]	[7:0]	[0]	—	23
Z8F2480PM, Z8F1680PM, Z8F0880PM	40-pin PDIP	8	1	[7:0]	[5:0]	[7:0]	[7:0]	[2:0]	33
Z8F2480AN, Z8F2480QN; Z8F1680AN, Z8F1680QN; Z8F0880AN, Z8F0880QN	44-pin LQFP QFN	8	1	[7:0]	[5:0]	[7:0]	[7:0]	[6:0]	37

## Architecture

Figure 9 illustrates a simplified block diagram of a GPIO port pin and does not illustrate the ability to accommodate alternate functions and variable port current drive strength.

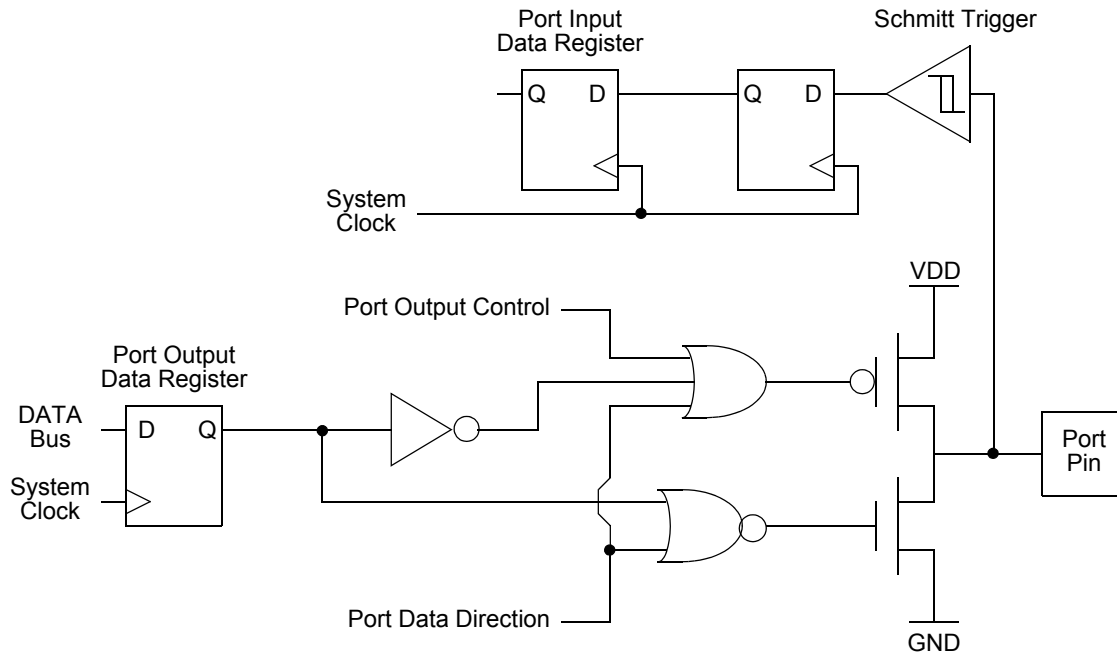


Figure 9. GPIO Port Pin Block Diagram

## GPIO Alternate Functions

Many GPIO port pins are used for GPIO and to access the on-chip peripheral functions like the timers and serial-communication devices. The Port A–E Alternate Function subregisters configure these pins for either GPIO or alternate function operation. When a pin is configured for alternate function, control of port-pin direction (input/output) is passed from Port A–E Data Direction registers to the alternate functions assigned to this pin. Table 15 on page 52, Table 16 on page 54, and Table 17 on page 57 list the alternate functions possible with each port pin for every package. The alternate function associated at a pin is defined through alternate function sets subregisters AFS1 and AFS2.

The crystal oscillator and the 32 kHz secondary oscillator functionalities are not controlled by the GPIO block. When the crystal oscillator or the 32 kHz secondary oscillator is enabled in the oscillator control block, the GPIO functionality of PA0 and PA1, or PA2 and PA3, is overridden. In such a case, those pins function as input and output for the crystal oscillator.

## Direct LED Drive

The Port C pins provide a current synched output capable of driving an LED without requiring an external resistor. The output synchs current at programmable levels of 3 mA, 7 mA, 13 mA, and 20 mA. This mode is enabled through the Alternate Function subregister AFS1 and is programmable through the LED control registers. For proper function, the LED anode must be connected to  $V_{DD}$  and the cathode to the GPIO pin.

Using all Port C pins in LED drive mode with maximum current may result in excessive total current. For the maximum total current for the applicable package, see [Electrical Characteristics](#) on page 337.

## Shared Reset Pin

On all the devices, the Port D0 pin shares function with a bidirectional reset pin. Unlike all other I/O pins, this pin does not default to GPIO pin on power-up. This pin acts as a bidirectional reset until user software reconfigures it. The Port D0 pin is output-only when in GPIO mode.

## Crystal Oscillator Override

For systems using the crystal oscillator, PA0 and PA1 is used to connect the crystal. When the main crystal oscillator is enabled (see [Oscillator Control0 Register](#) on page 306), the GPIO settings are overridden and PA0 and PA1 is disabled.

## 32 kHz Secondary Oscillator Override

For systems using a 32 kHz secondary oscillator, PA2 and PA3 is used to connect a watch crystal. When the 32 kHz secondary oscillator is enabled (see [Oscillator Control1 Register](#) on page 307), the GPIO settings are overridden and PA2 and PA3 is disabled.

## 5 V Tolerance

All GPIO pins, including those that share functionality with an ADC, crystal or comparator signals are 5 V tolerant and can handle inputs higher than  $V_{DD}$  even with the pull-ups enabled.

## External Clock Setup

For systems using an external TTL drive, PB3 is the clock source for 20-pin, 28-pin, 40-pin, and 44-pin devices. In this case, configure PB3 for alternate function CLKIN. Write the Oscillator Control Register (see page 306) such that the external oscillator is selected as the system clock.

**Table 15. Port Alternate Function Mapping (20-Pin Parts)**

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port A	PA0	T0IN/T0OUT	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0
		Reserved		AFS1[1]: 1
	PA2	DE0	UART 0 Driver Enable	AFS1[2]: 0
		Reserved		AFS1[2]: 1
	PA3	CTS0	UART 0 Clear to Send	AFS1[3]: 0
		Reserved		AFS1[3]: 1
	PA4	RXD0/IRRX0	UART 0/IrDA 0 Receive Data	AFS1[4]: 0
		T2IN/T2OUT	Timer 2 Input/Timer 2 Output Complement	AFS1[4]: 1
	PA5	TXD0/IRTX0	UART 0/IrDA 0 Transmit Data	AFS1[5]: 0
		T2OUT	Timer 2 Output	AFS1[5]: 1
	PA6	T1IN/T1OUT	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0
		SCL	I <sup>2</sup> C Serial Clock	AFS1[6]: 1
	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0
		SDA	I <sup>2</sup> C Serial Data	AFS1[7]: 1

Note: Because there are at most two choices of alternate function for some pins of Port A, the Alternate Function Set register AFS2 is implemented but not used to select the function. Also, alternate function selection as described in [Port A–E Alternate Function Subregisters](#) on page 64 must also be enabled.

**Table 15. Port Alternate Function Mapping (20-Pin Parts) (Continued)**

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
<b>Port B</b>	PB0	Reserved		AFS1[0]: 0
		ANA0/AMPOUT	ADC Analog Input/LPO Output	AFS1[0]: 1
	PB1	Reserved		AFS1[1]: 0
		ANA1/AMPINN	ADC Analog Input/LPO Input (N)	AFS1[1]: 1
	PB2	Reserved		AFS1[2]: 0
		ANA2/AMPINP	ADC Analog Input/LPO Input (P)	AFS1[2]: 1
	PB3	CLKIN	External Clock Input	AFS1[3]: 0
		ANA3	ADC Analog Input	AFS1[3]: 1
<b>Port C</b>	PC0	Reserved		AFS1[0]: 0
		ANA4/C0INP/LED	ADC or Comparator 0 Input (P), or LED drive	AFS1[0]: 1
	PC1	Reserved		AFS1[1]: 0
		ANA5/C0INN/LED	ADC or Comparator 0 Input (N), or LED drive	AFS1[1]: 1
	PC2	Reserved		AFS1[2]: 0
		VREF/ANA6/LED	Voltage Reference or ADC Analog Input or LED Drive	AFS1[2]: 1
	PC3	COUT0	Comparator 0 Output	AFS1[3]: 0
		LED	LED drive	AFS1[3]: 1
<b>Port D</b>	PD0	RESET	External Reset	N/A
		Reserved		

Note: Because there is only a single alternate function for each Port D pin, the Alternate Function Set registers are not implemented for Port D. Enabling alternate function selections as described in [Port A–E Alternate Function Subregisters](#) on page 64 automatically enables the associated alternate function.

**Table 16. Port Alternate Function Mapping (28-Pin Parts)**

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port A	PA0	T0IN/T0OUT	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0
		Reserved		AFS1[1]: 1
	PA2	DE0	UART 0 Driver Enable	AFS1[2]: 0
		Reserved		AFS1[2]: 1
	PA3	CTS0	UART 0 Clear to Send	AFS1[3]: 0
		Reserved		AFS1[3]: 1
	PA4	RXD0/IRRX0	UART 0/IrDA 0 Receive Data	AFS1[4]: 0
		Reserved		AFS1[4]: 1
	PA5	TXD0/IRTX0	UART 0/IrDA 0 Transmit Data	AFS1[5]: 0
		Reserved		AFS1[5]: 1
	PA6	T1IN/T1OUT	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0
		SCL	I <sup>2</sup> C Serial Clock	AFS1[6]: 1
	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0
		SDA	I <sup>2</sup> C Serial Data	AFS1[7]: 1

Note: Because there are at most two choices of alternate function for some pins of Port A, the Alternate Function Set register AFS2 is implemented but not used to select the function. Also, alternate function selection as described in [Port A–E Alternate Function Subregisters](#) on page 64 must also be enabled.

**Table 16. Port Alternate Function Mapping (28-Pin Parts) (Continued)**

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port B	PB0	Reserved		AFS1[0]: 0
		ANA0/AMPOUT	ADC Analog Input/LPO Output	AFS1[0]: 1
	PB1	Reserved		AFS1[1]: 0
		ANA1/AMPINN	ADC Analog Input/LPO Input (N)	AFS1[1]: 1
	PB2	Reserved		AFS1[2]: 0
		ANA2/AMPINP	ADC Analog Input/LPO Input (P)	AFS1[2]: 1
	PB3	CLKIN	External Clock Input	AFS1[3]: 0
		ANA3	ADC Analog Input	AFS1[3]: 1
	PB4	Reserved		AFS1[4]: 0
		ANA7	ADC Analog Input	AFS1[4]: 1
	PB5	Reserved		AFS1[5]: 0
		VREF	Voltage Reference	AFS1[5]: 1

Note: Because there are at most two choices of alternate function for some pins of Port B, the Alternate Function Set register AFS2 is implemented but not used to select the function. Also, alternate function selection as described in [Port A–E Alternate Function Subregisters](#) on page 64 must also be enabled.

**Table 16. Port Alternate Function Mapping (28-Pin Parts) (Continued)**

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
<b>Port C</b>	PC0	Reserved		AFS1[0]: 0
		ANA4/C0INP/ LED	ADC or Comparator 0 Input (P), or LED drive	AFS1[0]: 1
	PC1	MISO	SPI Master In/Slave Out	AFS1[1]: 0
		ANA5/C0INN/ LED	ADC or Comparator 0 Input (N), or LED Drive	AFS1[1]: 1
	PC2	$\overline{SS}$	SPI Slave Select	AFS1[2]: 0
		ANA6/LED	ADC Analog Input or LED Drive	AFS1[2]: 1
	PC3	COUT0	Comparator 0 Output	AFS1[3]: 0
		LED	LED drive	AFS1[3]: 1
	PC4	MOSI	SPI Master Out/Slave In	AFS1[4]: 0
		LED	LED Drive	AFS1[4]: 1
	PC5	SCK	SPI Serial Clock	AFS1[5]: 0
		LED	LED Drive	AFS1[5]: 1
	PC6	T2IN/ $\overline{T2OUT}$	Timer 2 Input/Timer 2 Output Complement	AFS1[6]: 0
		LED	LED Drive	AFS1[6]: 1
	PC7	T2OUT	Timer 2 Output	AFS1[7]: 0
		LED	LED Drive	AFS1[7]: 1
<b>Port D</b>	PD0	$\overline{RESET}$	External Reset	N/A
		Reserved		

Note: Because there is only a single alternate function for each Port D pin, the Alternate Function Set registers are not implemented for Port D. Enabling alternate function selections as described in [Port A–E Alternate Function Subregisters](#) on page 64 automatically enables the associated alternate function.

**Table 17. Port Alternate Function Mapping (40-pin/44-pin Parts)**

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port A	PA0	T0IN/T0OUT	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0
		Reserved		AFS1[1]: 1
	PA2	DE0	UART 0 Driver Enable	AFS1[2]: 0
		Reserved		AFS1[2]: 1
	PA3	CTS0	UART 0 Clear to Send	AFS1[3]: 0
		Reserved		AFS1[3]: 1
	PA4	RXD0/IRRX0	UART 0/IrDA 0 Receive Data	AFS1[4]: 0
				AFS1[4]: 1
	PA5	TXD0/IRTX0	UART 0/IrDA 0 Transmit Data	AFS1[5]: 0
				AFS1[5]: 1
	PA6	T1IN/T1OUT	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0
		Reserved		AFS1[6]: 1
	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0
		Reserved		AFS1[7]: 1

Note: Because there are at most two choices of alternate function for some pins of Port A, the Alternate Function Set register AFS2 is implemented but not used to select the function. Also, alternate function selection as described in [Port A–E Alternate Function Subregisters](#) on page 64 must also be enabled.

**Table 17. Port Alternate Function Mapping (40-pin/44-pin Parts) (Continued)**

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port B	PB0	Reserved		AFS1[0]: 0
		ANA0/AMPOUT	ADC Analog Input/LPO Output	AFS1[0]: 1
	PB1	Reserved		AFS1[1]: 0
		ANA1/AMPINN	ADC Analog Input/LPO Input (N)	AFS1[1]: 1
	PB2	Reserved		AFS1[2]: 0
		ANA2/AMPINP	ADC Analog Input/LPO Input (P)	AFS1[2]: 1
	PB3	CLKIN	External Clock Input	AFS1[3]: 0
		ANA3	ADC Analog Input	AFS1[3]: 1
	PB4	Reserved		AFS1[4]: 0
		ANA7	ADC Analog Input	AFS1[4]: 1
	PB5	Reserved		AFS1[5]: 0
		VREF	Voltage Reference	AFS1[5]: 1

Note: Because there are at most two choices of alternate function for some pins of Port B, the Alternate Function Set register AFS2 is implemented but not used to select the function. Also, alternate function selection as described in [Port A–E Alternate Function Subregisters](#) on page 64 must also be enabled.

Table 17. Port Alternate Function Mapping (40-pin/44-pin Parts) (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port C	PC0	Reserved		AFS1[0]: 0
		ANA4/C0INP/LED	ADC or Comparator 0 Input (P), or LED drive	AFS1[0]: 1
	PC1	Reserved		AFS1[1]: 0
		ANA5/C0INN/LED	ADC or Comparator 0 Input (N), or LED Drive	AFS1[1]: 1
	PC2	$\overline{SS}$	SPI Slave Select	AFS1[2]: 0
		ANA6/LED	ADC Analog Input or LED Drive	AFS1[2]: 1
	PC3	MISO	SPI Master In Slave Out	AFS1[3]: 0
		LED	LED drive	AFS1[3]: 1
	PC4	MOSI	SPI Master Out Slave In	AFS1[4]: 0
		LED	LED Drive	AFS1[4]: 1
	PC5	SCK	SPI Serial Clock	AFS1[5]: 0
		LED	LED Drive	AFS1[5]: 1
	PC6	T2IN/ $\overline{T2OUT}$	Timer 2 Input/Timer2 Output Complement	AFS1[6]: 0
		LED	LED Drive	AFS1[6]: 1
	PC7	T2OUT	Timer 2 Output	AFS1[7]: 0
		LED	LED Drive	AFS1[7]: 1

Note: Because there are at most two choices of alternate function for some pins of Port C, the Alternate Function Set register AFS2 is implemented but not used to select the function. Also, alternate function selection as described in [Port A–E Alternate Function Subregisters](#) on page 64 must also be enabled.

**Table 17. Port Alternate Function Mapping (40-pin/44-pin Parts) (Continued)**

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port D	PD0	RESET	External Reset	N/A
		Reserved		
	PD1	C1INN	Comparator 1 Input (N)	
		Reserved		
	PD2	C1INP	Comparator 1 Input (P)	
		Reserved		
	PD3	CTS1/COUT1	UART 1 Clear to Send or Comparator 1 Output	
		Reserved		
	PD4	RXD1/IRRX1	UART 1/IrDA 1 Receive Data	
		Reserved		
	PD5	TXD1/IRTX1	UART 1/IrDA 1 Transmit Data	
		Reserved		
	PD6	DE1		
		Reserved	UART 1 Driver Enable	
	PD7	COUT0	Comparator 0 Output	
		Reserved		

Note: Because there is only a single alternate function for each Port D pin, the Alternate Function Set registers are not implemented for Port D. Enabling alternate function selections as described in [Port A–E Alternate Function Subregisters](#) on page 64 automatically enables the associated alternate function.

Table 17. Port Alternate Function Mapping (40-pin/44-pin Parts) (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port E	PE0	T4IN*		N/A
		Reserved		
	PE1	SCL	I <sup>2</sup> C Serial Clock	
		Reserved		
	PE2	SDA	I <sup>2</sup> C Serial Data	
		Reserved		
	PE3	T4CHA*		
		Reserved		
	PE4	T4CHB*		
		Reserved		
	PE5	T4CHC*		
		Reserved		
	PE6	T4CHD*		
		Reserved		

Note: Because there is only a single alternate function for each Port E pin, the Alternate Function Set registers are not implemented for Port E. Enabling alternate function selections as described in [Port A–E Alternate Function Subregisters](#) on page 64 automatically enables the associated alternate function.

\* Timer Function only is available in 44-pin package. These alternative functions are reserved in 40-pin package.

## GPIO Interrupts

Many of the GPIO port pins can be used as interrupt sources. Some port pins can be configured to generate an interrupt request on either the rising edge or falling edge of the pin- input signal. Other port-pin interrupt sources generate an interrupt when any edge occurs (both rising and falling). For more details on interrupts using the GPIO pins, see [Interrupt Controller](#) on page 71.

## GPIO Control Register Definitions

Four registers for each port provide access to GPIO control, input data, and output data. [Table 18](#) lists these port registers. Use Port A–E Address and Control registers together to provide access to subregisters for port configuration and control.

**Table 18. GPIO Port Registers and Subregisters**

Port Register Mnemonic	Port Register Name
PxADDR	Port A–E Address Register (Selects subregisters)
PxCTL	Port A–E Control Register (Provides access to subregisters)
PxIN	Port A–E Input Data Register
PxOUT	Port A–E Output Data Register
Port Subregister Mnemonic	Port Register Name
PxDD	Data Direction
PxAF	Alternate Function
PxOC	Output Control (Open-Drain)
PxHDE	High Drive Enable
PxSMRE	Stop Mode Recovery Source Enable
PxPUE	Pull-up Enable
PxAFS1	Alternate Function Set 1
PxAFS2	Alternate Function Set 2

## Port A–E Address Registers

The Port A–E address registers select the GPIO Port functionality accessible through the Port A–E Control registers. The Port A–E Address and Control registers combine to provide access to all GPIO Port controls, see [Table 19](#).

**Table 19. Port A–E GPIO Address Registers (PxADDR)**

BITS	7	6	5	4	3	2	1	0
FIELD	PADDR[7:0]							
RESET	00H							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FD0H, FD4H, FD8H, FDCH, FE0H							

### PADDR[7:0]—Port Address

The Port Address selects one of the subregisters accessible through the Port Control register.

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

## Port A–E Control Registers

The Port A–E Control registers set the GPIO port operation. The value in the corresponding Port A–E Address register determines which subregister is read from or written to by a Port A–E Control register transaction, see [Table 20](#)

**Table 20. Port A–E Control Registers (PxCTL)**

BITS	7	6	5	4	3	2	1	0
FIELD	PCTL							
RESET	00H							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FD1H, FD5H, FD9H, FDDH, FE1H							

### PCTL[7:0]—Port Control

The Port Control register provides access to all subregisters that configure the GPIO Port operation.

## Port A–E Data Direction Subregisters

The Port A–E Data Direction subregister is accessed through the Port A–E Control register by writing 01H to the Port A–E Address register, see [Table 21](#).

**Table 21. Port A–E Data Direction Subregisters (PxDD)**

BITS	7	6	5	4	3	2	1	0
FIELD	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	If 01H in Port A–E Address Register, accessible through the Port A–E 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–E Output Data register is driven onto the port pin.

1 = Input. The port pin is sampled and the value written into the Port A–E Input Data Register. The output driver is tristated.

## Port A–E Alternate Function Subregisters

The Port A–E Alternate Function subregister (see [Table 22](#)) is accessed through the Port A–E Control register by writing 02H to the Port A–E Address register. The Port A–E Alternate Function subregisters enable the alternate function selection on pins. If disabled, pins functions as GPIO. If enabled, select one of the four alternate functions using Alternate Function set subregisters 1 and 2 as described in [Port A–E Alternate Function Set 1 Subregisters](#) on page 67 and [Port A–E Alternate Function Set 2 Subregisters](#) on page 67. To determine the alternate function associated with each port pin, see [GPIO Alternate Functions](#) on page 50 .



**Caution:** Do not enable alternate functions for GPIO port pins for which there is no associated alternate function. Failure to follow this guideline results in unpredictable operation.

**Table 22. Port A–E Alternate Function Subregisters (PxAF)**

BITS	7	6	5	4	3	2	1	0
FIELD	AF7	AF6	AF5	AF4	AF3	AF2	AF1	AF0
RESET	00H (Ports A–C); 01H (Port D); 00H (Port E);							
R/W	R/W							
ADDR	If 02H in Port A–D Address Register, accessible through the Port A–E 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–E Data Direction subregister determines the direction of the pin.

1 = The alternate function selected through Alternate Function set subregisters are enabled. Port-pin operation is controlled by the alternate function.

## Port A–E Output Control Subregisters

The Port A–E Output Control subregister (Table 23) is accessed through the Port A–E Control register by writing 03H to the Port A–E Address register. Setting the bits in the Port A–E Output Control subregisters to 1 configures the specified port pins for open-drain operation. These subregisters affect the pins directly and, as a result, alternate functions are also affected.

**Table 23. Port A–E Output Control Subregisters (PxOC)**

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

The Port A–E High Drive Enable subregister (Table 24) is accessed through the Port A–E Control register by writing 04H to the Port A–E Address register. Setting the bits in the Port A–E High Drive Enable subregisters to 1 configures the specified port pins for high-current output drive operation. The Port A–E High Drive Enable subregister affects the pins directly and, as a result, alternate functions are also affected.

**Table 24. Port A–E High Drive Enable Subregisters (PxHDE)**

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

The Port A–E Stop Mode Recovery Source Enable subregister (Table 25) is accessed through Port A–E Control register by writing 05H to the Port A–E Address register. Setting the bits in the Port A–E Stop Mode Recovery Source Enable subregisters to 1 configures the specified Port pins as a Stop Mode Recovery source. During STOP mode, any logic transition on a Port pin enabled as a Stop Mode Recovery source initiates Stop Mode Recovery.

**Table 25. Port A–E Stop Mode Recovery Source Enable Subregisters (PxSMRE)**

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

The Port A–E Pull-up Enable subregister (Table 26) is accessed through the Port A–E Control register by writing 06H to the Port A–E Address register. Setting the bits in the Port A–E Pull-up Enable subregisters enables a weak internal resistive pull-up on the specified Port pins.

**Table 26. Port A–E Pull-Up Enable Subregisters (PxPUE)**

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

The Port A–E Alternate Function Set1 subregister ([Table 27](#)) is accessed through the Port A–E Control register by writing 07H to the Port A–E Address register. The Alternate Function Set 1 subregisters select the alternate function available at a port pin.

Alternate Functions selected by setting or clearing bits of this register are defined in [GPIO Alternate Functions](#) on page 50.

► **Note:** *Alternate function selection on port pins must also be enabled as described in [Port A–E Alternate Function Subregisters](#) on page 64.*

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

BITS	7	6	5	4	3	2	1	0
FIELD	PAFS17	PAFS16	PAFS15	PAFS14	PAFS13	PAFS12	PAFS11	PAFS10
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	If 07H in Port A–E Address Register, accessible through the Port A–E Control Register							

**PAFS1[7:0]—Port Alternate Function Set 1**

0 = Port Alternate Function selected as defined in [Table 15](#) through [Table 17](#) in section [GPIO Alternate Functions](#) on page 50.

1 = Port Alternate Function selected as defined in [Table 15](#) through [Table 17](#) in section [GPIO Alternate Functions](#) on page 50.

## Port A–E Alternate Function Set 2 Subregisters

The Port A–E Alternate Function Set 2 subregister (see [Table 28](#) on page 68) is accessed through the Port A–E Control register by writing 08H to the Port A–E Address register.

The Alternate Function Set 2 subregisters selects the alternate function available at a port

pin. Alternate Functions selected by setting or clearing bits of this register is defined in [Table 15](#) through [Table 17](#) in section [GPIO Alternate Functions](#) on page 50.

► **Note:** *Alternate function selection on port pins must also be enabled as described in [Port A–E Alternate Function Subregisters](#) on page 64.*

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

BITS	7	6	5	4	3	2	1	0
FIELD	PAFS27	PAFS26	PAFS25	PAFS24	PAFS23	PAFS22	PAFS21	PAFS20
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	If 08H in Port A–E Address Register, accessible through the Port A–E Control Register							

**PAFS2[7:0]—Port Alternate Function Set 2**

0 = Port Alternate Function selected as defined in [Table 15](#) through [Table 17](#) in section [GPIO Alternate Functions](#) on page 50.

1 = Port Alternate Function selected as defined in [Table 15](#) through [Table 17](#) in section [GPIO Alternate Functions](#) on page 50.

## Port A–E Input Data Registers

Reading from the Port A–E Input Data registers ([Table 29](#)) returns the sampled values from the corresponding port pins. The Port A–E Input Data registers are read-only. The value returned for any unused ports is 0. Unused ports include those missing on the 8-pin and 28-pin packages as well as those missing on the ADC-enabled 28-pin packages.

**Table 29. Port A–E Input Data Registers (PxIN)**

BITS	7	6	5	4	3	2	1	0
FIELD	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
ADDR	FD2H, FD6H, FDAH, FDEH, FE2H							

**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–E Output Data Register

The Port A–E Output Data register (Table 30) controls the output data to the pins.

Table 30. Port A–E Output Data Register (PxOUT)

BITS	7	6	5	4	3	2	1	0
FIELD	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FD3H, FD7H, FDBH, FDFH, FE3H							

### 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.

## LED Drive Enable Register

The LED Drive Enable register (Table 31) activates the controlled current drive. The Port C pin must first be enabled by setting the Alternate Function register to select the LED function.

Table 31. LED Drive Enable (LEDEN)

BITS	7	6	5	4	3	2	1	0
FIELD	LEDEN[7:0]							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F82H							

### LEDEN[7:0]—LED Drive Enable

These bits determine which Port C pins are connected to an internal current sink.

0 = Tristate the Port C pin.

1 = Connect controlled current synch to Port C pin.

## LED Drive Level Registers

The LED Drive Level Registers consist of two registers: LED Drive Level High Bit Register—LEDLVLH[7:0] and LED Drive Level Low Bit Register—LEDLVLL[7:0] (see

Table 32 and Table 33 on page 70). Two control bits, LEDVLH[x] and LEDVLL[x], are used to select one of four programmable current drive levels for each associated Port C[x] pin. Each Port C pin is individually programmable.

**Table 32. LED Drive Level High Bit Register (LEDVLH)**

BITS	7	6	5	4	3	2	1	0
FIELD	LEDVLH[7:0]							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F83H							

**Table 33. LED Drive Level Low Bit Register (LEDVLL)**

BITS	7	6	5	4	3	2	1	0
FIELD	LEDVLL[7:0]							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F84H							

LEDVLH[7:0]—LED Drive Level High Bit and LEDVLL[7:0]—LED Drive Level Low Bit are used to set the LED drive current.

{LEDVLH[x], LEDVLL[x]}, where x=Port C[0] to Port C[7], select one of below four programmable current drive levels for each Port C pin.

00 = 3 mA

01 = 7 mA

10 = 13 mA

11 = 20 mA

# Interrupt Controller

## Overview

The interrupt controller on the Z8 Encore! XP F1680 Series products prioritizes the interrupt requests from the on-chip peripherals and the GPIO port pins. The interrupt controller includes the following features:

- Thirty-one interrupt sources using twenty-four unique interrupt vectors
  - 16 GPIO port pin interrupt sources (seven interrupt vectors are shared, see [Table 34](#) on page 72)
  - 15 on-chip peripheral interrupt sources (three interrupt vectors are shared, see [Table 34](#) on page 72)
- Flexible GPIO interrupts
  - Twelve selectable rising and falling edge GPIO interrupts
  - Four dual-edge interrupts
- Three levels of individually programmable interrupt priority
- WDT can be 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 interrupt service routine 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 controller has no effect on operation. For more information on interrupt servicing by the eZ8 CPU, refer to *eZ8 CPU User Manual (UM0077)*. The *eZ8 CPU User Manual* is available on [www.zilog.com](http://www.zilog.com).

## Interrupt Vector Listing

[Table 34](#) on page 72 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.

► **Note:** *Some port interrupts are not available on the 20-pin and 28-pin packages. The ADC interrupt is unavailable on devices not containing an ADC.*

**Table 34. Trap and Interrupt Vectors in Order of Priority**

Priority *	Program Memory Vector Address	Interrupt or Trap Source
Highest	0002H	Reset (not an interrupt)
	0004H	Watchdog Timer (see <a href="#">Watchdog Timer</a> on page 137)
	003AH	Primary Oscillator Fail Trap (not an interrupt)
	003CH	Watchdog Timer Oscillator Fail Trap (not an interrupt)
	0006H	Illegal Instruction Trap (not an interrupt)
	0008H	Timer 2
	000AH	Timer 1
	000CH	Timer 0
	000EH	UART 0 receiver
	0010H	UART 0 transmitter
	0012H	I <sup>2</sup> C
	0014H	SPI
	0016H	ADC
	0018H	Port A7, selectable rising or falling input edge or LVD (see <a href="#">Reset, Stop Mode Recovery, and Low-Voltage Detection</a> on page 33)
	001AH	Port A6, selectable rising or falling input edge or Comparator 0 Output
	001CH	Port A5, selectable rising or falling input edge or Comparator 1 Output
	001EH	Port A4 or Port D4, selectable rising or falling input edge
	0020H	Port A3 or Port D3, selectable rising or falling input edge
	0022H	Port A2 or Port D2, selectable rising or falling input edge
	0024H	Port A1 or Port D1, selectable rising or falling input edge
	0026H	Port A0, selectable rising or falling input edge
	0028H	Reserved
	002AH	Multi-channel Timer
	002CH	UART 1 receiver
	002EH	UART 1 transmitter
	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
	0038H	Reserved

Note: \* = The order of priority is only for identical interrupt level. The priority varies depending on different interrupt level setting. See [Interrupt Vectors and Priority](#) on page 74 for details.

## Architecture

Figure 10 illustrates the interrupt controller block diagram.

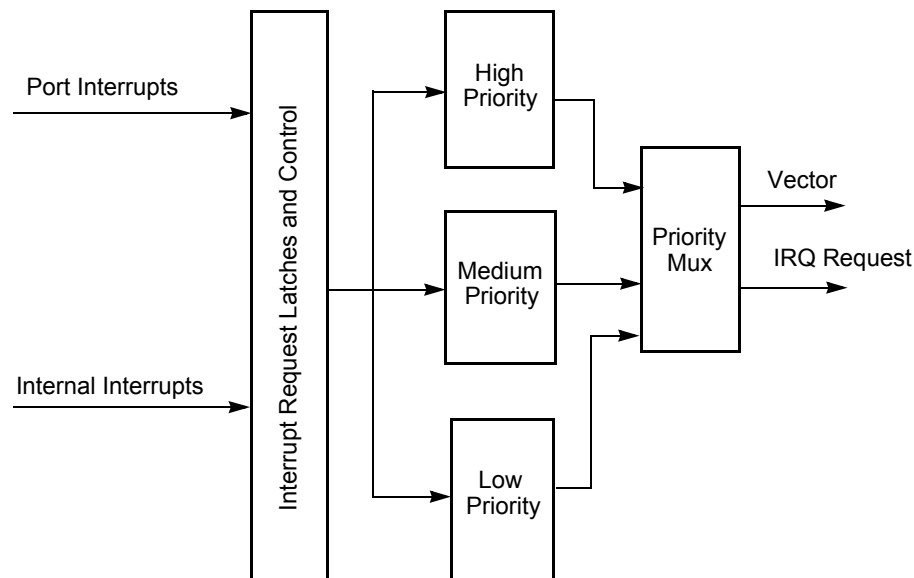


Figure 10. Interrupt Controller Block Diagram

## 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 Enable Interrupt (EI) instruction
- Execution of an Interrupt Return (IRET) instruction
- Writing 1 to the IRQE bit in the interrupt control register

Interrupts are globally disabled by any of the following actions:

- Execution of a Disable Interrupt (DI) instruction
- eZ8 CPU acknowledgement of an interrupt service request from the interrupt controller
- Writing 0 to the IRQE bit in the interrupt control register

- Reset
- Execution of a Trap instruction
- Illegal Instruction Trap
- Primary Oscillator Fail Trap
- Watchdog Oscillator Fail Trap

## Interrupt Vectors and Priority

The interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority, and Level 1 is the lowest priority. If all the interrupts are enabled with identical interrupt priority (for example, all as Level 2 interrupts), the interrupt priority is assigned from highest to lowest as specified in [Table 34](#) on page 72. Level 3 interrupts are always assigned higher priority than Level 2 interrupts which, in turn, always are assigned higher priority than Level 1 interrupts. Within each interrupt priority level (Level 1, Level 2, or Level 3), priority is assigned as specified in [Table 34](#) on page 72. Reset, Watchdog Timer interrupt (if enabled), Primary Oscillator Fail Trap, Watchdog Timer Oscillator Fail Trap, and Illegal Instruction Trap always have highest (Level 3) priority.

## Interrupt Assertion

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



**Caution:** *The following coding style that clears bits in the Interrupt Request registers is not recommended. All incoming interrupts received between execution of the first LDX command and the final LDX command are lost.*

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

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



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

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

```
ANDX IRQ0, MASK
```

## Software Interrupt Assertion

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



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

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

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



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

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

```
ORX IRQ0, MASK
```

## Interrupt Control Register Definitions

For all interrupts other than the Watchdog Timer interrupt, the Primary Oscillator Fail Trap, and the Watchdog Oscillator Fail Trap, the Interrupt Control registers enable individual interrupts, set interrupt priorities, and indicate interrupt requests.

### Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) register (see [Table 35](#) on page 75) 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 can read the Interrupt Request 0 register to determine if any interrupt requests are pending.

**Table 35. Interrupt Request 0 Register (IRQ0)**

BITS	7	6	5	4	3	2	1	0
FIELD	T2I	T1I	T0I	U0RXI	U0TXI	I <sup>2</sup> CI	SPII	ADCI
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FC0H							

#### **T2I—Timer 2 Interrupt Request**

0 = No interrupt request is pending for Timer 2.  
1 = An interrupt request from Timer 2 is awaiting service.

#### **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.

#### **I<sup>2</sup>CI—I<sup>2</sup>C Interrupt Request**

0 = No interrupt request is pending for the I<sup>2</sup>C.  
1 = An interrupt request from I<sup>2</sup>C 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 36](#)) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ1 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 1 register to determine if any interrupt requests are pending.

**Table 36. Interrupt Request 1 Register (IRQ1)**

<b>BITS</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>FIELD</b>	PA7VI	PA6CI	PA5CI	PAD4I	PAD3I	PAD2I	PAD1I	PA0I
<b>RESET</b>	0	0	0	0	0	0	0	0
<b>R/W</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>ADDR</b>	FC3H							

**PA7VI—Port A7 or LVD Interrupt Request**

0 = No interrupt request is pending for GPIO Port A7 or LVD.

1 = An interrupt request from GPIO Port A7 or LVD.

**PA6CI—Port A6 or Comparator 0 Interrupt Request**

0 = No interrupt request is pending for GPIO Port A6 or Comparator 0.

1 = An interrupt request from GPIO Port A6 or Comparator 0.

**PA5CI—Port A5 or Comparator 1 Interrupt Request**

0 = No interrupt request is pending for GPIO Port A5 or Comparator 1.

1 = An interrupt request from GPIO Port A5 or Comparator 1.

**PADxI—Port A or Port D Pin x Interrupt Request**

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

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

Where x indicates the specific GPIO Port pin number (1–4).

**PA0I—Port A Pin 0 Interrupt Request**

0 = No interrupt request is pending for GPIO Port A0.

1 = An interrupt request from GPIO Port A0 is awaiting service.

For interrupt source select description, see [Shared Interrupt Select Register](#) on page 82.

## Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) register ([Table 37](#)) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ2 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 2 register to determine if any interrupt requests are pending.

**Table 37. Interrupt Request 2 Register (IRQ2)**

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	MCTI	U1RXI	U1TXI	PC3I	PC2I	PC1I	PC0I
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FC6H							

**Reserved—Must be 0**

**MCTI—Multi-channel timer Interrupt Request**

0 = No interrupt request is pending for multi-channel timer.

1 = An interrupt request from multi-channel timer is awaiting service.

**U1RXI—UART 1 Receiver Interrupt Request**

0 = No interrupt request is pending for the UART 1 receiver.

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

#### UITXI—UART 1 Transmitter Interrupt Request

0 = No interrupt request is pending for the UART 1 transmitter.

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

#### 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 (0–3).

### IRQ0 Enable High and Low Bit Registers

Table 38 describes the priority control for IRQ0. The IRQ0 Enable High and Low Bit registers (see Table 39 and Table 40 on page 79) form a priority encoded enabling for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register.

**Table 38. IRQ0 Enable and Priority Encoding**

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

**Note:**  $x$  indicates the register bits from 0–7.

**Table 39. IRQ0 Enable High Bit Register (IRQ0ENH)**

BITS	7	6	5	4	3	2	1	0
FIELD	T2ENH	T1ENH	T0ENH	U0RENH	U0TENH	I <sup>2</sup> CENH	SPIENH	ADCENH
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FC1H							

T2ENH—Timer 2 Interrupt Request Enable High Bit.

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.

I<sup>2</sup>CENH—I<sup>2</sup>C Interrupt Request Enable High Bit.

SPIENH—SPI Interrupt Request Enable High Bit.

ADCENH—ADC Interrupt Request Enable High Bit.

**Table 40. IRQ0 Enable Low Bit Register (IRQ0ENL)**

BITS	7	6	5	4	3	2	1	0
FIELD	T2ENL	T1ENL	T0ENL	U0RENL	U0TENL	I2CENL	SPIENL	ADCENL
RESET	0	0	0	0	0	0	0	0
R/W	R	R/W	R/W	R/W	R/W	R	R	R/W
ADDR	FC2H							

T2ENL—Timer 2 Interrupt Request Enable Low Bit.

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.

I<sup>2</sup>CENL—I<sup>2</sup>C 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 41 on page 79 lists the priority control for IRQ1. The IRQ1 enable High and Low Bit registers (see Table 42 and Table 43 on page 80) form a priority encoded enabling for interrupts in the Interrupt Request 1 register. Priority is generated by setting bits in each register.

**Table 41. IRQ1 Enable and Priority Encoding**

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

**Note:** x indicates the register bits from 0–7.

**Table 42. IRQ1 Enable High Bit Register (IRQ1ENH)**

BITS	7	6	5	4	3	2	1	0
FIELD	PA7VENH	PA6C0ENH	PA5C1ENH	PAD4ENH	PAD3ENH	PAD2ENH	PAD1ENH	PA0ENH
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FC4H							

PA7VENH—Port A Bit[7] or LVD Interrupt Request Enable High Bit.  
 PA6C0ENH—Port A Bit[6] or Comparator 0 Interrupt Request Enable High Bit.  
 PA5C1ENH—Port A Bit[5] or Comparator 1 Interrupt Request Enable High Bit.  
 PADxENH—Port A or Port D Bit[x] (x=1, 2, 3, 4) Interrupt Request Enable High Bit.  
 PA0ENH—Port A Bit[0] Interrupt Request Enable High Bit.  
 See Interrupt Port Select register for selection of either Port A or Port D as the interrupt source.

**Table 43. IRQ1 Enable Low Bit Register (IRQ1ENL)**

BITS	7	6	5	4	3	2	1	0
FIELD	PA7VENL	PA6C0ENL	PA5C1ENL	PAD4ENL	PAD3ENL	PAD2ENL	PAD1ENL	PA0ENL
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FC5H							

PA7VENL—Port A Bit[7] or LVD Interrupt Request Enable Low Bit.  
 PA6C0ENL—Port A Bit[6] or Comparator 0 Interrupt Request Enable Low Bit.  
 PA5C1ENL—Port A Bit[5] or Comparator 1 Interrupt Request Enable Low Bit.  
 PADxENL—Port A or Port D Bit[x] (x=1, 2, 3, 4) Interrupt Request Enable Low Bit.  
 PA0ENL—Port A Bit[0] Interrupt Request Enable Low Bit.

## IRQ2 Enable High and Low Bit Registers

Table 44 describes the priority control for IRQ2. The IRQ2 enable High and Low Bit registers (see Table 45 and Table 46 on page 81) form a priority encoded enabling for interrupts in the Interrupt Request 2 register. Priority is generated by setting bits in each register.

**Table 44. IRQ2 Enable and Priority Encoding**

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

**Note:** x indicates the register bits from 0–7.

Table 45. IRQ2 Enable High Bit Register (IRQ2ENH)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	MCTENH	U1RENH	U1TENH	C3ENH	C2ENH	C1ENH	C0ENH
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FC7H							

**Reserved—Must be 0.**

MCTENH—Multi-Channel Timer Interrupt Request Enable High Bit.

U1RENH—UART1 Receive Interrupt Request Enable High Bit.

U1TENH—UART1 Transmit Interrupt Request Enable High Bit.

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 46. IRQ2 Enable Low Bit Register (IRQ2ENL)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	MCTENL	U1RENL	U1TENL	C3ENL	C2ENL	C1ENL	C0ENL
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FC8H							

**Reserved—Must be 0.**

MCTENL—Multi-Channel Timer Interrupt Request Enable Low Bit.

U1RENL—UART1 Receive Interrupt Request Enable Low Bit.

U1TENL—UART1 Transmit Interrupt Request Enable Low Bit.

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 47) determines whether an interrupt is generated for the rising edge or falling edge on the selected GPIO Port A or Port D input pin.

**Table 47. 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	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FCDH							

### IESx—Interrupt Edge Select x

0 = An interrupt request is generated on the falling edge of the PAx input or PDx.

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

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

## Shared Interrupt Select Register

The Shared Interrupt Select (IRQSS) register (Table 48) determines the source of the PADxS interrupts. The Shared Interrupt Select register selects between Port A and alternate sources for the individual interrupts.

**Table 48. Shared Interrupt Select Register (IRQSS)**

BITS	7	6	5	4	3	2	1	0
FIELD	PA7VS	PA6CS	PA5CS	PAD4S	PAD3S	PAD2S	PAD1S	Reserved
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FCEH							

### PA7VS—PA7/LVD Selection

0 = PA7 is used for the interrupt for PA7VS interrupt request.

1 = The LVD is used for the interrupt for PA7VS interrupt request.

### PA6CS—PA6/Comparator 0 Selection

0 = PA6 is used for the interrupt for PA6CS interrupt request.

1 = The Comparator 0 is used for the interrupt for PA6CS interrupt request.

**PA5CS—PA5/Comparator 1 Selection**

0 = PA5 is used for the interrupt for PA5CS interrupt request.

1 = The Comparator 1 is used for the interrupt for PA5CS interrupt request.

**PADxS—PAx/PDx Selection**

0 = PAx is used for the interrupt for PAx/PDx interrupt request

1 = PDx is used for the interrupt for PAx/PDx interrupt request

Where *x* indicates the specific GPIO Port pin number (1–4).

**Reserved—Must be 0**

## Interrupt Control Register

The Interrupt Control (IRQCTL) register ([Table 49](#)) contains the master enable bit for all interrupts.

**Table 49. Interrupt Control Register (IRQCTL)**

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

**IRQE—Interrupt Request Enable**

This bit is set to 1 by executing an Enable Interrupts (EI) 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

## Overview

The Z8 Encore! XP F1680 Series products contain three 16-bit reloadable timers that can be used for timing, event counting, or generation of pulse-width modulated signals. The timers' features include:

- 16-bit reload counter.
- Programmable prescaler with prescale values ranging from 1 to 128.
- PWM output generation.
- Capture and compare capability.
- Two independent capture/compare channels which reference the common timer.
- 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 timer clock frequency.
- Timer output pin.
- Timer interrupt.
- Noise Filter on Timer input signal.
- Operation in any mode with 32 kHz secondary oscillator.

In addition to the timers described in this chapter, the Baud Rate Generator (BRG) of unused UART peripheral may also be used to provide basic timing functionality. For more information on using the Baud Rate Generator as additional timers, see [LIN-UART](#) on page 141.

## Architecture

[Figure 11](#) on page 86 illustrates the architecture of the timers.

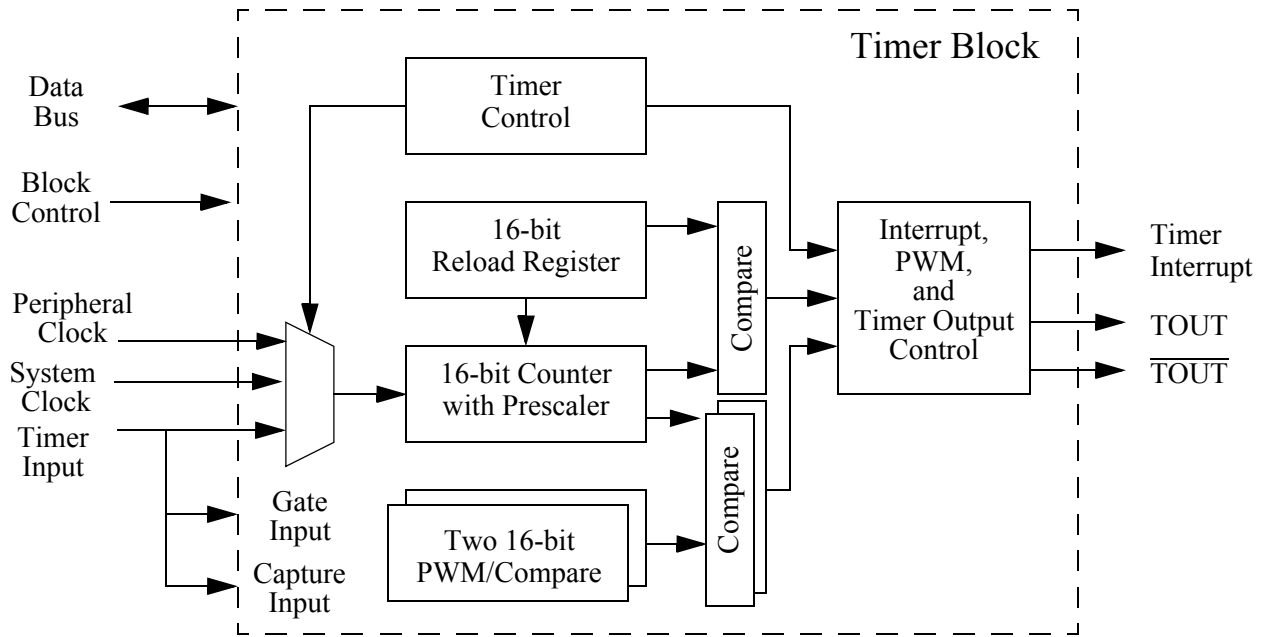


Figure 11. Timer Block Diagram

## Operation

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

### Timer Clock Source

The timer clock source can come from either the peripheral clock or the system clock. Peripheral clock is based on a low frequency/low power 32 kHz secondary oscillator that can be used with external watch crystal. Peripheral clock source is only available for driving Timer and Noise Filter operation. It is not supported for other peripherals.

For timer operation in STOP Mode, peripheral clock must be selected as the clock source. Peripheral clock can be selected as source for both ACTIVE and STOP mode operation. System clock is only for operation in ACTIVE and HALT modes. System clock is software selectable in Oscillator Control Module as external high frequency crystal or internal precision oscillator. The TCLKS field in the Timer Control 2 register selects the timer clock source.

**Caution:**

*When timer is operating on a peripheral clock, the timer clock is asynchronous to the CPU clock. To ensure error-free operation, disable the timer before modifying its operation (include changing the timer clock source). So any write to the timer control registers can not be done when timer is enabled and peripheral clock is used.*

When Timer uses peripheral clock and Timer is enabled, any read from TxH or TxL is not recommended, results may be unpredictable, disable Timer first, then read it. If timer work in capture, capture/compare, capture restart or demodulation mode, any read from TxPWM0H, TxPWM0L, TxPWM1H, TxPWM1L, or TxSTAT must be done after capture interrupt occurs, or results may be unpredictable. INPCAP bit of Timer Control 0 register is the same as these PWM registers. When Timer uses main clock, you can write/read all Timer registers at any time.

## Low-Power Modes

### Operation in HALT Mode

When the eZ8 CPU enters HALT mode, the timer will continue to operate if enabled. To minimize current in HALT mode, the timer can be disabled by clearing the TEN control bit. The noise filter, if enabled, will also continue to operate in HALT mode and rejects any noise on the timer input pin.

### Operation in STOP Mode

When the eZ8 CPU enters STOP Mode, the timer continues to operate if enabled and peripheral clock is chosen as the clock source. In STOP Mode, the timer interrupt (if enabled) automatically initiates a Stop Mode Recovery and generates an interrupt request. In the Reset Status Register, the stop bit is set to 1. Also, timer interrupt request bit in Interrupt Request 0 register is set. Following completion of the Stop Mode Recovery, if interrupts are enabled, the CPU responds to the interrupt request by fetching the timer interrupt vector. The noise filter, if enabled, will also continue to operate in STOP Mode and rejects any noise on the timer input pin.

If system clock is chosen as the clock source, the timer ceases to operate as a system clock and is put into STOP Mode. In this case the registers are not reset and operation will resume once Stop Mode Recovery occurs.

### Power Reduction During Operation

Removal of the TEN bit will inhibit clocking of the entire timer block. The CPU can still read/write registers when the enable bit(s) are taken out.

## Timer Operating Modes

The timers can be 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 counts timer clocks up to the 16-bit Reload value. 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 clock cycle (from Low to High or from High to Low) on timer Reload. If it is desired to have the Timer Output make a permanent state change on One-Shot timeout. First set the TPOL bit in the Timer Control 1 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 1 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 Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value.
5. Write to the Timer Reload High and Low Byte registers to set the Reload value.
6. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
8. Write to the Timer Control 1 register to enable the timer and initiate counting.

In ONE-SHOT mode, the timer clock always provides the timer input. The timer period is given by the following equation:

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

### Triggered ONE-SHOT Mode

In Triggered ONE-SHOT mode, the Timer operates as follows:

1. The Timer idles until a trigger is received. The Timer trigger is taken from the GPIO Port pin Timer Input alternate function. The TPOL bit in the Timer Control 1 register selects whether the trigger occurs on the rising edge or the falling edge of the Timer Input signal.
2. Following the trigger event, the Timer counts timer clocks up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers.
3. On reaching the Reload value, the Timer outputs a pulse on the Timer Output pin, generates an interrupt, and resets the count value in the Timer High and Low Byte registers to 0001H. The period of the output pulse is a single timer clock. The TPOL bit also sets the polarity of the output pulse.
4. The Timer now idles until the next trigger event.

In Triggered ONE-SHOT mode, the timer clock always provides the timer input. The timer period is given by the following equation:

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

Table 50 provides an example initialization sequence for configuring Timer 0 in Triggered ONE-SHOT mode and initiating operation.

**Table 50. Triggered ONE-SHOT Mode Initialization Example**

Register	Value	Comment
T0CTL0	E0H	TMODE[3:0] = 1011B selects Triggered ONE-SHOT mode.
T0CTL1	03H	TICONFIG[1:0] = 11B enables interrupts on Timer reload only.
T0CTL2	01H	CSC = 0 selects the Timer Input (Trigger) from the GPIO pin. PWMD[2:0] = 000B has no effect. INPCAP = 0 has no effect. TEN = 0 disables the timer. TPOL = 0 enables triggering on rising edge of Timer. Input and sets Timer Out signal to 0. PRES[2:0] = 000B sets prescaler to divide by 1. TCLKS = 1 sets 32 kHz peripheral clock as the Timer clock source.
T0H	00H	Timer starting value = 0001H.
T0L	01H	
T0RH	ABH	Timer reload value = ABCDH.
T0RL	CDH	

**Table 50. Triggered ONE-SHOT Mode Initialization Example (Continued)**

Register	Value	Comment
PAADDR	02H	Selects Port A Alternate Function control register.
PACTL[1:0]	11B	PACTL[0] enables Timer 0 Input Alternate function. PACTL[1] enables Timer 0 Output Alternate function.
IRQ0ENH[5]	0B	Disables the Timer 0 interrupt.
IRQ0ENL[5]	0B	
TOCTL1	83H	TEN = 1 enables the timer. All other bits left in desired settings.

Note: After receiving the input trigger, Timer 0 will

1. Count ABCDH timer clocks.
2. Upon Timer 0 reload, generate single clock cycle active High output pulse on Timer 0 Output pin.
3. Wait for next input trigger event.

### 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 counts timer clocks up to the 16-bit Reload value. On 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 High to Low) on timer Reload.

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

1. Write to the Timer Control 1 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 Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt-configuration field TICONFIG.
4. 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.
5. Write to the Timer Reload High and Low Byte registers to set the Reload value.

6. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
8. Write to the Timer Control 1 register to enable the timer and initiate counting.

In CONTINUOUS mode, the timer clock always provides the timer input. The timer period is given by the following equation:

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

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 timeout 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 1 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 the timer clock frequency.*

On 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 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 1 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 need not be enabled.
2. Write to the Timer Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.

4. 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.
5. Write to the Timer Reload High and Low Byte registers to set the Reload value.
6. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. Configure the associated GPIO port pin for the Timer Input alternate function.
8. When using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
9. Write to the Timer Control 1 register to enable the timer.

In COUNTER mode, the number of Timer Input transitions since the timer start is given by the following equation:

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

### COMPARATOR COUNTER Mode

In COMPARATOR COUNTER mode, the timer counts output transitions from an analog comparator output. Timer 0 takes its input from the output of Comparator 0. Timer 1 takes its input from the output of Comparator 1. The TPOL bit in the Timer Control 1 register selects whether the count occurs on the rising edge or the falling edge of the comparator output signal. In COMPARATOR COUNTER mode, the prescaler is disabled.



**Caution:** *The frequency of the comparator output signal must not exceed one-fourth the timer clock frequency.*

On 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 High to Low) at timer Reload.

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

1. Write to the Timer Control 1 register to:
  - Disable the timer.
  - Configure the timer for COMPARATOR COUNTER mode.
  - Select either the rising edge or falling edge of the comparator output signal for the count. This also sets the initial logic level (High or Low) for the Timer Output alternate function. The Timer Output function does not have to be enabled.

2. Write to the Timer Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value. This only affects the first pass in COMPARATOR COUNTER mode. After the first timer Reload in COMPARATOR COUNTER mode, counting always begins at the reset value of 0001H. Generally, in COMPARATOR COUNTER mode the Timer High and Low Byte registers must be written with the value 0001H.
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
8. Write to the Timer Control 1 register to enable the timer.

In COMPARATOR COUNTER mode, the number of comparator output transitions since the timer start is given by the following equation:

$$\text{Comparator Output Transitions} = \text{Current Count Value} - \text{Start Value}$$

### **PWM Single Output Mode**

In PWM Single Output mode, the timer outputs a Pulse Width Modulator output signal through a GPIO Port pin. The Timer counts timer clocks up to the 16-bit Reload value. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM0 High and Low Byte registers. When the timer count value matches the PWM value, the Timer Output toggles. The timer continues counting till it reaches the Reload value stored in the Timer Reload High and Low Byte registers. On 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 1 register is set to 1, the Timer Output signal begins as High (1) and then transitions to Low (0) when the timer value matches the PWM value. The Timer Output signal returns to High (1) after the timer reaches the Reload value and is reset to 0001H.

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

Follow the steps below for configuring a timer for PWM Single Output mode and initiating the PWM operation:

1. Write to the Timer Control 1 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 Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
4. 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.
5. Write to the Timer PWM0 High and Low Byte registers to set the PWM value.
6. 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.
7. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
8. Configure the associated GPIO port pin for the Timer Output alternate function.
9. Write to the Timer Control 1 register to enable the timer and initiate counting.

The PWM period is given by the following equation:

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

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 PWM timeout 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:

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

## PWM Dual Output Mode

In PWM Dual Output mode, the timer outputs a Pulse Width Modulator output signal and also its complement through two GPIO Port pins. The Timer counts timer clocks up to the 16-bit Reload value. The timer first counts up to 16-bit PWM match value stored in the Timer PWM0 High and Low Byte registers. When the timer count value matches the PWM value, the Timer Outputs (TOUT and  $\overline{\text{TOUT}}$ ) toggle. The timer continues counting until it reaches the Reload value stored in the Timer Reload High and Low Byte registers. On 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 TOUT, and  $\overline{\text{TOUT}}$  toggles again and counting resumes.

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

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

The timer also generates a second PWM output signal, Timer Output Complement ( $\overline{\text{TOUT}}$ ).  $\overline{\text{TOUT}}$  is the complement of the Timer Output PWM signal (TOUT).

A programmable deadband delay can be configured to time delay (0 to 128 timer clock cycles) when one PWM output transition from High to Low and the other PWM output transition from a Low to High. This ensures a time gap between the removal of one PWM output and the assertion of its complement.

Follow the steps below for configuring a timer for PWM Dual Output mode and initiating the PWM operation:

1. Write to the Timer Control 1 register to:
  - Disable the timer.
  - Configure the timer for PWM Dual Output mode. Setting the mode also involves writing to TMODE[3] bit in TxCTL0 register.
  - Set the prescale value.
  - Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output Alternate Function.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H). This only affects the first pass in PWM mode. After the first timer reset in PWM mode, counting always begins at the reset value of 0001H.
3. Write to the Timer PWM0 High and Low Byte registers to set the PWM value.
4. Write to the Timer Control 0 register:
  - To set the PWM deadband delay value
  - To choose the timer clock source

5. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
6. 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.
7. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
8. Configure the associated GPIO port pin for the Timer Output and Timer Output Complement alternate functions.
9. Write to the Timer Control 1 register to enable the timer and initiate counting.

The PWM period is given by the following equation:

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

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 PWM timeout 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 PWM0 High and Low Byte Registers. The Timer counts timer clocks up to the 16-bit Reload value. The TPOL bit in the Timer Control 1 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 INPCAP bit in Timer Control 0 register is set to indicate the timer interrupt is due to an input capture event.

The timer continues counting up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. On reaching the Reload value, the timer generates an interrupt and continues counting. The INPCAP bit in Timer Control 0 register is cleared to indicate the timer interrupt is not due to an input capture event.

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

1. Write to the Timer Control 1 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 Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
5. Write to the Timer Reload High and Low Byte registers to set the Reload value.
6. 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 contain 0000H after the interrupt, then the interrupt was generated by a Reload.
7. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the input capture event or the reload event by setting TICONFIG field of the Timer Control 0 register.
8. Configure the associated GPIO port pin for the Timer Input alternate function.
9. Write to the Timer Control 1 register to enable the timer and initiate counting.

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

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

### **CAPTURE RESTART Mode**

In CAPTURE RESTART 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 counts timer clocks up to the 16-bit Reload value. The TPOL bit in the Timer Control 1 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 count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. The INPCAP bit in Timer Control 0 register is set to indicate the timer interrupt is due to an input capture event.

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. On 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. The INPCAP bit in Timer Control 0 register is cleared to indicate the timer interrupt is not due to an input capture event.

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

1. Write to the Timer Control 1 register to:
  - Disable the timer.
  - Configure the timer for CAPTURE RESTART mode. Setting the mode also involves writing to TMODE[3] bit in TxCTL0 register.
  - Set the prescale value.
  - Set the Capture edge (rising or falling) for the Timer Input.
2. Write to the Timer Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
5. Write to the Timer Reload High and Low Byte registers to set the Reload value.
6. Clear the Timer PWM High and Low Byte registers to 0000H. This allows user software to determine if interrupts are generated by either a Capture Event or a Reload. If the PWM High and Low Byte registers still contain 0000H after the interrupt, then the interrupt is generated by a Reload.
7. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the Input Capture event or the reload event by setting TICONFIG field of the Timer Control 0 register.
8. Configure the associated GPIO port pin for the Timer Input alternate function.
9. Write to the Timer Control 1 register to enable the timer and initiate counting.

In CAPTURE mode, the elapsed time from Timer start to Capture event can be calculated using the following equation:

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

## 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 counts timer clocks up to 16-bit Reload value. On 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) on 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 1 register to:
  - Disable the timer
  - Configure the timer for COMPARE mode
  - Set the prescale valu.
  - Set the initial logic level (High or Low) for the Timer Output alternate function, if required
2. Write to the Timer Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value.
5. Write to the Timer Reload High and Low Byte registers to set the Compare value.
6. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. When using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
8. Write to the Timer Control 1 register to enable the timer and initiate counting.

In COMPARE mode, the timer clock always provides the timer input. The Compare time is given by the following equation:

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

## 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 1 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

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 timer 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 1 register to:
  - Disable the timer
  - Configure the timer for GATED mode
  - Set the prescale value
2. Write to the Timer Control 2 register to choose the timer clock source.
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
4. 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.
5. Write to the Timer Reload High and Low Byte registers to set the Reload value.
6. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input deassertion and reload events. If required, configure the timer interrupt to be generated only at the Input Deassertion event or the Reload event by setting TICONFIG field of the Timer Control 0 register.
7. Configure the associated GPIO port pin for the Timer Input alternate function.
8. Write to the Timer Control 1 register to enable the timer.
9. 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 desired transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control 1 register. The Timer counts timer clocks up to the 16-bit Reload value.

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 PWM0 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. The INPCAP bit in Timer Control 0 register is set to indicate the timer interrupt is due to an input capture event.

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. On 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. The INPCAP bit in Timer Control 0 register is cleared to indicate the timer interrupt is not due to an input capture event.

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

1. Write to the Timer Control 1 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 Control 2 register to choose the timer clock source.
4. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
5. Write to the Timer Reload High and Low Byte registers to set the Compare value.
6. If required, enable the timer interrupt and set the timer-interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the input Capture event or the Reload event by setting TICONFIG field of the Timer Control 0 register.
7. Configure the associated GPIO port pin for the Timer Input alternate function.
8. Write to the Timer Control 1 register to enable the timer.
9. Counting begins on the first 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:

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

## **DEMODULATION Mode**

In DEMODULATION mode, the timer begins counting on the first external Timer Input transition. The desired transition (rising edge or falling edge or both) is set by the TPOL bit in the Timer Control 1 register and TPOLHI bit in the Timer Control 2 register. The Timer counts timer clocks up to the 16-bit Reload value.

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 PWM0 High and Low Byte Registers for rising input edges of the timer input signal. For falling edges the capture count value is written to the Timer PWM1 High and Low Byte Registers. The TPOL bit in the Timer Control 1 register determines if the Capture occurs on a rising edge or a falling edge of the Timer Input signal. If the TPOLHI bit in the Timer Control 2 register is set, Capture is done on both the rising and falling edges of the input signal.

Whenever the Capture event occurs, an interrupt is generated and the timer continues counting. The corresponding event flag bit PWMxEF in Timer Status register is set to indicate the timer interrupt is due to an input Capture event.

The timer counts up to the 16-bit Compare 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. The RTOEF event flag bit in Timer Status register is set to indicate the timer interrupt is due to a Reload event. Software can use this bit to determine if a reload occurred prior to a Capture.

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

1. Write to the Timer Control 1 register to:
  - Disable the timer.
  - Configure the timer for DEMODULATION mode. Setting the mode also involves writing to TMODEHI bit in TxCTL0 register.
  - Set the prescale value.
  - Set TPOL bit to set the Capture edge (rising or falling) for the Timer Input. This applies only if TPOLHI bit in TxCTL2 register is not set.
2. Write to the Timer Control 2 register to:
  - Choose the timer clock source
  - Set the TPOLHI bit if the Capture is required on both edges of the input signal
3. Write to the Timer Control 0 register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
5. Write to the Timer Reload High and Low Byte registers to set the Reload value.

6. Clear the Timer TxPWM0 and TxPWM1 High and Low Byte registers to 0000H.
7. If required, enable the noise filter and set the noise filter control by writing to the relevant bits in the Noise Filter Control Register.
8. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the input Capture event or the Reload event by setting TICONFIG field of the Timer Control 0 register.
9. Configure the associated GPIO port pin for the Timer Input alternate function.
10. Write to the Timer Control 1 register to enable the timer. Counting will start on the occurrence of the first external input transition.

In DEMODULATION mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

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

Table 51 provides an example initialization sequence for configuring Timer 0 in DEMODULATION mode and initiating operation.

**Table 51. DEMODULATION Mode Initialization Example**

Register	Value	Comment
T0CTL0	C0H	TMODE[3:0] = 1100B selects DEMODULATION mode.
T0CTL1	04H	TICONFIG[1:0] = 10B enables interrupt only on Capture events.
T0CTL2	11H	CSC = 0 selects the Timer Input from the GPIO pin.
		PWMD[2:0] = 000B has no effect.
		INPCAP = 0 has no effect.
		TEN = 0 disables the timer.
		PRES[2:0] = 000B sets prescaler to divide by 1.
		TPOLHI,TPOL = 10 enables trigger and Capture on both rising and falling edges of Timer Input.
		TCLKS = 1 enables 32 kHz peripheral clock as timer clock source
T0H	00H	Timer starting value = 0001H
T0L	01H	
T0RH	ABH	Timer reload value = ABCDH
T0RL	CDH	
T0PWM0H	00H	Initial PWM0 value = 0000H
T0PWM0L	00H	

**Table 51. DEMODULATION Mode Initialization Example (Continued)**

Register	Value	Comment
T0PWM1H	00H	Initial PWM1 value = 0000H
T0PWM1H	00H	
T0NFC	C0H	NFEN = 1 enables noise filter NFCTL = 100B enables 8-bit up/down counter
PAADDR	02H	Selects Port A Alternate Function control register.
PACTL[1:0]	11B	PACTL[0] enables Timer 0 Input alternate function. PACTL[1] enables Timer 0 Output alternate function.
IRQ0ENH[5]	0B	Disables the Timer 0 interrupt.
IRQ0ENL[5]	0B	
T0CTL1	84H	TEN = 1 enables the timer. All other bits left in desired settings.

Notes: After receiving the input trigger (rising or falling edge), Timer 0 will:

1. Start counting on timer clock.
2. Upon receiving a Timer 0 Input rising edge, save Capture value in T0PWM0 registers, generate interrupt, and continue to count.
3. Upon receiving a Timer 0 Input falling edge, save Capture value in T0PWM1 registers, generate interrupt, and continue to count.
4. After timer count to ABCD clocks, set reload event flag, and reset Timer count to start value.

## 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 Noise Filter

A Noise Filter circuit is included which filters noise on a Timer Input signal before the data is sampled by the block.

The Noise Filter has the following features:

- Synchronizes the receive input data to the Timer Clock
- NFEN (Noise Filter Enable) input selects whether the Noise Filter is bypassed (NFEN=0) or included (NFEN=1) in the receive data path
- NFCTL (Noise Filter Control) input selects the width of the up/down saturating counter digital filter. The available widths range from 4 bits to 11 bits
- The digital filter output has hysteresis
- Provides an active low “Saturated State” output (FiltSatB) which is used as an indication of the presence of noise
- Available for operation in STOP mode

## Architecture

Figure 12 illustrates how the Noise Filter is integrated with the Timer.

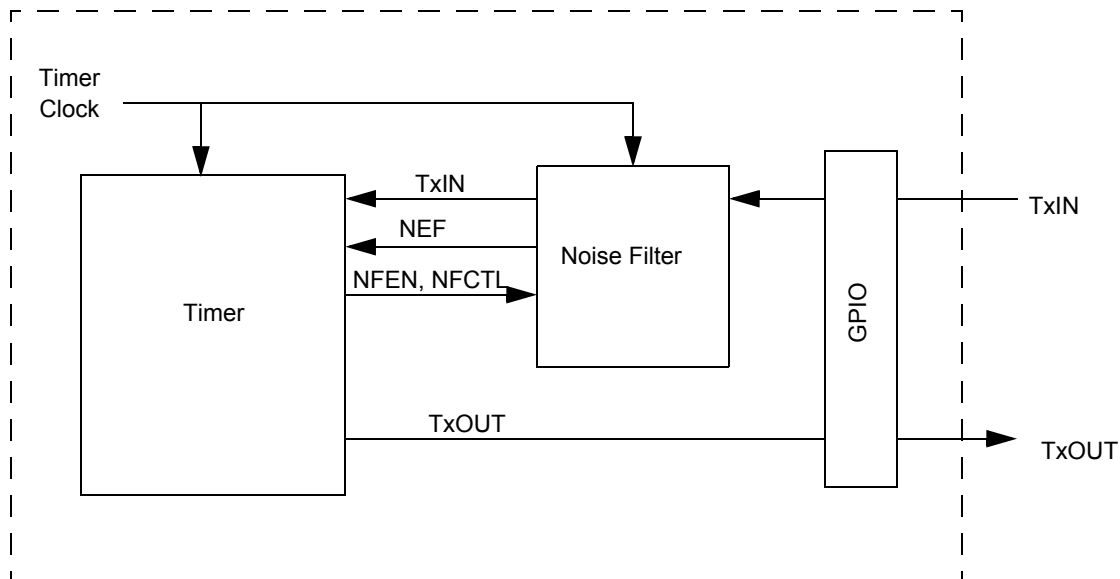


Figure 12. Noise Filter System Block Diagram

## Operation

Figure 13 on page 106 illustrates the operation of the Noise Filter with and without noise. The Noise Filter in this example is a 2-bit up/down counter which saturates at 00 and 11. A 2-bit counter is described for convenience, the operation of wider counters is similar. The output of the filter switches from 1 to 0 when the counter counts down from 01 to 00 and

switches from 0 to 1 when the counter counts up from 10 to 11. The Noise Filter delays the receive data by three Timer Clock cycles.

The NEF output signal is checked when the filtered TxIN input signal is sampled. The Timer samples the filtered TxIN input near the center of the bit time. The NEF signal must be sampled at the same time to detect whether there is noise near the center of the bit time. The presence of noise (NEF = 1 at center of bit time) does not mean the sampled data is incorrect, rather it is intended to be an indicator of the level of noise in the network.

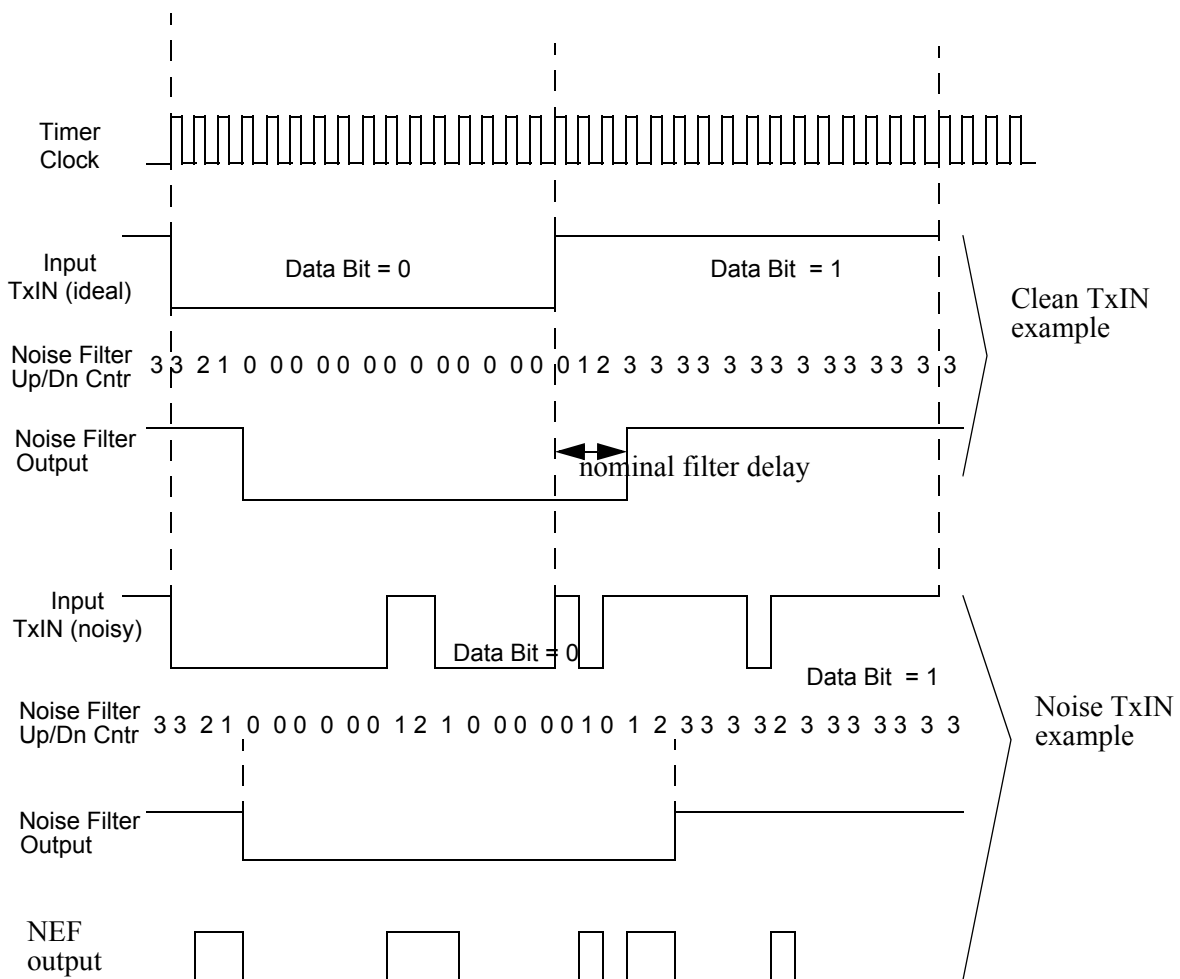


Figure 13. Noise Filter Operation

## Timer Control Register Definitions

### Timer 0–2 High and Low Byte Registers

The Timer 0–2 High and Low Byte (TxH and TxL) registers (Table 52 and Table 53) 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 TxL always returns this temporary register when the timers are enabled. When the timer is disabled, reading from the TxL reads the register directly.

Writing to the Timer High and Low Byte registers when 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 52. Timer 0–2 High Byte Register (TxH)**

BITS	7	6	5	4	3	2	1	0
FIELD	TH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F00H, F08H, F10H							

**Table 53. Timer 0–2 Low Byte Register (TxL)**

BITS	7	6	5	4	3	2	1	0
FIELD	TL							
RESET	0	0	0	0	0	0	0	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F01H, F09H, F11H							

#### TH and TL—Timer High and Low Bytes

These 2 bytes, {TH[7:0], TL[7:0]}, contain the current 16-bit timer count value.

## Timer Reload High and Low Byte Registers

The Timer 0–2 Reload High and Low Byte (TxRH and TxRL) registers (Table 54 and Table 55) 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 54. Timer 0–2 Reload High Byte Register (TxRH)**

BITS	7	6	5	4	3	2	1	0
FIELD	TRH							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F02H, F0AH, F12H							

**Table 55. Timer 0–2 Reload Low Byte Register (TxRL)**

BITS	7	6	5	4	3	2	1	0
FIELD	TRL							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F03H, F0BH, F13H							

### 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 is used to set the maximum count value which initiates a timer reload to 0001H. In COMPARE mode, these two bytes form the 16-bit Compare value.

## Timer 0–2 PWM0 High and Low Byte Registers

The Timer 0–2 PWM0 High and Low Byte (TxPWM0H and TxPWM0L) registers (see Table 56 and Table 57 on page 109) are used for Pulse Width Modulator (PWM) operations. These registers also store the Capture values for the CAPTURE, CAPTURE/COMPARE and DEMODULATION modes. When the timer is enabled, writes to these registers are buffered and loading of the registers is delayed until a timer reload to 0001H unless PWM0UE = 1

**Table 56. Timer 0–2 PWM0 High Byte Register (TxPWM0H)**

BITS	7	6	5	4	3	2	1	0
FIELD	PWM0H							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F04H, F0CH, F14H							

**Table 57. Timer 0–2 PWM0 Low Byte Register (TxPWM0L)**

BITS	7	6	5	4	3	2	1	0
FIELD	PWM0L							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F05H, F0DH, F15H							

#### **PWM0H and PWM0L—Pulse Width Modulator 0 High and Low Bytes**

These two bytes, {PWM0H[7:0], PWM0L[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 1 register (TxCTL1).

The TxPWM0H and TxPWM0L registers also store the 16-bit captured timer value when operating in CAPTURE, CAPTURE/COMPARE, and DEMODULATION modes.

### **Timer 0–2 PWM1 High and Low Byte Registers**

The Timer 0–2 PWM1 High and Low Byte (TxPWM1H and TxPWM1L) registers (see [Table 58](#) and [Table 59](#)) store Capture values for DEMODULATION mode.

**Table 58. Timer 0-2 PWM1 High Byte Register (TxPWM1H)**

BITS	7	6	5	4	3	2	1	0
FIELD	PWM1H							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F20H, F24H, F28H							

Table 59. Timer 0–2 PWM1 Low Byte Register (TxPWM1L)

BITS	7	6	5	4	3	2	1	0
FIELD	PWM1L							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F21H, F25H, F29H							

#### PWM1H and PWM1L—Pulse Width Modulator 1 High and Low Bytes

These two bytes, {PWM1H[7:0], PWM1L[7:0]}, store the 16-bit captured timer value for the DEMODULATION mode.

## Timer 0–2 Control Registers

### Time 0–2 Control 0 Register

The Timer 0–2 Control 0 (TxCTL0) register together with TxCTL1 register determines the timer operating mode. It also includes a programmable PWM deadband delay, two bits to configure timer interrupt definition and a status bit to identify if the last timer interrupt is due to an input capture event.

Table 60. Timer 0–2 Control 0 Register (TxCTL0)

BITS	7	6	5	4	3	2	1	0
FIELD	TMODE[3]	TICONFIG		CSC	PWMD			INPCAP
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F06H, F0EH, F16H							

#### TMODE[3]—Timer Mode High Bit

This bit along with the TMODE[2:0] field in TxCTL1 register determines the operating mode of the timer. This is the most significant bit of the Timer mode selection value. For more details, see TxCTL1 register description in [Table 61](#) on page 111.

#### TICONFIG—Timer Interrupt Configuration

This field configures timer interrupt definition.

0x = Timer Interrupt occurs on all defined Reload, Compare and Input Events.

10 = Timer Interrupt only on defined Input Capture/Deassertion Events.

11 = Timer Interrupt only on defined Reload/Compare Events.

#### CSC—Cascade Timers

0 = Timer Input signal comes from the pin.

1 = For Timer 0, Input signal is connected to Timer 2 output.

For Timer 1, Input signal is connected to Timer 0 output.

For Timer 2, Input signal is connected to Timer 1 output.

#### PWMD—PWM Delay Value

This field is a programmable delay to control the number of timer clock cycles time delay before the Timer Output and the Timer Output Complement is forced to their active state.

000 = No delay

001 = 2 cycles delay

010 = 4 cycles delay

011 = 8 cycles delay

100 = 16 cycles delay

101 = 32 cycles delay

110 = 64 cycles delay

111 = 128 cycles delay

#### INPCAP—Input Capture Event

This bit indicates if the last timer interrupt is due to a Timer Input Capture Event.

0 = Previous timer interrupt is not a result of Timer Input Capture Event

1 = Previous timer interrupt is a result of Timer Input Capture Event

#### Timer 0–2 Control 1 Register

The Timer 0–2 Control 1 (TxCTL1) registers enable/disable the timers, set the prescaler value, and determine the timer operating mode.

**Table 61. Timer 0–2 Control 1 Register (TxCTL1)**

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	TPOL	PRES			TMODE		
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F07H, F0FH, F17H							

#### TEN—Timer Enable

0 = Timer is disabled.

1 = Timer enabled to count.

#### TPOL—Timer Input/Output Polarity

Operation of this field is a function of the current operating modes 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**

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.

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 Single Output mode**

0 = Timer Output is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced High (1) on PWM count match and forced Low (0) on Reload.

1 = Timer Output is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced Low (0) on PWM count match and forced High (1) on 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 on 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.

### **PWM Dual Output mode**

- 0 = Timer Output is forced Low (0) and Timer Output Complement is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon Reload. When enabled, the Timer Output Complement is forced Low (0) upon PWM count match and forced High (1) upon Reload. The PWMD field in Timer Control 0 register is a programmable delay to control the number of cycles time delay before the Timer Output and the Timer Output Complement is forced to High (1).
- 1 = Timer Output is forced High (1) and Timer Output Complement is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon Reload. When enabled, the Timer Output Complement is forced High (1) upon PWM count match and forced Low (0) upon Reload. The PWMD field in Timer Control 0 register is a programmable delay to control the number of cycles time delay before the Timer Output and the Timer Output Complement is forced to Low (0).

### **CAPTURE RESTART 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.

### **CAMPARATOR COUNTER 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.

### **Triggered ONE-SHOT mode**

- 0 = Timer counting is triggered on the rising edge of the Timer Input signal.
- 1 = Timer counting is triggered on the falling edge of the Timer Input signal.

### **DEMULATION mode**

- 0 = Timer counting is triggered on the rising edge of the Timer Input signal.  
The current count is captured into PWM0 High and Low byte registers on subsequent rising edges of the Timer Input signal.
- 1 = Timer counting is triggered on the falling edge of the Timer Input signal.  
The current count is captured into PWM1 High and Low byte registers on subsequent falling edges of the Timer Input signal.

The above functionality applies only if TPOLHI bit in Timer Control 2 register is 0. If TPOLHI bit is 1 then timer counting is triggered on any edge of the Timer Input signal and the current count is captured on both edges. The current count is captured into PWM0 registers on rising edges and PWM1 registers on falling edges of the Timer Input signal.

#### **PRES—Prescale value**

The timer input clock is divided by 2PRES, where PRES can be 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[2:0]—Timer mode**

This field along with the TMODE[3] bit in TxCTL0 register determines the operating mode of the timer. TMODE[3:0] selects among the following modes:

0000 = ONE-SHOT mode  
0001 = CONTINUOUS mode  
0010 = COUNTER mode  
0011 = PWM Single Output mode  
0100 = CAPTURE mode  
0101 = COMPARE mode  
0110 = GATED mode  
0111 = CAPTURE/COMPARE mode  
1000 = PWM Dual Output mode  
1001 = CAPTURE RESTART mode  
1010 = COMPARATOR COUNTER mode  
1011 = Triggered ONE-SHOT mode  
1100 = DEMODULATION mode

#### **Timer 0–2 Control 2 Register**

The Timer 0–2 Control 2 (TxCTL2) registers allow selection of timer clock source and control of timer input polarity in DEMODULATION mode.

**Table 62. Timer 0–2 Control 2 Register (TxCTL2)**

<b>BITS</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>FIELD</b>	Reserved		PWM0UE	TPOLHI	Reserved			TCLKS
<b>RESET</b>	0	0	0	0	0	0	0	0
<b>R/W</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>ADDR</b>	F22H, F26H, F2AH							

#### PWM0UE—PWM0 Update Enable

This bit determines whether writes to the PWM0 High and Low Byte registers are buffered when TEN = 1. Writes to these registers are not buffered when TEN = 0 regardless of the value of this bit.

0 = Writes to the Channel High and Low Byte registers are buffered when TEN = 1 and only take affect on a timer reload to 0001H.

1 = Writes to the Channel High and Low Byte registers are not buffered when TEN = 1.

#### TPOLHI—Timer Input/Output Polarity High Bit

This bit determines if timer count is triggered and captured on both edges of the input signal. This applies only to DEMODULATION mode.

0 = Count is captured only on one edge in DEMODULATION mode. In this case, edge polarity is determined by TPOL bit in TxCTL1 register.

1 = Count is triggered on any edge and captured on both rising and falling edges of the Timer Input signal in DEMODULATION mode

#### Reserved—Must be 0

#### TCLKS—Timer Clock Source

0 = System Clock

1 = Peripheral Clock

## Timer 0–2 Status Registers

The Timer 0–2 Status (TxSTAT) indicates PWM capture/compare event occurrence, overrun errors, noise event occurrence and reload timeout status.

**Table 63. Timer 0–2 Status Register (TxSTAT)**

BITS	7	6	5	4	3	2	1	0
FIELD	NEF	Reserved	PWM1EO	PWM0EO	RTOEF	Reserved	PWM1EF	PWM0EF
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F23H, F27H, F2BH							

#### NEF—Noise Event Flag

This status is applicable only if the Timer Noise Filter is enabled. The NEF bit will be asserted if digital noise is detected on the Timer input (TxIN) line when the data is being sampled (center of bit time). If this bit is set, it does not mean that the timer input data is corrupted (though it may be in extreme cases), just that one or more Noise Filter data samples near the center of the bit time did not match the average data value.

#### PWMxEO—PWM x Event Overrun

This bit indicates that an overrun error has occurred. An overrun occurs when a new capture/compare event occurs before the previous PWMxEF bit is cleared. Clearing the associated PWMxEF bit in the TxSTAT register clears this bit.

0 = No Overrun

1 = Capture/Compare Event Flag Overrun

#### RTOEF—Reload Timeout Event Flag

This flag is set if timer counts up to the reload value and is reset to 0001H.

Software can use this bit to determine if a reload occurred prior to a capture.

It can also determine if timer interrupt is due to a reload event.

0 = No Reload Timeout event occurred

1 = A Reload Timeout event occurred

#### Reserved—Must be 0

#### PWMxEF—PWM x Event Flag

This bit indicates if a capture/compare event occurred for this PWM channel.

Software can use this bit to determine the PWM channel responsible for generating the timer interrupt. This event flag is cleared by writing a 1 to the bit. These bits will be set when an event occurs independent of the setting of the timer interrupt enable bit.

0 = No Capture/Compare Event occurred for this PWM channel

1 = A Capture/Compare Event occurred for this PWM channel

## Timer 0–2 Noise Filter Control Register

The Timer 0–2 Noise Filter Control Register (TxNFC) enable/disable Timer Noise Filter and set the noise filter control.

Table 64. Timer 0–2 Noise Filter Control Register (TxNFC)

BITS	7	6	5	4	3	2	1	0
FIELD	NFEN	NFCTL			Reserved			
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W			R			
ADDR	F2CH, F2DH, F2EH							

#### NFEN—Noise Filter Enable

0 = Noise Filter is disabled.

1 = Noise Filter is enabled. Receive data is preprocessed by the noise filter.

**NFCTL—Noise Filter Control**

This field controls the delay and noise rejection characteristics of the Noise Filter. The wider the counter the more delay that is introduced by the filter and the wider the noise event that will be filtered.

000 = 2-bit up/down counter

001 = 3-bit up/down counter

010 = 4-bit up/down counter

011 = 5-bit up/down counter

100 = 6-bit up/down counter

101 = 7-bit up/down counter

110 = 8-bit up/down counter

111 = 9-bit up/down counter

**Reserved—Must be 0**



# Multi-Channel Timer

## Overview

The Multi-Channel timer has a 16-bit up/down counter and a 4-channel Capture/Compare/PWM channel array. This timer enables the support of multiple synchronous Capture/Compare/PWM channels based on a single timer. The Multi-Channel Timer features include:

- 16-bit up/down timer counter with programmable prescale.
- Selectable clock source (system clock or external input pin).
- Count Modulo and Count up/down COUNTER modes.
- Four independent capture/compare channels which reference the common timer.
- Channel modes:
  - ONE-SHOT compare mode
  - CONTINUOUS compare mode
  - PWM Output compare mode
  - CAPTURE mode

## Architecture

Figure 14 illustrates the Multi-Channel Timer architecture.

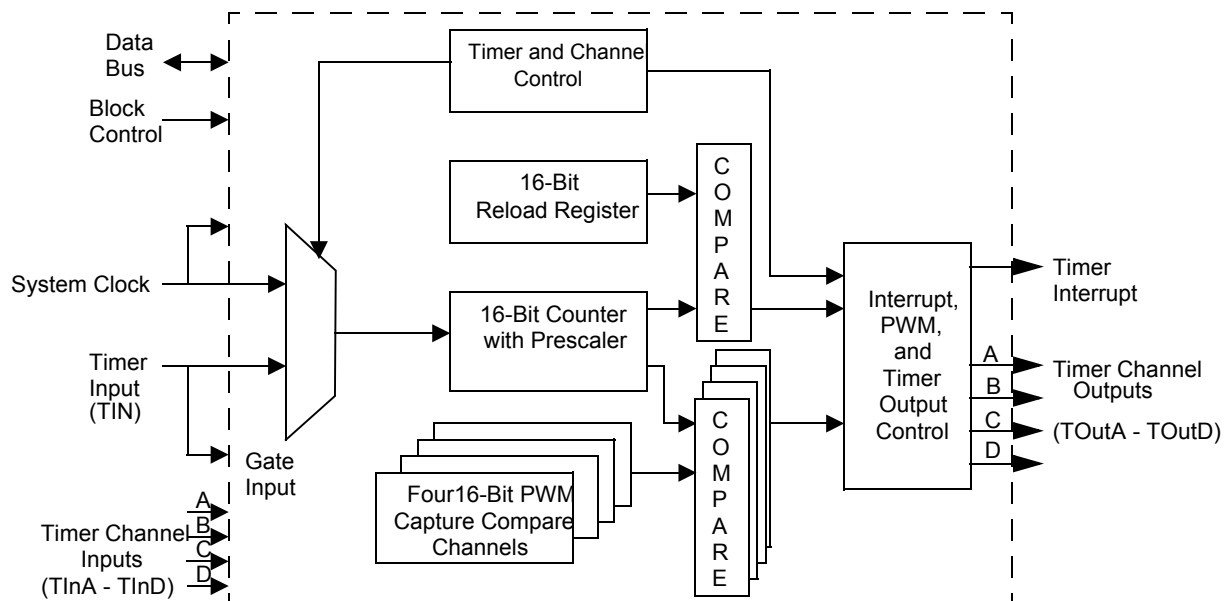


Figure 14. Multi-Channel Timer Block Diagram

## Timer Operation

### Multi-Channel Timer Counter

The Multi-Channel Timer is based around a 16-bit up/down counter. The counter, depending on the TIMER mode counts up or down with each rising edge of the clock signal. Timer Counter Registers MCTH and MCTL can be read/written by software.

### Clock Source

The Multi-Channel Timer clock source can come from either the system clock or the system clock gated by the alternate function TIN pin, the alternate function TIN input pin operating as a clock input of gated clock. The TCLKS field in the MCTCTL0 register selects the timer clock source. When using the TIN pin, the associated GPIO pin must be configured as an input. The TIN frequency cannot exceed one-fourth the system clock frequency.

### Multi-Channel Timer Clock Prescaler

The prescaler allows the system clock signal to be decreased by factors of 1, 2, 4, 8, 16, 32, 64, or 128. The PRES[2:0] bit field in the MCTCTL1 register controls prescaler operation. The PRES field is buffered for the prescale value to change only on a MCT end of cycle count. The prescaler has no effect when the TIN is selected as the clock source.

### Multi-Channel Timer Start

The Multi-Channel Timer starts counting when TEN bit in MCTCTL1 register is set and the clock source is active. In Count Modulo or Count up/down mode the timer counting can be stopped without disabling the timer by setting the Reload Register to 0. The timer will then stop when the counter next reaches 0. Writing a nonzero value to the Reload Register restarts the timer counting.

### Multi-Channel Timer Mode Control

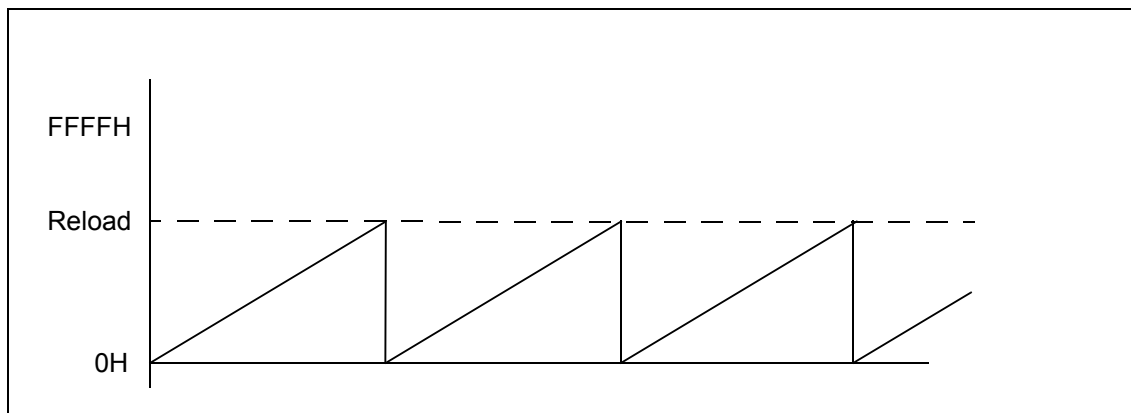
The Multi-Channel Timer supports two modes of operation: Count Modulo and Count up/down. The operating mode is selected with the TMODE[1:0] field in MCTCTL1 register. The timer modes are described below in [Table 65](#) on page 121.

**Table 65. Timer Count Modes**

TMODE	Timer Mode	Description
00	Count Modulo	Timer counts up to Reload Register value. Then it is reset to 0000H and counting resumes.
01	Reserved	
10	Count Up/Down	Timer counts up to Reload and then counts down to 0000H. The Count up/down cycle continues.
11	Reserved	

### Count Modulo Mode

In the Count Modulo mode, the Timer counts up to the Reload Register value (max value = FFFFH). Then it is reset to 0000H and counting resumes. As displayed in [Figure 15](#) the counting cycle continues with Reload + 1 as the period. A timer count interrupt request is generated when the timer count resets from Reload to 0000H. If Count Modulo is selected when the timer count is greater than Reload, the timer immediately restarts counting from zero.



**Figure 15. Count Modulo Mode**

### Count Up/Down Mode

In the Count Up/Down mode, the timer counts up to the Reload Register value and then counts down to 0000H. As illustrated in [Figure 16](#) on page 122, the counting cycle continues with twice the reload value as the period. A timer count interrupt is generated when the timer count decrements to zero.

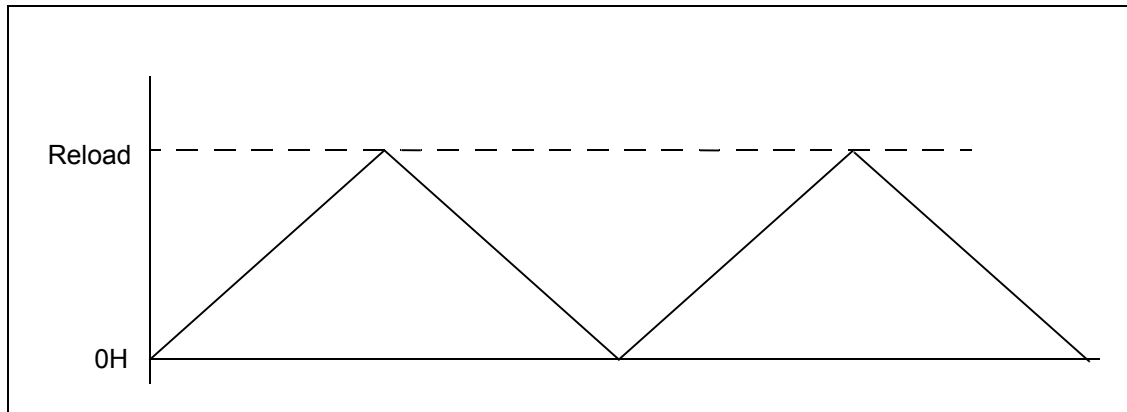


Figure 16. Count Up/Down Mode

## Capture/Compare Channel Operation

The Multi-Channel timer supports four Capture/Compare channels: CHA, CHB, CHC, and CHD. Each channel has the following features:

- A 16-bit Capture/Compare Register (MCTCHyH and MCTCHyL registers) used to capture input event times or to generate time intervals. Any user software update of the Capture/Compare Register value when the timer is running takes effect only at the end of the counting cycle, not immediately. The end of the counting cycle is when the counter transitions from the Reload value to 0 (count modulo mode) or from 1 to 0 (count up/down mode).
- A dedicated bidirectional pin (TInA, B, C, or D) that can be configured for the input capture function or to generate an output compare match or one-shot pulse.

Each channel is configured to operate in either ONE-SHOT Compare, CONTINUOUS Compare, PWM Output, or Input CAPTURE mode.

## One-Shot Compare Operation

In One-Shot compare operation, a channel interrupt is generated when the channel compare value matches the timer count. The channel event flag (CHyEF) is set in the Channel Status1 register (MCTCHS1) to identify the responsible channel. Then the channel is automatically disabled. The timer continues counting according to the programmed mode. If the timer channel output alternate function is enabled, the channel output pin (TOutA, B, C, or D) changes state for one system clock cycle (from Low to High then back to Low or High to Low then back to High as determined by the CHPOL bit) on match.

## Continuous Compare Operation

In Continuous Compare operation, a channel interrupt is generated when the channel compare value matches the timer count. The channel event flag (CHyEF) is set in the Channel Status1 register (MCTCHS1) and the channel remains enabled. The timer continues counting according to the programmed mode. If the channel output alternate function is enabled, the channel output pin (TOutA, B, C, or D) changes state (from Low to High then back to Low, or High to Low then back to High as determined by the CHPOL bit) on match.

## PWM Output Operation

In PWM Output operation, the timer generates a PWM output signal on the channel output pin (TOutA, B, C, or D). The channel output toggles whenever the timer count matches the channel compare value (defined in the MCTCHyH and MCTCHyL) registers. In addition, a channel interrupt is generated and the channel event flag is set in the status register. The timer continues counting according to its programmed mode.

The channel output signal begins with the output value = CHPOL and then transitions to  $\overline{\text{CHPOL}}$  when timer value matches the PWM value. If timer mode is Count Modulo, the channel output signal returns to output = CHPOL when timer reaches the Reload value and is reset. If timer mode is Count up/down, channel output signal returns to output = CHPOL when the timer count matches the PWM value again (when counting down).

## Capture Operation

In Capture operation, the current timer count is recorded when the selected transition occurs on TInA, B, C or D. The Capture count value is written to the Channel High and Low Byte Registers. In addition, a channel interrupt is generated and the channel event flag (CHyEF) is set in the Channel Status register. The CHPOL bit in the Channel Control register determines if the Capture occurs on a rising edge or a falling edge of the Channel Input signal. The timer continues counting according to the programmed mode.

## Multi-Channel Timer Interrupts

The Multi-Channel Timer provides a single interrupt which has five possible sources. These sources are the internal timer and the four channel inputs (TInA, TInB, TInC, TInD).

### Timer Interrupt

If enabled by TCIEN bit of the MCTCTL0 register, the timer interrupt will be generated when the timer completes a count cycle. This occurs during transition from counter = reload register value to counter = 0 in count modulo mode, and occurs during transition from counter = 1 to counter = 0 in count up/down mode.

## Capture/Compare Channel Interrupt

A channel interrupt is generated whenever there is a successful Capture/Compare Event on the Timer Channel and the associated CHIEN bit is set.

## Low-Power Modes

### Operation in HALT Mode

When the eZ8 CPU is operating in HALT mode, the Multi-Channel Timer will continue to operate if enabled. To minimize current in HALT mode, the Multi-Channel Timer must be disabled by clearing the TEN control bit.

### Operation in STOP Mode

When the eZ8 CPU is operating in STOP mode, the Multi-Channel Timer ceases to operate as the system clock is stopped. The registers are not reset and operation will resume once Stop Mode Recovery occurs.

### Power Reduction During Operation

Deassertion of the TEN bit will inhibit clocking of the entire Multi-Channel Timer block. Deassertion of the CHEN bit of individual channels will inhibit clocking of channel specific logic to minimize power consumption of unused channels. The CPU can still read/write registers when the enable bit(s) are deasserted.

## Multi-Channel Timer Applications Examples

### PWM Programmable Deadband Generation

The count up/down mode supports motor control applications that require dead time between output signals. [Figure 17](#) on page 125 illustrates dead time generation between two channels operating in count up/down mode.

### Multiple Timer Intervals Generation

[Figure 18](#) on page 125 illustrates generation of two constant time intervals  $t_0$  and  $t_1$ . The timer is in Count Modulo mode with reload = FFFFH. Channels 0 and 1 are set up for Continuous Compare operation. After every channel compare interrupt, the channel Capture/Compare registers are updated in the interrupt service routine by adding a constant equal to the time interval required. This operation requires that the CHUE bit (Channel Update Enable) must be set in channels 0 and 1 so that writes to the Capture/Compare registers take affect immediately.

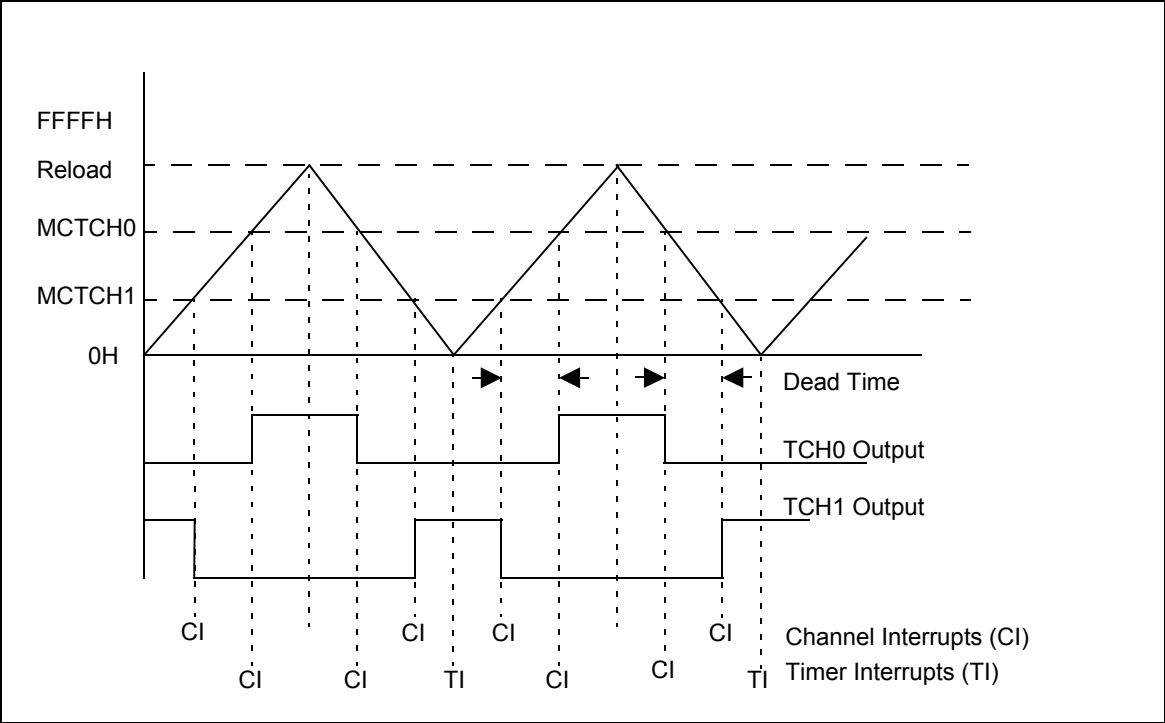


Figure 17. Count Up/Down Mode with PWM Channel Outputs and Deadband

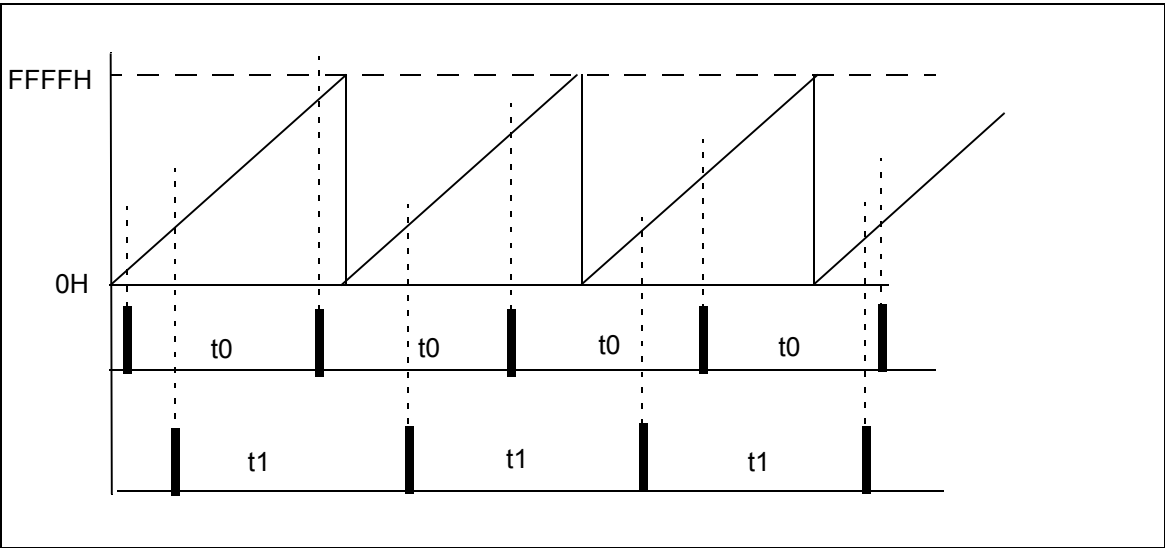


Figure 18. Count Max Mode with Channel Compare

## Multi-Channel Timer Control Register Definitions

### Multi-Channel Timer Address Map

Table 66 defines the byte address offsets for the Multi-channel Timer registers. For saving address space, sub-address is used for Timer Control 0 register, Timer Control 1 register, Channel Status 0 register, Channel Status 1 register, Channel-y Control register, Channel-y High, and Low byte register. Only Timer High and Low byte register and Reload High and Low byte register can be directly accessed.

While writing a subregister, first write sub-address to Timer Sub-Address Register, then write data to subregister0, subregister1, or subregister2. Read is the same as Write.

**Table 66. Multi-Channel Timer Address Map**

Address/Sub-address	Register/Subregister Name
<b>Direct Access Register</b>	
FA0	Timer (Counter) High
FA1	Timer (Counter) Low
FA2	Timer Reload High
FA3	Timer Reload Low
FA4	Timer Sub-Address
FA5	SubRegister 0
FA6	SubRegister 1
FA7	SubRegister 2
<b>Subregister 0</b>	
0	Timer Control 0
1	Channel Status 0
2	Channel A Capture/Compare High
3	Channel B Capture/Compare High
4	Channel C Capture/Compare High
5	Channel D Capture/Compare High
<b>Subregister 1</b>	
0	Timer Control 1
1	Channel Status 1
2	Channel A Capture/Compare Low
3	Channel B Capture/Compare Low
4	Channel C Capture/Compare Low
5	Channel D Capture/Compare High

**Table 66. Multi-Channel Timer Address Map (Continued)**

Address/Sub-address	Register/Subregister Name
<b>Subregister 2</b>	
0	Reserved
1	Reserved
2	Channel A Control
3	Channel B Control
4	Channel C Control
5	Channel D Control

## Multi-Channel Timer High and Low Byte Registers

The High and Low Byte (MCTH and MCTL) registers (see [Table 67](#) and [Table 68](#)) contain the current 16-bit MCT count value.

Writing to the MCT High and Low Byte registers while the MCT is enabled is not recommended. If either or both the MCT High or Low byte registers are written during counting, the 8-bit written value is placed in the counter (High and/or Low byte) at the next system clock edge. The counter continues counting from the new value.

**Table 67. MCT High Byte Register (MCTH)**

BITS	7	6	5	4	3	2	1	0
FIELD	MCTH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FA0H							

**Table 68. MCT Low Byte Register (MCTL)**

BITS	7	6	5	4	3	2	1	0
FIELD	MCTL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FA1H							

When MCT is enabled, a read from MCTH causes the value in MCTL to be stored in a temporary holding register. A read from MCTL returns this temporary register when MCT is enabled. When MCT is disabled, reads from MCTL read the register directory. The MCT High and Low byte registers are not reset when TEN = 0.

#### MCTH and MCTL—MCT High and Low Bytes

These 2 bytes, {MCTH[7:0], MCTL[7:0]}, contain the current 16-bit MCT count value.

### MCT Reload High and Low Byte Registers

The MCT Reload High and Low Byte (MCTRH and MCTRL) registers (see [Table 69](#) and [Table 70](#)) store a 16-bit reload value, {MCTRH[7:0], MCTRL[7:0]}. When TEN = 0, writes to this address update the register on the next clock cycle. When TEN = 1 writes to this register are buffered and transferred into the register when the counter reaches the end of the count cycle.

$$\text{Modulo Mode Period} = \frac{\text{Prescaler} \times (\text{Reload Value} + 1)}{f_{\text{MCTclk}}}$$

$$\text{Up/Down Mode Period} = \frac{2 \times \text{Prescaler} \times \text{Reload Value}}{f_{\text{MCTclk}}}$$

**Table 69. MCT Reload High Byte Register (MCTRH)**

BITS	7	6	5	4	3	2	1	0
FIELD	MCTRH							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FA2H							

**Table 70. MCT Reload Low Byte Register (MCTRL)**

BITS	7	6	5	4	3	2	1	0
FIELD	MCTRL							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FA3H							

The value written to the MCTRH is stored in a temporary holding register. When a write to the MCTRL occurs, the temporary holding register value is written to the MCTRH. This operation allows simultaneous updates of the 16-bit MCT Reload value.

#### MCTRH and MCTRL—MCT Reload Register High and Low

These two bytes form the 16-bit Reload value, {MCTRH[7:0], MCTRL[7:0]}. This value sets the MCT period in Modulo and Up/Down Count modes.

### MCT Sub-Address Register

The MCT Sub-Address Register stores 3-bit sub-addresses for subregisters. These three bits are from MCTSAR[2:0], all other bits are reserved. When accessing subregister (writing or reading), set MCTSA right value first, then access subregister by Writing or Reading SubRegisters 0, 1, or 2.

Table 71. MCT Sub-Address Register (MCTSA)

BITS	7	6	5	4	3	2	1	0
FIELD	MCTSA							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FA4H							

### MCT SubRegister x (0, 1, or 2)

The MCT Subregisters 0, 1, or 2 store the 8-bit data write to subregister or 8-bit data read from subregister. The MCT Sub-Address register selects the subregister to be written to or read from.

Table 72. MCT SubRegister x (MCTSRx)

BITS	7	6	5	4	3	2	1	0
FIELD	MCTSRx							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FA5H, FA6H, FA7H							

## Multi-Channel Timer Control 0, Control 1 Registers

The Multi-Channel Timer Control registers (MCTCTL0, MCTCTL1) control Multi-Channel Timer operation. Writes to the PRES field of MCTCTL1 register are buffered when TEN = 1, and will not take effect until the next end of cycle count occurs.

**Table 73. Multi-Channel Timer Control 0 Register (MCTCTL0)**

BITS	7	6	5	4	3	2	1	0
FIELD	TCTST	CHST	TCIEN	Reserved	Reserved	TCLKS		
RESET	0	0	0	0	0	0	0	0
R/W	R/W1C	R	R/W	R	R	R/W	R/W	R/W
ADDR	00H in Sub-Address Register, accessible through SubRegister 0							

### TCTST—Timer Count Status

This bit indicates if a timer count cycle is complete and is cleared by writing 1 to the bit and is cleared when TEN = 0.

0 = Timer count cycle is not complete.

1 = Timer count cycle is complete.

### CHST—Channel Status

This bit indicates if a channel Capture/Compare event occurred. This bit is the logical OR of the CHyEF bits in the MCTCHS1 register. This bit is cleared when TEN=0.

0 = No channel capture/compare event has occurred.

1 = A channel capture/compare event has occurred. One or more of the CHDEF, CHCEF, CHBEF, and CHAEF bits in the MCTCHS1 register are set.

### TCIEN—Timer Count Interrupt Enable

This bit enables generation of timer count interrupt. A timer count interrupt is generated whenever the timer completes a count cycle: counting up to Reload Register value or counting down to zero depending on whether the TIMER mode is Count Modulo or Count up/down.

0 = Timer Count Interrupt is disabled.

1 = Timer Count Interrupt is enabled.

### TCLKS—Timer Clock Source

000 = System Clock (Prescaling enabled)

001 = Reserved

010 = System Clock gated by active High Timer Input signal (Prescaling enabled).

011 = System Clock gated by active Low Timer Input signal (Prescaling enabled).

100 = Timer I/O pin input rising edge (Prescaler disabled).

101 = Timer I/O pin input falling edge (Prescaler disabled).

110 = Reserved.

111 = Reserved.

► **Note:** *The input frequency of the Timer Input Signal must not exceed one-fourth the system clock frequency.*

**Table 74. Multi-Channel Timer Control 1 Register (MCTCTL1)**

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	Reserved	PRES			Reserved	TMODE	
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R	R/W	R/W	R/W	R	R/W	R/W
ADDR	00H in Sub-Address Register, accessible through SubRegister 1							

**TEN—Timer Enable**

0 = Timer is disabled and the counter is reset.

1 = Timer is enabled to count.

**PRES—Prescale value**

The system clock is divided by 2<sup>PRES</sup>, where PRES can be set from 0 to 7.

The prescaling operation is not applied when the alternate function input pin is selected as the timer clock source.

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**

00 = Count Modulo: Timer Counts up to Reload Register value. Then it is reset to 0000H and counting up resumes.

01 = Reserved.

10 = Count up/down: Timer Counts up to Reload and then counts down to 0000H.

The count up and count down cycle continues.

11 = Reserved.

## Multi-Channel Timer Channel Status 0 and Status 1 Registers

Multi-Channel Timer Channel Status 0 and Status 1 Registers (MCTCHS0, MCTCHS1) indicate channel overrun and channel capture/compare event.

**Table 75. Multi-Channel Timer Channel Status 0 Register (MCTCHS0)**

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				CHDEO	CHCEO	CHBEO	CHAE0
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
ADDR	01H in Sub-Address Register, accessible through SubRegister 0							

### CHyEO—Channel y Event Flag Overrun

This bit indicates that an overrun error has occurred. An overrun occurs when a new Capture/Compare event occurs before the previous CHyEF bit is cleared. Clearing the associated CHyEF bit in the MCTCHS1 register clears this bit. This bit is cleared when TEN=0 (TEN is the MSB of MCTCTL1).

0 = No Overrun.

1 = Capture/Compare Event Flag Overrun.

**Table 76. Multi-Channel Timer Channel Status 1 Register (MCTCHS1)**

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				CHDEF	CHCEF	CHBEF	CHAEF
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W1C	R/W1C	R/W1C	R/W1C
ADDR	01H in Sub-Address Register, accessible through SubRegister 1							

### CHyEF—Channel y Event Flag

This bit indicates if a Capture/Compare event occurred for this channel. Software can use this bit to determine the channel(s) responsible for generating the MCT channel interrupt. This event flag is cleared by writing a 1 to the bit. These bits will be set when an event occurs independent of the setting of the CHIEN bit. This bit is cleared when TEN=0 (TEN is the MSB of MCTCTL1).

0 = No Capture/Compare Event occurred for this channel.

1 = A Capture/Compare Event occurred for this channel.

## Multi-Channel Timer Channel-y Control Registers

Each channel has a control register to enable the channel, select the input/output polarity, enable channel interrupts and select the channel mode of operation.

**Table 77. Multi-Channel Timer Channel Control Register (MCTCHyCTL)**

BITS	7	6	5	4	3	2	1	0
FIELD	CHEN	CHPOL	CHIEN	CHUE	Reserved	CHOP		
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
ADDR	02H, 03H, 04H, 05H in Sub-Address Register, accessible through SubRegister 2							

**y = A, B, C, D.**

**CHEN—Channel Enable.**

0 = Channel is disabled.

1 = Channel is enabled.

**CHPOL—Channel Input/Output Polarity**

Operation of this bit is a function of the current operating method of the channel.

### One-Shot Operation

When the channel is disabled, the Channel Output signal is set to the value of this bit.

When the channel is enabled, the Channel Output signal toggles for one system clock on reaching the Channel Capture/Compare Register value.

### Continuous Compare Operation

When the channel is disabled, the Channel Output signal is set to the value of this bit.

When the channel is enabled, the Channel Output signal toggles (from Low to High or High to Low) on reaching the Channel Capture/Compare Register value.

### PWM Output Operation

0 = Channel Output is forced Low when the channel is disabled. When enabled, the Channel Output is forced High on Channel Capture/Compare Register value match and forced Low on reaching the Timer Reload Register value (modulo mode) or counting down through the channel Capture/Compare register value (count up/down mode).

1 = Channel Output is forced Low when the channel is disabled. When enabled, the Channel Output is forced High on Channel Capture/Compare Register value match and forced Low on reaching the Timer Reload Register value (modulo mode) or counting down through the channel Capture/Compare register value (count up/down mode).

## Capture Operation

0 = Count is captured on the rising edge of the Channel Input signal.  
1 = Count is captured on the falling edge of the Channel Input signal.

### CHIEN—Channel Interrupt Enable

This bit enables generation of channel interrupt. A channel interrupt is generated whenever there is a capture/compare event on the Timer Channel.

0 = Channel interrupt is disabled.  
1 = Channel interrupt is enabled.

### CHUE—Channel Update Enable

This bit determines whether writes to the Channel High and Low Byte registers are buffered when TEN = 1. Writes to these registers are not buffered when TEN = 0 regardless of the value of this bit.

0 = Writes to the Channel High and Low Byte registers are buffered when TEN = 1 and only take affect on the next end of cycle count.  
1 = Writes to the Channel High and Low Byte registers are not buffered when TEN = 1.

### CHOP—Channel Operation method

This field determines the operating mode of the channel. For a detailed description of the operating modes, see [Count Up/Down Mode](#) on page 121.

000 = One-Shot Compare operation.  
001 = Continuous Compare operation.  
010 = PWM Output operation.  
011 = Capture operation.  
100 – 111 = Reserved.

## Multi-Channel Timer Channel-y High and Low Byte Registers

Each channel has a 16-bit capture/compare register defined here as the Channel-y High and Low Byte registers. When the timer is enabled, writes to these registers are buffered and loading of the registers is delayed till the next timer end count, unless CHUE = 1.

**Table 78. Multi-Channel Timer Channel-y High Byte Registers (MCTCHyH)**

BITS	7	6	5	4	3	2	1	0
FIELD	CHyH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	02H, 03H, 04H, 05H in Sub-Address Register, accessible through SubRegister 0							

**Table 79. Multi-Channel Timer Channel-y Low Byte Registers (MCTCHyL)**

<b>BITS</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>FIELD</b>	CHyL							
<b>RESET</b>	0	0	0	0	0	0	0	0
<b>R/W</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>ADDR</b>	02H, 03H, 04H, 05H in Sub-Address Register, accessible through SubRegister 1							

**y = A, B, C, D**

**CHyH and CHyL—Multi-Channel Timer Channel-y High and Low Bytes.**

During a compare operation, these two bytes, {CHyH[7:0], CHyL[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the Channel Output changes state. The Channel Output value is set by the TPOL bit in the Channel-y Control subregister.

During a capture operation, the current Timer Count is recorded in these two bytes when the desired Channel Input transition occurs.



# Watchdog Timer

Watchdog Timer (WDT) helps protect against corrupted, or unreliable software and other system-level problems that may place the Z8 Encore! XP F1680 Series into unsuitable operating states. The WDT includes the following features:

- On-chip RC oscillator
- A selectable timeout response: Reset or System Exception
- 16-bit programmable timeout value

## Operation

The WDT is a retriggerable one-shot timer that resets or interrupts the Z8 Encore! XP F1680 Series when the WDT reaches its terminal count. The WDT uses its own dedicated on-chip RC oscillator as its clock source. The WDT has only two modes of operation—ON and OFF. Once enabled, it always counts and must be refreshed to prevent a timeout. An enable can be 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.

To minimize power consumption, the RC oscillator can be disabled. The RC oscillator is disabled by clearing the `WDTEN` bit in the [Table 172](#) on page 306. If the RC oscillator is disabled, the WDT will not operate.

The WDT is a 16-bit reloadable downcounter that uses two 8-bit registers in the eZ8 CPU register space to set the reload value. The nominal WDT timeout period is calculated by the following equation:

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

where the WDT reload value is given by  $\{\text{WDTH}[7:0], \text{WDTL}[7:0]\}$  and the typical Watchdog Timer RC oscillator frequency is 10 kHz. You must consider system requirements when selecting the timeout delay. [Table 80](#) provides information on approximate timeout delays for the default and maximum WDT reload values.

**Table 80. Watchdog Timer Approximate Timeout Delays**

WDT Reload Value (Hex)	WDT Reload Value (Decimal)	Approximate Timeout Delay (with 10 kHz typical WDT Oscillator Frequency)	
		Typical	Description
0400	1024	102 ms	Reset default value timeout delay.
FFFF	65,536	6.55 s	Maximum timeout 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 0000H 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! is operating in DEBUG Mode (through the OCD), the WDT is continuously refreshed to prevent unnecessary WDT timeouts.

## Watchdog Timer Timeout Response

The WDT times out when the counter reaches 0000H. A timeout of the WDT generates either a system exception or a Reset. The WDT\_RES Option Bit determines the timeout response of the WDT. For information on programming of the WDT\_RES Option Bit, see [Flash Option Bits](#) on page 267.

### WDT System Exception in Normal Operation

If configured to generate a system exception when a timeout occurs, the WDT issues an exception request to the interrupt controller. The eZ8 CPU responds to the request by fetching the System Exception vector and executing code from the vector address. After timeout and system exception generation, the WDT is reloaded automatically and continues counting.

### WDT System Exception in STOP Mode

The WDT automatically initiates a Stop Mode Recovery and generates a system exception request if configured to generate a system exception when a timeout occurs and the Z8 Encore! XP F1680 Series is in STOP mode. Both WDT status bit and the STOP bit in the Reset Status register are set to 1 following WDT timeout in STOP mode.

Following completion of the Stop Mode Recovery the eZ8 CPU responds to the system exception request by fetching the System Exception vector and executing code from the vector address.

### WDT Reset in Normal Operation

The WDT forces the device into the Reset state if configured to generate a Reset when a timeout occurs. The WDT status bit is set to 1, see [Reset Status Register](#) on page 42. For more information on Reset and the WDT status bit, see [Reset, Stop Mode Recovery, and Low-Voltage Detection](#) on page 33. Following a Reset sequence, the WDT Counter is initialized with its reset value.

### WDT Reset in STOP Mode

If enabled in STOP mode and configured to generate a Reset when a timeout 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 [Table 11](#) on page 42 are set to 1 following WDT timeout in STOP mode. For more information, see [Reset, Stop Mode Recovery, and Low-Voltage Detection](#) on page 33.

### Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watchdog Timer Reload High (WDTH) register address unlocks the two Watchdog Timer Reload registers (WDTH and WDTL) to allow changes to the timeout period. These write operations to the WDTH register address produce no effect on the bits in the WDTH register. The locking mechanism prevents unwarranted writes to the Reload registers. The following sequence is required to unlock the Watchdog Timer Reload registers (WDTH and WDTL) for write access.

1. Write 55H to the Watchdog Timer Reload High register (WDTH).
2. Write AAH to the Watchdog Timer Reload High register (WDTH).
3. Write the desired value to the Watchdog Timer Reload High register (WDTH).
4. Write the desired value to the Watchdog Timer Reload Low register (WDTL).

All steps of the WDT Reload Unlock sequence must be written in the order as listed above. The value in the WDT Reload registers is loaded into the counter every time a WDT instruction is executed.

## Watchdog Timer Register Definitions

The two Watchdog Timer Reload registers (WDTH and WDTL) are described in the following tables.

### Watchdog Timer Reload High and Low Byte Registers

The Watchdog Timer Reload High and Low Byte (WDTH, WDTL) registers (see [Table 81](#) and [Table 82](#) on page 140) form the 16-bit reload value that is loaded into the Watchdog Timer when a WDT instruction executes. The 16-bit reload value is {WDTH[7:0], WDTL[7:0]}. Writing to these registers following the unlock sequence sets the desired Reload Value. Reading from these registers returns the current WDT count value.

**Table 81. Watchdog Timer Reload High Byte Register (WDTH = FF2H)**

BITS	7	6	5	4	3	2	1	0
FIELD	WDTH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF2H							

**Table 82. Watchdog Timer Reload Low Byte Register (WDTL = FF3H)**

BITS	7	6	5	4	3	2	1	0
FIELD	WDTL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF3H							

WDTH and WDTL—Watchdog Timer Reload High and Low Bytes

WDTH—The WDT Reload High Byte is the most significant byte, or bits [15:8] of the 16-bit WDT reload value.

WDTL—The WDT Reload Low Byte is the least significant byte, or bits [7:0] of the 16-bit WDT reload value.

# LIN-UART

The Local Interconnect Network Universal Asynchronous Receiver/Transmitter (LIN-UART) is a full-duplex communication channel capable of handling asynchronous data transfers in standard UART applications and providing LIN protocol support. The LIN-UART is a superset of the standard Z8 Encore!<sup>®</sup> UART, providing all its standard features, LIN protocol support, and a digital noise filter.

LIN-UART includes the following features:

- 8-bit asynchronous data transfer.
- Selectable even- and odd-parity generation and checking.
- Option of 1 or 2 stop bits.
- Selectable MULTIPROCESSOR (9-bit) mode with three configurable interrupt schemes.
- Separate transmit and receive interrupts.
- Framing, parity, overrun and break detection.
- 16-bit baud rate generator (BRG) which may function as a general purpose timer with interrupt.
- Driver Enable output for external bus transceivers.
- LIN protocol support for both MASTER and SLAVE modes:
  - Break generation and detection.
  - Selectable Slave Autobaud.
  - Check Tx versus Rx data when sending.
- Configuring digital-noise filter on Receive Data line.

## Architecture

The LIN-UART consists of three primary functional blocks: transmitter, receiver, and baud-rate generator. The LIN-UART's transmitter and receiver function independently but use the same baud rate and data format. The basic UART operation is enhanced by the Noise Filter and IrDA blocks. [Figure 19](#) on page 142 illustrates the LIN-UART architecture.

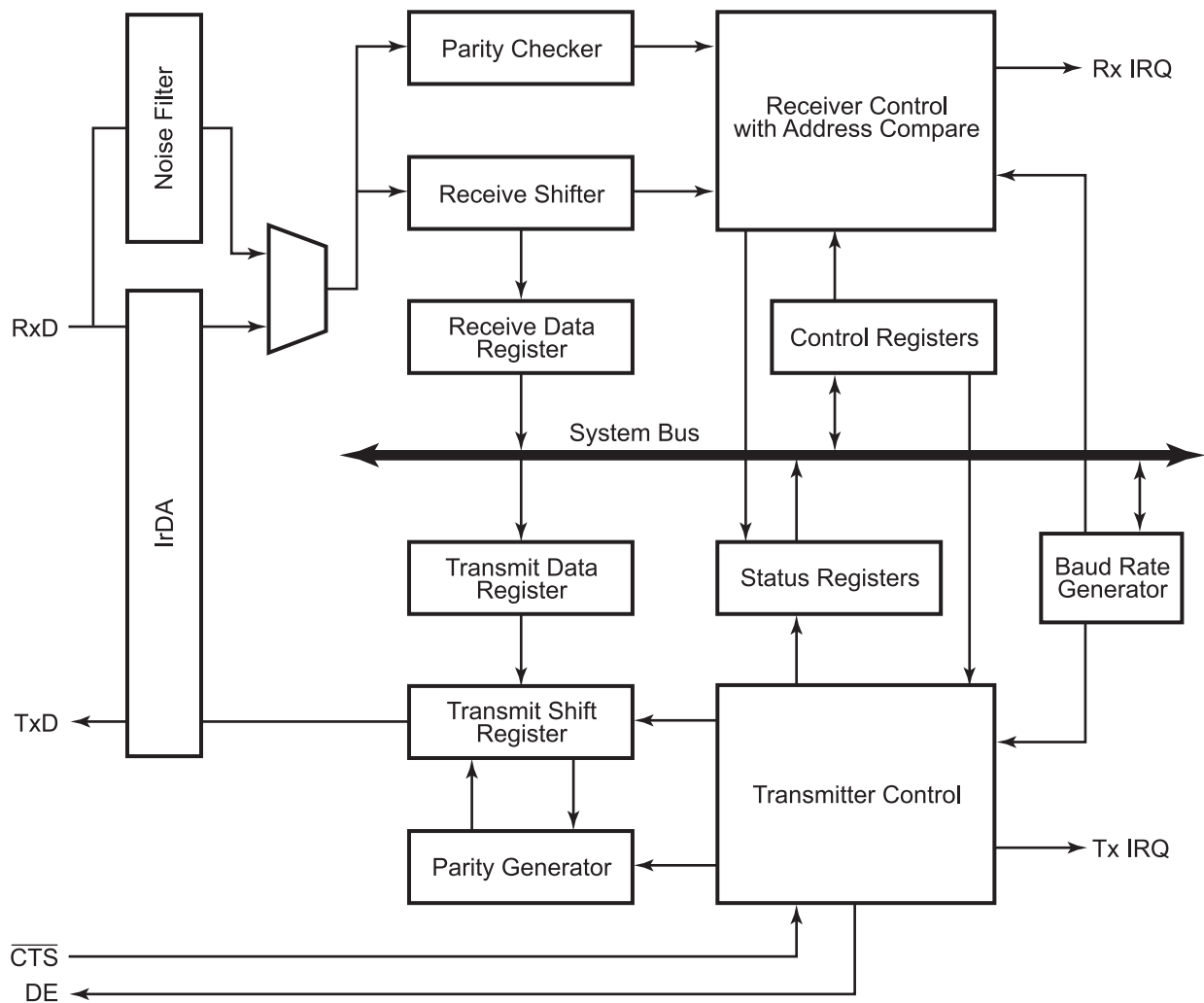


Figure 19. LIN-UART Block Diagram

## Data Format for Standard UART Modes

The LIN-UART always transmits and receives data in an 8-bit data format with the least significant bit first. An even-or-odd parity bit or multiprocessor address/data bit can be 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. [Figure 20](#) and [Figure 21](#) on page 143 illustrate the asynchronous data format employed by the LIN-UART without parity and with parity, respectively.

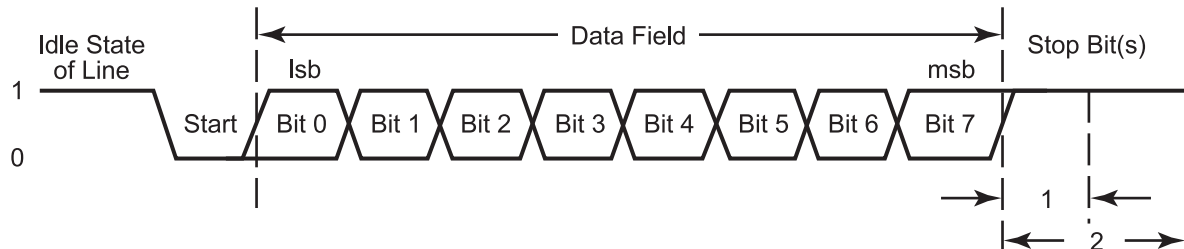


Figure 20. LIN-UART Asynchronous Data Format without Parity

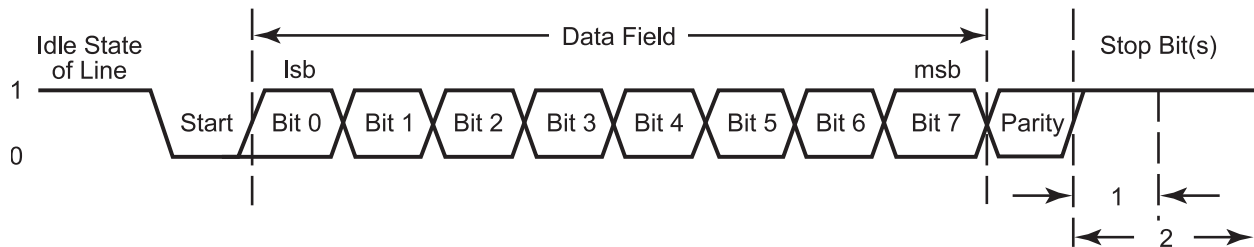


Figure 21. LIN-UART Asynchronous Data Format with Parity

## Transmitting Data Using Polled Method

Follow the steps below to transmit data using the polled-operating method:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate-function operation.
3. If MULTIPROCESSOR mode is appropriate, write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-bit) mode functions.
4. Set the MULTIPROCESSOR Mode Select ( $MPEN$ ) to enable MULTIPROCESSOR mode.
5. Write to the LIN-UART Control 0 Register to:
  - (a) Set the transmit enable bit ( $TEN$ ) to enable the LIN-UART for data transmission.
  - (b) If parity is appropriate and MULTIPROCESSOR mode is not enabled, set the parity enable bit ( $PEN$ ) and select either even-or-odd parity ( $PSEL$ ).
  - (c) Set or clear the  $CTSE$  bit to enable or disable control from the remote receiver using the  $\overline{CTS}$  pin.

6. Check the TDRE bit in the LIN-UART Status 0 register to determine if the Transmit Data Register is empty (indicated by a 1). If empty, continue to [Step 7](#). If the Transmit Data Register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data Register becomes available to receive new data.
7. If in MULTIPROCESSOR mode, write the LIN-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.
8. Write the data byte to the LIN-UART Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift register and transmits the data.
9. If appropriate and if MULTIPROCESSOR mode is enabled, make any changes to the Multiprocessor Bit Transmitter (MPBT) value.
10. To transmit additional bytes, return to [Step 5](#).

## Transmitting Data Using Interrupt-Driven Method

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

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-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 LIN-UART Transmitter interrupt and set the appropriate priority.
5. If MULTIPROCESSOR mode is appropriate, write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-bit) mode functions.
6. Set the MULTIPROCESSOR Mode Select (MPEN) to enable MULTIPROCESSOR mode.
7. Write to the LIN-UART Control 0 Register to:
  - (a) Set the transmit enable bit (TEN) to enable the LIN-UART for data transmission
  - (b) If MULTIPROCESSOR mode is not enabled, then enable parity if appropriate, and select either even or odd parity.
  - (c) Set or clear the CTSE bit to enable or disable control from the remote receiver via the CTS pin.
8. Execute an EI instruction to enable interrupts.

The LIN-UART is now configured for interrupt-driven data transmission. Because the LIN-UART Transmit Data Register is empty, an interrupt is generated immediately. When the LIN-UART Transmit interrupt is detected and there is transmit data ready to send, the associated interrupt service routine (ISR) performs the following:

1. If in MULTIPROCESSOR mode, write the LIN-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 LIN-UART Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift register and transmits the data.
3. Execute the IRET instruction to return from the interrupt-service routine and wait for the Transmit Data Register to again become empty.

If a transmit interrupt occurs and there is no transmit data ready to send the interrupt-service routine will execute the IRET instruction. When the application does have data to transmit, software can set the appropriate interrupt request bit in the Interrupt Controller to initiate a new transmit interrupt. Another alternative would be for the software to write the data to the Transmit Data Register instead of invoking the interrupt-service routine.

## Receiving Data Using Polled Method

Follow the steps below to configure the LIN-UART for polled data reception:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. If MULTIPROCESSOR mode is appropriate, write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-bit) mode functions.
4. Write to the LIN-UART Control 0 Register to:
  - (a) Set the Receive Enable Bit (REN) to enable the LIN-UART for data reception.
  - (b) If MULTIPROCESSOR mode is not enabled then enable parity, if appropriate, and select either even or odd parity.
5. Check the RDA bit in the LIN-UART Status 0 register to determine if the Receive Data Register contains a valid data byte (indicated by a 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 LIN-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 the Interrupt-Driven Method**

The LIN-UART Receiver interrupt indicates the availability of new data (as well as error conditions). Follow the steps below to configure the LIN-UART receiver for interrupt-driven operation:

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

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

1. Checks the LIN-UART Status 0 register to determine the source of the interrupt-error, break, or received data.
2. If the interrupt is due to data available, read the data from the LIN-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. Execute the `IRET` instruction to return from the ISR and await more data.

## Clear To Send Operation

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

## External Driver Enable

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

Driver Enable is a programmable polarity signal which envelopes the entire transmitted data frame including parity and stop bits as illustrated in Figure 22. The Driver Enable signal asserts when a byte is written to the LIN-UART Transmit Data Register. The Driver Enable signal asserts at least one bit period and no greater than two bit periods before the start bit is transmitted. This allows a set-up time to enable the transceiver. The Driver Enable signal deasserts one system clock period after the last stop bit is transmitted. This 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  $\text{DEPOL}$  bit in the LIN-UART Control Register 1 sets the polarity of the Driver Enable signal.

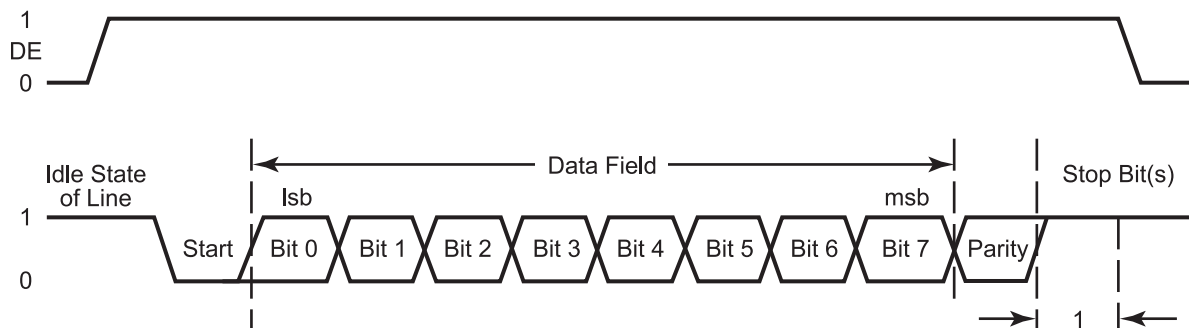


Figure 22. LIN-UART Driver Enable Signal Timing with One Stop Bit and Parity

The Driver Enable to **START** bit set-up time is calculated as follows:

$$\frac{1}{\text{Baud Rate (Hz)}} \leq \text{DE to Start Bit Set-up Time(s)} \leq \frac{2}{\text{Baud Rate (Hz)}}$$

## LIN-UART Special Modes

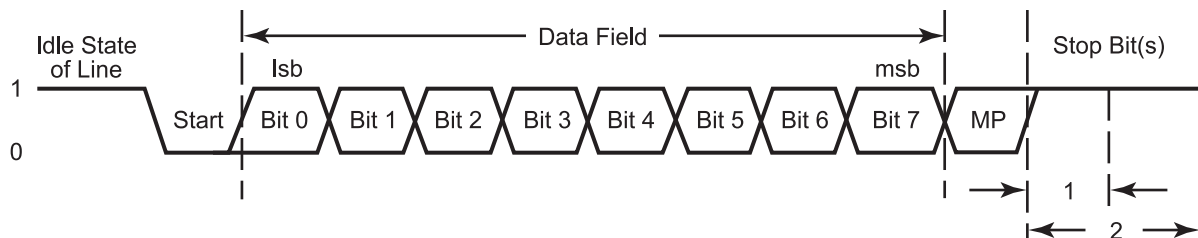
The special modes of the LIN-UART are:

- MULTIPROCESSOR Mode
- LIN Mode

The LIN-UART features a common control register (Control 0) that has a unique register address and several mode-specific control registers (Multiprocessor Control, Noise Filter Control, and LIN Control) that share a common register address (Control 1). When the Control 1 address is read or written, the **MSEL**[2:0] (Mode Select) field of the Mode Select and Status Register determines which physical register is accessed. Similarly, there are mode-specific status registers, one of which is returned when the Status 0 Register is read depending on the **MSEL** field.

## MULTIPROCESSOR Mode

The LIN-UART features a MULTIPROCESSOR (9-bit) mode that uses an extra (9<sup>th</sup>) bit for selective communication when a number of processors share a common UART bus. In MULTIPROCESSOR mode (also referred to as 9-bit mode), the multiprocessor bit (**MP**) is transmitted immediately following the 8 bits of data and immediately preceding the **STOP** bit(s) as illustrated in [Figure 23](#).



**Figure 23. LIN-UART Asynchronous MULTIPROCESSOR Mode Data Format**

In MULTIPROCESSOR (9-bit) mode, the Parity bit location (9<sup>th</sup> bit) becomes the Multiprocessor control bit. The LIN-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 LIN-UART Address Compare register holds the network address of the device.

## MULTIPROCESSOR Mode Receive Interrupts

When MULTIPROCESSOR (9-bit) mode is enabled, the LIN-UART processes only frames addressed to it. To determine whether a frame of data is addressed to the LIN-UART can be made in hardware, software or a 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 LIN-UART when it receives data directed to other devices on the multinode network. The following three MULTIPROCESSOR modes are available in hardware:

1. Interrupt on all address bytes.
2. Interrupt on matched address bytes and correctly framed data bytes.
3. Interrupt only on correctly framed data bytes.

These modes are selected with `MPMD[1:0]` in the LIN-UART Control 1 Register. For all MULTIPROCESSOR modes, bit `MPEN` of the LIN-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 interrupt service routine checks the address byte which triggered the interrupt. If it matches the LIN-UART address, the software clears `MPMD[0]`. At this point, each new incoming byte interrupts the CPU. The software determines the end of the frame and checks for it by reading the `MPRX` bit of the LIN-UART Status 1 Register for each incoming byte. If `MPRX=1`, a new frame begins. If the address of this new frame is different from the LIN-UART's address, then `MPMD[0]` must be set to 1 by software, causing the LIN-UART interrupts to go inactive until the next address byte. If the new frame's address matches the LIN-UART's, then the data in the new frame is also processed.

The second scheme is enabled by setting `MPMD[1:0]` to `10B` and writing the LIN-UART's address into the LIN-UART Address Compare Register. This mode introduces more hardware control, interrupting only on frames that match the LIN-UART's address. When an incoming address byte does not match the LIN-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 has `NEWFRM=1` in the LIN-UART Status 1 Register. When the next address byte occurs, the hardware compares it to the LIN-UART's address. If there is a match, the interrupt occurs and the `NEWFRM` bit is set to the first byte of the new frame. If there is no match, the LIN-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 LIN-UART's address into the LIN-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 remains accompanied by a `NEWFRM` assertion.

## LIN Protocol Mode

The Local Interconnect Network (LIN) protocol as supported by the LIN-UART module is defined in rev 2.0 of the LIN Specification Package. The LIN protocol specification covers all aspects of transferring information between LIN Master and Slave devices using *message frames* including error detection and recovery, SLEEP mode and wake up from SLEEP mode. The LIN-UART hardware in LIN mode provides character transfers to support the LIN protocol including Break transmission and detection, Wake-up transmission and detection, and Slave autobauding. Part of the error detection of the LIN protocol is for both master and slave devices to monitor their receive data when transmitting. If the receive and transmit data streams do not match, the LIN-UART asserts the PLE bit (physical layer error bit in Status0 register). The *message frame* timeout aspect of the protocol depends on software requiring the use of an additional general purpose timer. The LIN mode of the LIN-UART does not provide any hardware support for computing/verifying the checksum field or verifying the contents of the identifier field. These fields are treated as data and are not interpreted by hardware. The checksum calculation/verification can easily be implemented in software via the ADC (Add with Carry) instruction.

The LIN bus contains a single Master and one or more Slaves. The LIN master is responsible for transmitting the message frame header which consists of the Break, Synch and Identifier fields. Either the master or one of the slaves transmits the associated *response* section of the message which consists of data characters followed by a checksum character.

In LIN mode, the interrupts defined for normal UART operation still apply with the following changes:

- Parity Error (PE bit in Status0 register) is redefined as the Physical Layer Error (PLE) bit. The PLE bit indicates that receive data does not match transmit data when the LIN-UART is transmitting. This applies to both Master and Slave operating modes.
- The Break Detect interrupt (BRKD bit in Status0 register) indicates when a Break is detected by the slave (break condition for at least 11 bit times). Software can use this interrupt to start a timer checking for message frame timeout. The duration of the break can be read in the RxBreakLength[3:0] field of the Mode Select and Status register.
- The Break Detect interrupt (BRKD bit in Status0 register) indicates when a Wake-up message has been received, if the LIN-UART is in Lin Sleep state.
- In LIN SLAVE mode, if the BRG counter overflows while measuring the autobaud period (Start bit to beginning of bit 7 of autobaud character) an Overrun Error is indicated (OE bit in the Status0 register). In this case, software sets the LinState field back to 10b, where the slave ignores the current message and waits for the next break. The Baud Reload High and Low registers are not updated by hardware if this autobaud error occurs. The OE bit is also set if a data overrun error occurs.

## LIN System Clock Requirements

The LIN Master provides the timing reference for the LIN network and is required to have a clock source with a tolerance of  $\pm 0.5\%$ . A slave with autobaud capability is required to have a baud clock matching the master oscillator within  $\pm 14\%$ . The slave nodes autobaud to lock onto the master timing reference with an accuracy of  $\pm 2\%$ . If a slave does not contain autobaud capability it must include a baud clock which deviates from the masters by not more than  $\pm 1.5\%$ . These accuracy requirements must include the effects such as voltage and temperature drift during operation.

Before sending/receiving messages, the Baud Reload High/Low registers must be initialized. Unlike standard UART modes, the Baud Reload High/Low registers must be loaded with the baud interval rather than 1/16 of the baud interval.

In order to autobaud with the required accuracy, the LIN Slave system clock must be at least 100 times the baud rate.

## LIN Mode Initialization and Operation

The LIN protocol mode is selected by setting either the `LMST` (LIN Master) or `LSLV` (LIN Slave), and optionally (for LIN slave) the `ABEN` (Autobaud Enable) bits in the LIN Control Register. To access the LIN Control Register, the `MSEL` (Mode Select) field of the LIN-UART Mode Select/Status register must be = 010B. The LIN-UART Control0 register must be initialized with `TEN` = 1, `REN` = 1, and all other bits = 0.

In addition to the `LMST`, `LSLV`, and `ABEN` bits in the LIN Control Register, a `LinState[1:0]` field exists which defines the current state of the LIN logic. This field is initially set by software. In the LIN SLAVE mode, the `LinState` field is updated by hardware as the slave moves through the Wait For Break, AutoBaud, and Active states.

## LIN MASTER Mode Operation

LIN MASTER mode is selected by setting `LMST` = 1, `LSLV` = 0, `ABEN` = 0, and `LinState[1:0]` = 11B. If the LIN bus protocol indicates the bus is required go into the *LIN sleep* state, the `LinState[1:0]` bits must be set to 00B by software.

The Break is the first part of the message frame transmitted by the master, consisting of at least 13 bit periods of logical zero on the LIN bus. During initialization of the LIN master, the duration (in bit times) of the Break is written to the `TxBreakLength` field of the LIN Control Register. The transmission of the Break is performed by setting the `SBRK` bit in the Control 0 Register. The LIN-UART starts the Break once the `SBRK` bit is set and any character transmission currently underway has completed. The `SBRK` bit is deasserted by hardware till the break is completed.

If it is necessary to generate a Break longer than 15 bit times, the `SBRK` bit can be used in normal UART mode where software times the duration of the Break.

The Synch character is transmitted by writing a 55H to the Transmit Data Register (TDRE must = 1 before writing). The Synch character is not transmitted by the hardware till the Break is complete.

The Identifier character is transmitted by writing the appropriate value to the Transmit Data Register (TDRE must = 1 before writing).

If the master is sending the *response* portion of the message, these data and checksum characters are written to the Transmit Data Register when the TDRE bit asserts. If the transmit data register is written after TDRE asserts, but before TXE asserts, the hardware inserts one or two stop bits between each character as determined by the Stop bit in the Control 0 Register. Additional idle time occurs between characters, if TXE asserts before the next character is written.

If the selected slave is sending the response portion of the frame to the master, each receive byte will be signalled by the receive data interrupt (RDA bit will be set in the Status0 register). If the selected slave is sending the response to a different slave, the master can ignore the response characters by deasserting the REN bit in the Control 0 Register until the frame time slot is completed.

### LIN SLEEP Mode

While the LIN bus is in the *sleep* state, the CPU can either be in low power STOP mode, in HALT mode, or in normal operational state. Any device on the LIN bus may issue a Wake-up message if it requires the master to initiate a LIN message frame. Following the Wake-up message, the master wakes up and initiates a new message. A Wake-up message is accomplished by pulling the bus Low for at least 250  $\mu$ s but less than 5 ms. Transmitting a 00H character is one way to transmit the Wake-up message.

If the CPU is in STOP mode, the LIN-UART is not active and the Wake-up message must be detected by a GPIO edge detect Stop Mode Recovery. The duration of the Stop Mode Recovery sequence may preclude making an accurate measurement of the Wake-up message duration.

If the CPU is in HALT or OPERATIONAL mode, the LIN-UART (if enabled) times the duration of the Wake-up and provides an interrupt following the end of the break sequence if the duration is  $\geq 3$  bit times. The total duration of the Wake-up message in bit times may be obtained by reading the RxBreakLength field in the Mode Select and Status register. After a Wake-up message has been detected, the LIN-UART can be placed (by software) either into LIN Master or LIN Slave Wait for Break states as appropriate. If the break duration exceeds 15 bit times, the RxBreakLength field contains the value Fh. If the LIN-UART is disabled, Wake-up message is detected via a port pin interrupt and timed by software. If the device is in STOP mode, the High to Low transition on the port pin will bring the device out of STOP mode.

The LIN Sleep state is selected by software setting LinState[1:0] = 00. The decision to move from an active state to sleep state is based on the LIN messages as interpreted by software.

## LIN Slave Operation

LIN SLAVE mode is selected by setting  $LMST = 0$ ,  $LSLV = 1$ ,  $ABEN = 1$  or  $0$  and  $LinState[1:0] = 01b$  (Wait for Break State). The LIN slave detects the start of a new message by the Break which appears to the Slave as a break of at least 11 bit times in duration. The LIN-UART detects the Break and generates an interrupt to the CPU. The duration of the Break is observable in the  $RxBreakLength$  field of the Mode Select and Status register. A Break of less than 11 bit times in duration does not generate a break interrupt when the LIN-UART is in Wait for Break state. If the Break duration exceeds 15 bit times, the  $RxBreakLength$  field contains the value  $Fh$ .

Following the Break, the LIN-UART hardware automatically transits to the *Autobaud* state, where it autobauds by timing the duration of the first 8 bit times of the Synch character as defined in the LIN standard. The duration of the autobaud period is measured by the BRG Counter which will update every 8th system clock cycle between the start bit and the beginning of bit 7 of the autobaud sequence. At the end of the autobaud period, the duration measured by the BRG counter (auto baud period divided by 8) is automatically transferred to the Baud Reload High and Low registers if the  $ABEN$  bit of the LIN control register is set. If the BRG Counter overflows before reaching the start of bit 7 in the autobaud sequence the Autobaud Overrun Error interrupt occurs, the  $OE$  bit in the Status0 register is set, and the Baud Reload registers are not updated. To autobaud within 2% of the master's baud rate, the slave system clock must be a minimum of 100 times the baud rate. To avoid an autobaud overrun error, the system clock must not be greater than  $2^{19}$  times the baud rate (16 bit counter following 3-bit prescaler when counting the 8 bit times of the Autobaud sequence).

Following the Synch character, the LIN-UART hardware transits to the Active state where the identifier character is received and the characters of the Response section of the message are sent or received. The slave remains in the active state until a Break is received or software forces a state change. Once in Active State (autobaud has completed), a Break of 10 or more bit times is recognized and will cause a transition to the Autobaud state.

If the Identifier character indicates that this slave device is not participating in the message, software sets the  $LinState[1:0] = 01b$  (Wait for Break State) to ignore the rest of the message. No further receive interrupts will occur until the next Break.

## LIN-UART Interrupts

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

### Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register Empty bit ( $TDRE$ ) is set to 1. This indicates that the transmitter is ready to accept new data for transmission. The  $TDRE$  interrupt occurs when the transmitter is initially enabled and

after the Transmit Shift Register has shifted out the first bit of a character. At this point, the Transmit Data Register may 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 LIN-UART Transmit Data Register clears the `TDRE` bit to 0.

### Receiver Interrupts

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

- A data byte has been received and is available in the LIN-UART Receive Data Register. This interrupt can be disabled independent of the other receiver interrupt sources via the `RDAIRQ` bit (this feature is useful in devices which support DMA). The received data interrupt occurs once the receive character has been 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.

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

- A Break is received.
- A Receive Data Overrun or LIN Slave Autobaud Overrun Error is detected.
- A Data Framing Error is detected.
- A Parity Error is detected (physical layer error in LIN mode).

### LIN-UART Overrun Errors

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

After the valid data has been read, the `OE` bit of the Status 0 register is updated to indicate the overrun condition (and Break Detect, if applicable). The `RDA` bit is set to 1 to indicate that the Receive Data Register contains a data byte. However, because the overrun error occurred, this byte may not contain valid data and must be ignored. The `BRKD` bit indicates if the overrun is 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 LIN-UART Status 0 register.

In LIN mode, an Overrun Error is signalled for receive-data overruns as described above and in the LIN Slave if the BRG Counter overflows during the autobaud sequence (the `ATB` bit will also be set in this case). There is no data associated with the autobaud overflow interrupt; however the Receive Data Register must be read to clear the `OE` bit. In this case, software must write a 10B to the `LinState` field, forcing the LIN slave back to a *Wait for Break* state.

## LIN-UART Data- and Error-Handling Procedure

Figure 24 illustrates the recommended procedure for use in LIN-UART receiver interrupt service routines.

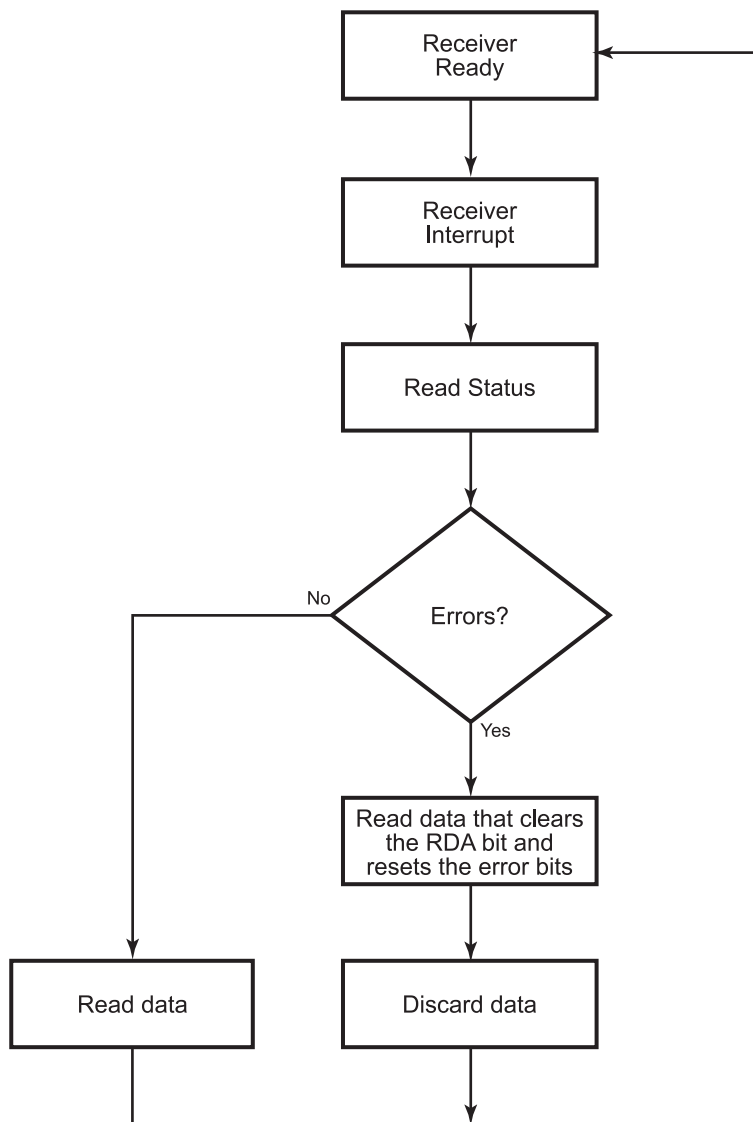


Figure 24. LIN-UART Receiver Interrupt Service Routine Flow

### Baud Rate Generator Interrupts

If the `BRGCTL` bit of the Multiprocessor Control Register (LIN-UART Control 1 Register with `MSEL = 000b`) register is set, and the `REN` bit of the Control 0 Register is 0. The LIN-UART Receiver interrupt asserts when the LIN-UART Baud Rate Generator reloads. This action allows the Baud Rate Generator to function as an additional counter, if the LIN-UART receiver functionality is not employed. The transmitter can be enabled in this mode.

### LIN-UART Baud Rate Generator

The LIN-UART Baud Rate Generator creates a lower frequency baud rate clock for data transmission. The input to the Baud Rate Generator is the system clock. The LIN-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 LIN-UART. The LIN-UART data rate for normal UART operation is calculated using the following equation:

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

The LIN-UART data rate for LIN mode UART operation is calculated using the following equation:

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

When the LIN-UART is disabled, the BRG functions as a basic 16-bit timer with interrupt on timeout. To configure the BRG as a timer with interrupt on timeout, follow the procedure below:

1. Disable the LIN-UART receiver by clearing the `REN` bit in the LIN-UART Control 0 Register to 0 (`TEN` bit may be asserted, transmit activity may occur).
2. Load the appropriate 16-bit count value into the LIN-UART Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and the associated interrupt by setting the `BRGCTL` bit in the LIN-UART Control 1 Register to 1.

## Noise Filter

A noise filter circuit is included which filters noise on a digital input signal (such as UART Receive Data) before the data is sampled by the block. This is likely to be a requirement for protocols with a noisy environment.

The noise filter contains the following features:

- Synchronizes the receive input data to the System Clock.
- Noise Filter Enable (NFEN) input selects whether the noise filter is bypassed (NFEN = 0) or included (NFEN = 1) in the receive data path.
- Noise Filter Control (NFCTL[2:0]) input selects the width of the up/down saturating counter digital filter. The available width ranges from 4 to 11 bits.
- The digital filter output features hysteresis.
- Provides an active low *Saturated State* output (FiltSatB) which is used as an indication of the presence of noise.

## Architecture

Figure 25 illustrates how the noise filter is integrated with the LIN-UART on a LIN network.

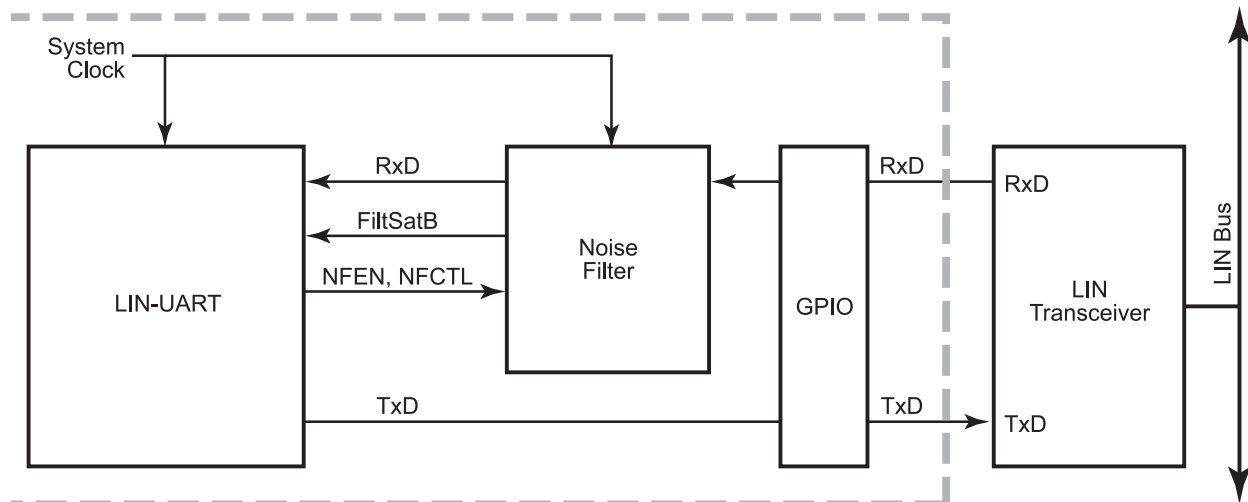


Figure 25. Noise Filter System Block Diagram

## Operation

Figure 26 on page 158 illustrates the operation of the noise filter both with and without noise. The noise filter in this example is a 2-bit up/down counter which saturates at 00b

and 11b. A 2-bit counter is shown for convenience, the operation of wider counters is similar. The output of the filter switches from 1 to 0, when the counter counts down from 01b to 00b; and switches from 0 to 1, when the counter counts up from 10b to 11b. The noise filter delays the receive data by three System Clock cycles.

The FiltSatB signal is checked when the filtered RxD is sampled in the center of the bit time. The presence of noise (FiltSatB = 1 at center of bit time) does not mean that the sampled data is incorrect, but just that the filter is not in its 'saturated' state of all 1s or all 0s. If FiltSatB = 1 then RxD is sampled during a receive character, the NE bit in the ModeStatus[4:0] field is set. By observing this bit, an indication of the level of noise in the network can be obtained.

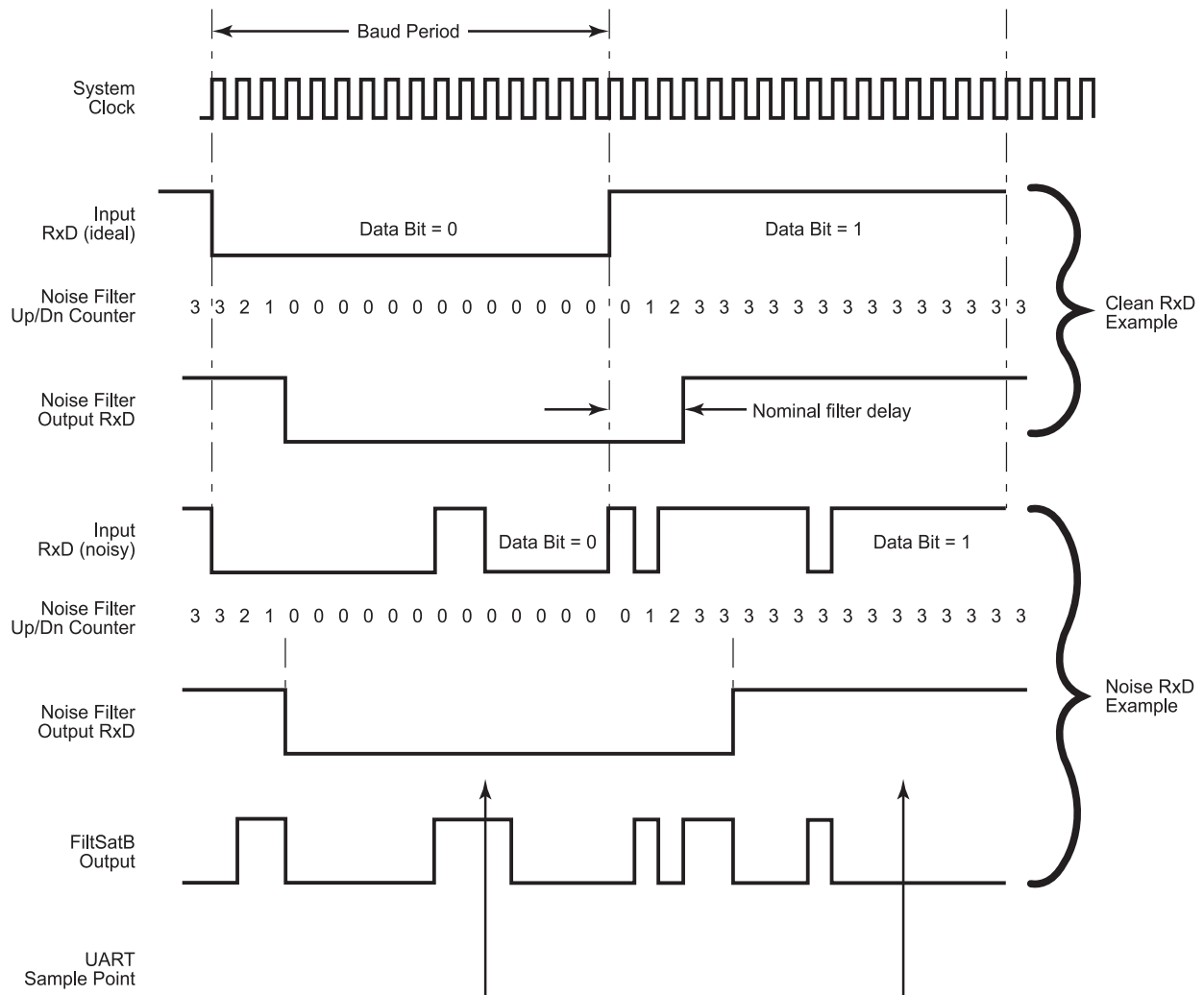


Figure 26. Noise Filter Operation

## LIN-UART Control Register Definitions

The LIN-UART control registers support the LIN-UART, the associated Infrared Encoder/Decoder, and the noise filter. For more information on the infrared operation, see [Infrared Encoder/Decoder](#) on page 177.

### LIN-UART Transmit Data Register

Data bytes written to the LIN-UART Transmit Data Register ([Table 83](#)) are shifted out on the TxD pin. The Write-only LIN-UART Transmit Data Register shares a Register File address with the Read-only LIN-UART Receive Data Register.

**Table 83. LIN-UART Transmit Data Register (U0TXD = F40H)**

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, F48H							

**Note:** W = Write; X = undefined

TxD—Transmit Data

LIN-UART transmitter data byte to be shifted out through the TxD pin.

### LIN-UART Receive Data Register

Data bytes received through the RxD pin are stored in the LIN-UART Receive Data Register as listed in [Table 84](#). The Read-only LIN-UART Receive Data Register shares a Register File address with the Write-only LIN-UART Transmit Data Register.

**Table 84. LIN-UART Receive Data Register (U0RXD = F40H)**

BITS	7	6	5	4	3	2	1	0
FIELD	RxD							
RESET	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
ADDR	F40H, F48H							

**Note:** R = Read

RxD—Receive Data

LIN-UART receiver data byte from the RxD pin

## LIN-UART Status 0 Register

The LIN-UART Status 0 Register identifies the current LIN-UART operating configuration and status. [Table 85](#) describes the Status 0 Register for standard UART mode. [Table 86](#) on page 161 describes the Status 0 Register for LIN mode. A more detailed discussion of each bit follows each table.

**Table 85. LIN-UART Status 0 Register—Standard UART Mode (U0STAT0 = F41H)**

BITS	7	6	5	4	3	2	1	0
FIELD	RDA	PE	OE	FE	BRKD	TDRE	TXE	CTS
RESET	0	0	0	0	0	1	1	X
R/W	R	R	R	R	R	R	R	R
ADDR	F41H, F49H							

**Note:** R = Read; X = undefined.

### **RDA— Receive Data Available**

0 = The LIN-UART Receive Data Register is empty.

1 = There is a byte in the LIN-UART Receive Data Register.

### **PE— Parity Error**

0 = No parity error occurred.

1 = A parity error occurred.

### **OE— Overrun Error**

0 = No overrun error occurred.

1 = An overrun error occurred.

### **FE— Framing Error**

0 = No framing error occurred.

1 = A framing error occurred.

### **BRKD— Break Detect**

0 = No break occurred.

1 = A break occurred.

### **TDRE— Transmitter Data Register Empty**

0 = Do not write to the Transmit Data Register.

1 = The Transmit Data Register is ready to receive an additional byte for transmission.

### **TXE— Transmitter Empty**

0 = Data is currently transmitting.

1 = Transmission is complete.

### **CTS— Clear to Send Signal**

**Receive Data Available (RDA)**—This bit indicates that the LIN-UART Receive Data Register has received data. Reading the LIN-UART Receive Data Register clears this bit.

**Parity Error (PE)**—This bit indicates that a parity error has occurred. Reading the Receive Data Register clears this bit.

**Overrun Error (OE)**—This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the Receive Data Register is not read. Reading the Receive Data Register clears this bit.

**Framing Error (FE)**—This bit indicates that a framing error (no STOP bit following data reception) was detected. Reading the Receive Data Register clears this bit.

**Break Detect (BRKD)**—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 Receive Data Register clears this bit.

**Transmitter Data Register Empty (TDRE)**—This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit.

**Transmitter Empty (TXE)**—This bit indicates that the Transmit Shift Register is empty and character transmission is finished.

**Clear To Send Signal ( $\overline{\text{CTS}}$ )**—When this bit is read it returns the level of the  $\overline{\text{CTS}}$  signal. If LBEN = 1, the  $\overline{\text{CTS}}$  input signal is replaced by the internal Receive Data Available signal to provide flow control in loopback mode. CTS only affects transmission if the CTSE bit = 1.

**Table 86. LIN-UART Status 0 Register—LIN Mode (U0STAT0 = F41H)**

BITS	7	6	5	4	3	2	1	0
FIELD	RDA	PLE	OE	FE	BRKD	TDRE	TXE	ATB
RESET	0	0	0	0	0	1	1	0
R/W	R	R	R	R	R	R	R	R
ADDR	F41H, F49H							

**Note:** R = Read

**RDA— Receive Data Available**

0 = The Receive Data Register is empty.

1 = There is a byte in the Receive Data Register.

**PLE— Physical Layer Error**

0 = Transmit and Receive data match.

1 = Transmit and Receive data do not match.

**OE— Receive Data and Autobaud Overrun Error**

0 = No autobaud or data overrun error occurred.

1 = An autobaud or data overrun error occurred.

**FE— Framing Error**

0 = No framing error occurred.

1 = A framing error occurred.

**BRKD— Break Detect**

0 = No LIN break occurred.

1 = LIN break occurred.

**TDRE— Transmitter Data Register Empty**

0 = Do not write to the Transmit Data Register.

1 = The Transmit Data Register is ready to receive an additional byte for transmission.

**TXE— Transmitter Empty**

0 = Data is currently transmitting.

1 = Transmission is complete.

**ATB— LIN Slave Autobaud complete.**

**Receive Data Available (RDA)**—This bit indicates that the Receive Data Register has received data. Reading the Receive Data Register clears this bit.

**Physical Layer Error (PLE)**—This bit indicates that transmit and receive data do not match when a LIN slave or master is transmitting. This could be by a fault in the physical layer or multiple devices driving the bus simultaneously. Reading the Status 0 Register or the Receive Data Register clears this bit.

**Receive Data and Autobaud Overrun Error (OE)**—This bit is set just as in normal UART operation if a receive data overrun error occurs. This bit is also set during LIN Slave autobaud if the BRG counter overflows before the end of the autobaud sequence. This indicates that the receive activity is not an autobaud character or the master baud rate is too slow. The ATB status bit will also be set in this case. This bit is cleared by reading the Receive Data Register.

**Framing Error (FE)**—This bit indicates that a framing error (no STOP bit following data reception) is detected. Reading the Receive Data Register clears this bit.

**Break Detect (BRKD)**—This bit is set in LIN mode if:

1. It is in Lin Sleep state and a break of at least 4 bit times occurred (Wake-up event) or
2. It is in Slave Wait Break state and a break of at least 11 bit times occurred (Break event) or
3. It is in Slave Active state and a break of at least 10 bit times occurs. Reading the Status 0 Register or the Receive Data Register clears this bit.

**Transmitter Data Register Empty (TDRE)**—This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit.

**Transmitter Empty (TXE)**—This bit indicates that the transmit shift register is empty and character transmission is completed.

**LIN Slave Autobaud Complete (ATB)**—This bit is set in LIN SLAVE mode when an autobaud character is received. If the ABIEN bit is set in the LIN Control Register, then a receive interrupt is generated when this bit is set. Reading the Status 0 Register clears this bit. This bit will be 0 in LIN MASTER mode.

## LIN-UART Mode Select and Status Register

The LIN-UART Mode Select and Status Register ([Table 87](#)) contains mode select and status bits. A more detailed discussion of each bit follows the table.

**Table 87. LIN-UART Mode Select and Status Register (U0MDSTAT = F44H)**

BITS	7	6	5	4	3	2	1	0
FIELD	MSEL			MODE STATUS				
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R	R	R
ADDR	F44H, F4CH							

**Note:** R = Read; R/W = Read/Write

### 7–5 Mode Select

000 = Multiprocessor and normal UART control/status

001 = Noise filter control/status

010 = LIN protocol control/status

011 = Reserved

100 = Reserved

101 = Reserved

110 = Reserved

111 = LIN-UART hardware revision (allows hardware revision to be read in the Mode Status field)

### 4–0 Mode Status

MSEL[2:0]=000, MULTIPROCESSOR mode status = {0,0,0,NEWFRM, MPRX}

MSEL[2:0]=001, Noise filter status = {NE,0,0,0,0}

MSEL[2:0]=010, LIN mode status = {NE, RxBreakLength}

MSEL[2:0]=011, Reserved; must be 00000

MSEL[2:0]=100, Reserved; must be 00000

MSEL[2:0]=101, Reserved; must be 00000

MSEL[2:0]=110, Reserved; must be 00000

MSEL[2:0]=111, LIN-UART hardware revision

**Mode Select (MSEL)**—This R/W field determines which control register is accessed when performing a Write or Read to the UART Control 1 Register address. This field also determines which status is returned in the Mode Status field when reading this register.

**Mode Status**—This read-only field returns status corresponding to one of four modes selected by MSEL. These four modes are described in [Table 88](#) on page 164.

**Table 88. Mode Status Fields**

MULTIPROCESSOR Mode Status Field	<p><b>NEWFRM</b> Status bit denoting the start of a new frame. Reading the LIN-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.</p> <hr/> <p><b>Multiprocessor Receive (MPRX)</b> Returns the value of the last multiprocessor bit received. Reading from the LIN-UART Receive Data Register resets this bit to 0.</p>
Digital Noise Filter Mode Status Field	<p><b>Noise Event (NE); MSEL = 001b</b> This bit is asserted if digital noise is detected on the receive data line when the data is sampled (center of bit-time). If this bit is set, it does not mean that the receive data is corrupted (though it may be in extreme cases), means that one or more of the noise filter data samples near the center of the bit-time did not match the average data value.</p>
LIN Mode Status Field	<p><b>Noise Event (NE); MSEL = 010b</b> This bit is asserted if some noise level is detected on the receive data line when the data is sampled (center of bit-time). If this bit is set, it does not indicate that the receive data is corrupt (though it may be in extreme cases), means that one or more of the 16x data samples near the center of the bit-time did not match the average data value.</p> <hr/> <p><b>RxBreakLength</b> LIN mode received break length. This field may be read following a break (LIN Wake-up or Break) so that the software can determine the measured duration of the break. If the break exceeds 15 bit times the value saturates at 1111b.</p>
Hardware Revision Mode Status Field	<p><b>Noise Event (NE); MSEL = 111b</b> This field indicates the hardware revision of the LIN-UART block. 00_xxx LIN UART hardware revision. 01_xxx Reserved. 10_xxx Reserved. 11_xxx Reserved.</p>

## LIN-UART Control 0 Register

The LIN-UART Control 0 Register (Table 89) configures the basic properties of LIN-UART's transmit and receive operations. A more detailed discussion of each bit follows the table.

**Table 89. LIN-UART Control 0 Register (U0CTL0 = F42H)**

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F42H, F4AH							

Note: R/W = Read/Write.

### TEN—Transmit Enable

0 = Transmitter disabled.

1 = Transmitter enabled.

### REN—Receive Enable

0 = Receiver disabled.

1 = Receiver enabled.

### CTSE—Clear To Send Enable

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

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

### PEN—Parity Enable

0 = Parity is disabled. This bit is overridden by the MPEN bit.

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 sent as an additional parity bit for the transmitter/receiver.

1 = Odd parity is sent as an additional parity bit for the transmitter/receiver.

### SBRK—Send Break

0 = No break is sent.

1 = The output of the transmitter is 0.

### 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 within the IrDA module.

**Transmit Enable (TEN)**—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.

**Receive Enable (REN)**—This bit enables or disables the receiver.

**Clear To Send Enable (CTSE)**—See the bit descriptions in [Table 89](#) on page 165.

**Parity Enable (PEN)**—This bit enables or disables parity. Even or odd is determined by the PSEL bit.

**Parity Select (PSEL)**—See the bit descriptions in [Table 89](#) on page 165.

**Send Break (SBRK)**—This bit pauses or breaks data transmission. Sending a break interrupts any transmission in progress, so ensure that the transmitter has completed sending data before setting this bit. In standard UART mode, the duration of the break is determined by how long the software leaves this bit asserted. Also the duration of any required STOP bits following the break must be timed by software before writing a new byte to be transmitted to the Transmit Data Register. In LIN mode, the master sends a Break character by asserting SBRK. The duration of the break is timed by hardware and the SBRK bit is deasserted by hardware when the Break is completed. The duration of the Break is determined by the TxBreakLength field of the LIN Control Register. One or two STOP bits are automatically provided by the hardware in LIN mode as defined by the STOP bit.

**Stop Bit Select (STOP)**—See the bit descriptions in [Table 89](#) on page 165.

**Loop Back Enable (LBEN)**—See the bit descriptions in [Table 89](#) on page 165.

## LIN-UART Control 1 Registers

Multiple registers, (see [Table 90](#) through [Table 92](#)) are accessible by a single bus address. The register selected is determined by the Mode Select (MSEL) field. These registers provide additional control over LIN-UART operation.

### Multiprocessor Control Register

When MSEL = 000b, the Multiprocessor Control Register (see [Table 90](#)) provides control for UART MULTIPROCESSOR mode, IRDA mode, Baud Rate Timer mode as well as other features that may apply to multiple modes. A more detailed discussion of each bit follows the table.

**Table 90. Multiprocessor Control Register (U0CTL1 = F43H with MSEL = 000b)**

BITS	7	6	5	4	3	2	1	0
FIELD	MPMD1	MPEN	MPMD0	MPBT	DEPOL	BRGCTL	RDAIRQ	IREN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F43H, F4BH							

**Note:** R/W = Read/Write

Bit Position	Value	Description
[7,5] MPMD[1:0]	<b>MULTIPROCESSOR Mode</b>	
	00	The LIN-UART generates an interrupt request on all data and address bytes.
	01	The LIN-UART generates an interrupt request only on received address bytes.
	10	The LIN-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 LIN-UART generates an interrupt request on all received data bytes for which the most recent address byte matched the value in the Address Compare Register.
6 MPEN	<b>Multiprocessor Enable</b>	
	0	Disable MULTIPROCESSOR (9-bit) mode.
	1	Enable MULTIPROCESSOR (9-bit) mode.
4 MPBT	<b>Multiprocessor Bit Transmit</b>	
	0	Send a 0 in the multiprocessor bit location of the data stream (9th bit).
	1	Send a 1 in the multiprocessor bit location of the data stream (9th bit).
3 DEPOL	<b>Driver Enable Polarity</b>	
	0	DE signal is active High.
	1	DE signal is active Low.
2 BRGCTL	<b>Baud Rate Generator Control, when LIN-UART receiver not enabled</b>	
	0	BRG is disabled. Reads from the Baud Rate High and Low Byte registers return the BRG reload value.
	1	BRG is enabled and counting. The Baud Rate Generator generates a receive interrupt when it counts down to 0. Reads from the Baud Rate High and Low Byte registers return the current BRG count value.
	<b>Baud Rate Generator Control, when LIN-UART receiver enabled</b>	
	0	Reads from the Baud Rate High and Low Byte registers return the BRG Reload value.
	1	Reads from the Baud Rate High and Low Byte registers return the current BRG count value. Unlike the timers, there is no mechanism to latch the High Byte when the Low Byte is read.
1 RDAIRQ	<b>Stop Bit Select</b>	
	0	Received data and receiver errors generates an interrupt request to the Interrupt controller.
	1	Received data does not generate an interrupt request to the Interrupt controller. Only receiver errors generate an interrupt request.
0 IREN	<b>Loop Back Enable</b>	
	0	Infrared Encoder/Decoder is disabled. LIN-UART operates normally.
	1	Infrared Encoder/Decoder is enabled. The LIN-UART transmits and receives data through the Infrared Encoder/Decoder.

**MPMD [1:0]—MULTIPROCESSOR Mode**

- 00 = The LIN-UART generates an interrupt request on all received bytes (data and address).
- 01 = The LIN-UART generates an interrupt request on all received address bytes.
- 10 = The LIN-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 LIN-UART generates an interrupt request on all received data bytes for the most recent address byte which matches the value in the Address Compare Register.

**MPEN—Multiprocessor Enable**

- 0 = Disable MULTIPROCESSOR (9-bit) mode.
- 1 = Enable MULTIPROCESSOR (9-bit) mode.

**MULTIPROCESSOR Mode (MPMD[1:0])—MULTIPROCESSOR (9-bit) mode—bits 7 and 5.**

**Multiprocessor (9-Bit) Enable (MPEN)—**This bit is used to enable | MULTIPROCESSOR (9-bit) mode.

**Multiprocessor Bit Transmit (MPBT)—**This bit is applicable only when MULTIPROCESSOR (9-bit) mode is enabled.

**Driver Enable Polarity (DEPOL)—**See the bit descriptions in [Table 90](#) on page 166.

**Baud Rate Generator Control (BRGCTL)—**This bit causes different LIN-UART behavior depending on whether the LIN-UART receiver is enabled (REN = 1 in the LIN-UART Control 0 Register). When the LIN-UART receiver is not enabled, this bit determines whether the Baud Rate Generator issues interrupts. When the LIN-UART receiver is enabled, this bit allows Reads from the baud rate registers to return the BRG count value instead of the reload value.

**Receive Data Interrupt Enable ( $\overline{\text{RDAIRQ}}$ )—**See the bit descriptions in [Table 90](#) on page 166.

**Infrared Encoder/Decoder Enable (IREN)—**See the bit descriptions in [Table 90](#) on page 166.

## Noise Filter Control Register

When MSEL = 001b, the Noise Filter Control Register (see [Table 91](#) on page 169) provides control for the digital noise filter.

**Table 91. Noise Filter Control Register (U0CTL1 = F43H with MSEL = 001b)**

BITS	7	6	5	4	3	2	1	0
FIELD	NFEN	NFCTL			—			
RESET	0	0	0	0	0	0	0	0
CPU ACCESS	R/W	R/W	R/W	R/W	R	R	R	R
ADDR	F43H, F4BH							

**Note:** R = Read; R/W = Read/Write

Bit Position	Value	Description
7 NFEN	<b>Noise Filter Enable</b>	
	0	Noise filter is disabled.
	1	Noise filter is enabled. Receive data is preprocessed by the noise filter.
[6:4] NFCTL	<b>Noise Filter Control</b>	
	000	4-bit up/down counter.
	001	5-bit up/down counter.
	010	6-bit up/down counter.
	011	7-bit up/down counter.
	100	8-bit up/down counter.
	101	9-bit up/down counter.
	110	10-bit up/down counter.
	111	11-bit up/down counter.
[3:0] Reserved	—	Reserved; must be 0000.

**Noise Filter Enable (NFEN)**—See the bit descriptions in [Table 91](#) on page 169.

**Noise Filter Control (NFCTL)**—This field controls the delay and noise rejection characteristics of the noise filter. The wider the counter is, the more delay is introduced by the filter, and the wider the noise event is filtered.

## LIN Control Register

When  $MSEL = 010b$ , the LIN Control Register provides control for the LIN mode of operation. A more detailed discussion of each bit follows the table.

**Table 92. LIN Control Register (U0CTL1 = F43H with MSEL = 010b)**

BITS	7	6	5	4	3	2	1	0
FIELD	LMST	LSLV	ABEN	ABIEN	LinState[1:0]		TxBreakLength	
RESET	0	0	0	0	0	0	0	0
CPU ACCESS	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F43H, F4BH							

**Note:** R/W = Read/Write

Bit Position	Value	Description
7 LMST	<b>LIN MASTER Mode</b>	
	0	LIN MASTER mode not selected.
	1	LIN MASTER mode selected (if MPEN, PEN, LSLV = 0).
6 LSLV	<b>LIN SLAVE Mode</b>	
	0	LIN SLAVE mode not selected.
	1	LIN SLAVE mode selected (if MPEN, PEN, LMST = 0).
5 ABEN	<b>Autobaud Enable</b>	
	0	Autobaud not enabled.
	1	Autobaud enabled, if in LIN SLAVE mode.
4 ABIEN	<b>Autobaud Interrupt Enable</b>	
	0	Interrupt following autobaud does not occur.
	1	Interrupt following autobaud enabled, if in LIN SLAVE mode. When the autobaud character is received, a receive interrupt is generated and the ATB bit is set in the Status0 Register.
[3:2] LinState[1:0]	<b>LIN State Machine</b>	
	00	Sleep state (either LMST or LSLV can be set).
	01	Wait for Break state (only valid for LSLV = 1).
	10	Autobaud state (only valid for LSLV = 1).
	11	Active state (either LMST or LSLV can be set).

Bit Position	Value	Description
[1:0]	<b>TxBreakLength</b>	
TxBreakLength	00	13 bit times.
	01	14 bit times.
	10	15 bit times.
	11	16 bit times.

**LIN MASTER Mode (LMST)**—See the bit descriptions in [Table 92](#) on page 170.

**LIN SLAVE Mode (LSLV)**—See the bit descriptions in [Table 92](#) on page 170.

**Autobaud Enable (ABEN)**—See the bit descriptions in [Table 92](#) on page 170.

**Autobaud Interrupt Enable (ABIEN)**—See the bit descriptions in [Table 92](#) on page 170.

**LIN State Machine (LinState[1:0])**—The LinState is controlled by both hardware and software. Software can force a state change at any time if necessary. In normal operation, software moves the state in and out of Sleep state. For a LIN slave, software changes the state from Sleep to Wait for Break, after which hardware cycles through the Wait for Break, Autobaud, and Active states. Software changes the state from one of the active states to Sleep state, if the LIN bus goes into SLEEP mode. For a LIN master, software changes the state from Sleep to Active, where it remains till the software sets it back to the Sleep state. After configuration, software does not alter the LinState field during operation.

**Transmit Break Length (TxBreakLength)**—Used in LIN mode by the master to control the duration of the transmitted break.

## LIN-UART Address Compare Register

The LIN-UART Address Compare Register stores the multinode network address of the LIN-UART. When the MPMD[1] bit of the LIN-UART Control Register 0 is set, all incoming address bytes are compared to the value stored in this Address Compare Register. Receive interrupts and RDA assertions only occur in the event of a match, see [Table 93](#).

**Table 93. LIN-UART Address Compare Register (U0ADDR = F45H)**

BITS	7	6	5	4	3	2	1	0
FIELD	COMP_ADDR							
RESET	0	0	0	0	0	0	0	0
CPU ACCESS	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F45H, F4DH							

**Note:** R/W = Read/Write

Bit Position	Value	Description
[7:0] COMP_ADDR	—	<b>Compare Address</b> This 8-bit value is compared to the incoming address bytes.

## LIN-UART Baud Rate High and Low Byte Registers

The LIN-UART Baud Rate High and Low Byte registers (see [Table 94](#) and [Table 95](#)) combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the LIN-UART.

**Table 94. LIN-UART Baud Rate High Byte Register (U0BRH = F46H)**

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1	1	1	1	1	1	1	1
CPU ACCESS	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F46H, F4EH							

Note: R/W = Read/Write

Bit Position	Value	Description
[7:0] BRH	—	<b>Baud Rate High</b> High Byte of baud rate divisor value.

**Table 95. LIN-UART Baud Rate Low Byte Register (U0BRL = F47H)**

Bits	7	6	5	4	3	2	1	0
Field	BRL							
Reset	1	1	1	1	1	1	1	1
CPU Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F47H, F4FH							

Note: R/W = Read/Write

Bit Position	Value	Description
[7:0] BRL	—	<b>Baud Rate Low</b> Low Byte of baud rate divisor value.

The LIN-UART data rate is calculated using the following equation for standard UART modes. For LIN protocol, the Baud Rate registers must be programmed with the baud period rather than 1/16 baud period.

► **Note:** *The UART must be disabled when updating the Baud Rate registers because the high and low registers must be written independently.*

The LIN-UART data rate is calculated using the following equation for standard UART operation:

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

The LIN-UART data rate is calculated using the following equation for LIN mode UART operation:

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

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for standard UART operation:

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

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for LIN mode UART operation:

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

The baud rate error relative to the appropriate 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 LIN-UART baud rate error must never exceed 5 percent. [Table 96](#) through [Table 100](#) provide error data for popular baud rates and commonly-used crystal oscillator frequencies for normal UART modes of operation.

**Table 96. LIN-UART Baud Rates, 20.0 MHz System Clock**

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	1	1250.0	0.00	9.60	130	9.62	0.16
625.0	2	625.0	0.00	4.80	260	4.81	0.16
250.0	5	250.0	0.00	2.40	521	2.399	-0.03
115.2	11	113.64	-1.19	1.20	1042	1.199	-0.03
57.6	22	56.82	-1.36	0.60	2083	0.60	0.02
38.4	33	37.88	-1.36	0.30	4167	0.299	-0.01
19.2	65	19.23	0.16				

**Table 97. LIN-UART Baud Rates, 10.0 MHz System Clock**

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	9.60	65	9.62	0.16
625.0	1	625.0	0.00	4.80	130	4.81	0.16
250.0	3	208.33	-16.67	2.40	260	2.40	-0.03
115.2	5	125.0	8.51	1.20	521	1.20	-0.03
57.6	11	56.8	-1.36	0.60	1042	0.60	-0.03
38.4	16	39.1	1.73	0.30	2083	0.30	0.2
19.2	33	18.9	0.16				

**Table 98. LIN-UART Baud Rates, 5.5296 MHz System Clock**

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	9.60	36	9.60	0.00
625.0	N/A	N/A	N/A	4.80	72	4.80	0.00
250.0	1	345.6	38.24	2.40	144	2.40	0.00
115.2	3	115.2	0.00	1.20	288	1.20	0.00
57.6	6	57.6	0.00	0.60	576	0.60	0.00
38.4	9	38.4	0.00	0.30	1152	0.30	0.00
19.2	18	19.2	0.00				

**Table 99. LIN-UART Baud Rates, 3.579545 MHz System Clock**

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	9.60	23	9.73	1.32
625.0	N/A	N/A	N/A	4.80	47	4.76	−0.83
250.0	1	223.72	−10.51	2.40	93	2.41	0.23
115.2	2	111.9	−2.90	1.20	186	1.20	0.23
57.6	4	55.9	−2.90	0.60	373	0.60	−0.04
38.4	6	37.3	−2.90	0.30	746	0.30	−0.04
19.2	12	18.6	−2.90				

**Table 100. LIN-UART Baud Rates, 1.8432 MHz System Clock**

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error(%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error(%)
1250.0	N/A	N/A	N/A	9.60	12	9.60	0.00
625.0	N/A	N/A	N/A	4.80	24	4.80	0.00
250.0	N/A	N/A	N/A	2.40	48	2.40	0.00
115.2	1	115.2	0.00	1.20	96	1.20	0.00
57.6	2	57.6	0.00	0.60	192	0.60	0.00
38.4	3	38.4	0.00	0.30	384	0.30	0.00
19.2	6	19.2	0.00				



# Infrared Encoder/Decoder

## Overview

The Z8 Encore! XP F1680 Series products contain a fully-functional, high-performance UART to Infrared Encoder/Decoder (Endec). The Infrared Endec is integrated with an on-chip UART to allow easy communication between the Z8 Encore! and IrDA Physical Layer Specification, and version 1.3-compliant infrared transceivers. Infrared communication provides secure, reliable, low-cost, point-to-point communication between PCs, PDAs, cell phones, printers, and other infrared enabled devices.

## Architecture

Figure 27 illustrates the architecture of the Infrared Endec.

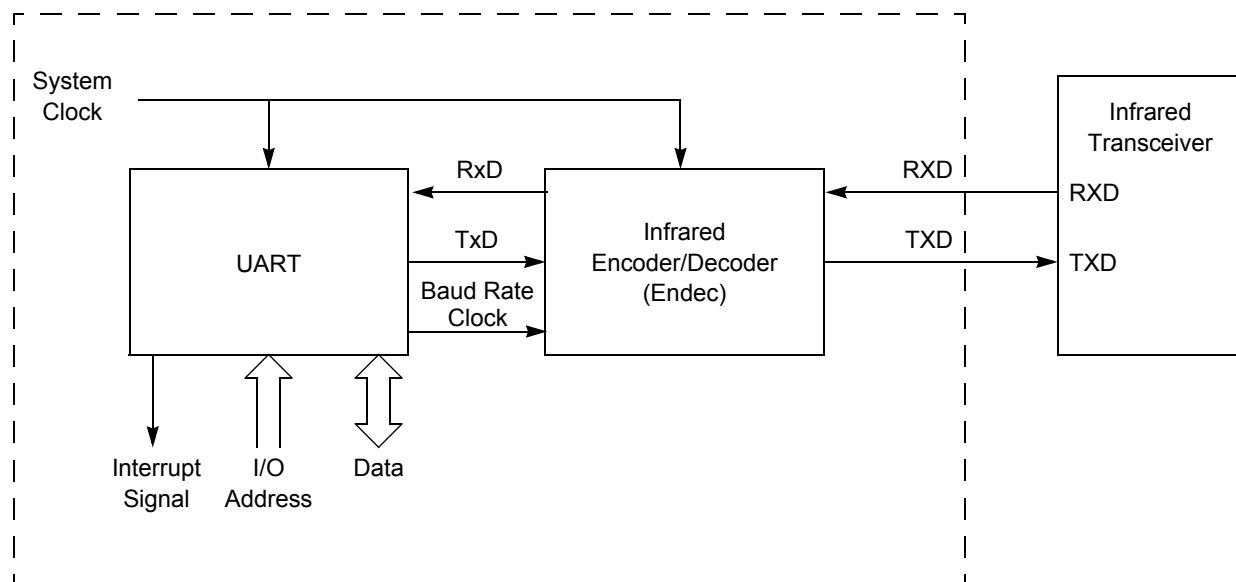


Figure 27. Infrared Data Communication System Block Diagram

## Operation

When the Infrared Endec is enabled, the transmit data from the associated on-chip UART is encoded as digital signals in accordance with the IrDA standard and output to the infrared transceiver through the TXD pin. Likewise, data received from the infrared

transceiver is passed to the Infrared Endec through the RXD pin, decoded by the Infrared Endec and passed to the UART. Communication is half-duplex, that is, simultaneous data transmission and reception is not allowed.

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

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

## Transmitting IrDA Data

The data to be transmitted using the infrared transceiver is first sent to the UART. The UART's transmit signal (TXD) and baud rate clock are used by the IrDA to generate the modulation signal (IR\_TXD) that drives the infrared transceiver. Each UART/Infrared data bit is 16 clocks wide. If the data to be transmitted is 1, the IR\_TXD signal remains low for the full 16-clock period. If the data to be transmitted is 0, the transmitter first outputs a 7-clock low period, followed by a 3-clock high pulse. Finally, a 6-clock low pulse is the output to complete the full 16 clock data period. [Figure 28](#) illustrates IrDA data transmission. When the Infrared Endec is enabled, the UART's TXD signal is internal to the Z8 Encore! XP F1680 Series products while the IR\_TXD signal is the output through the TXD pin.

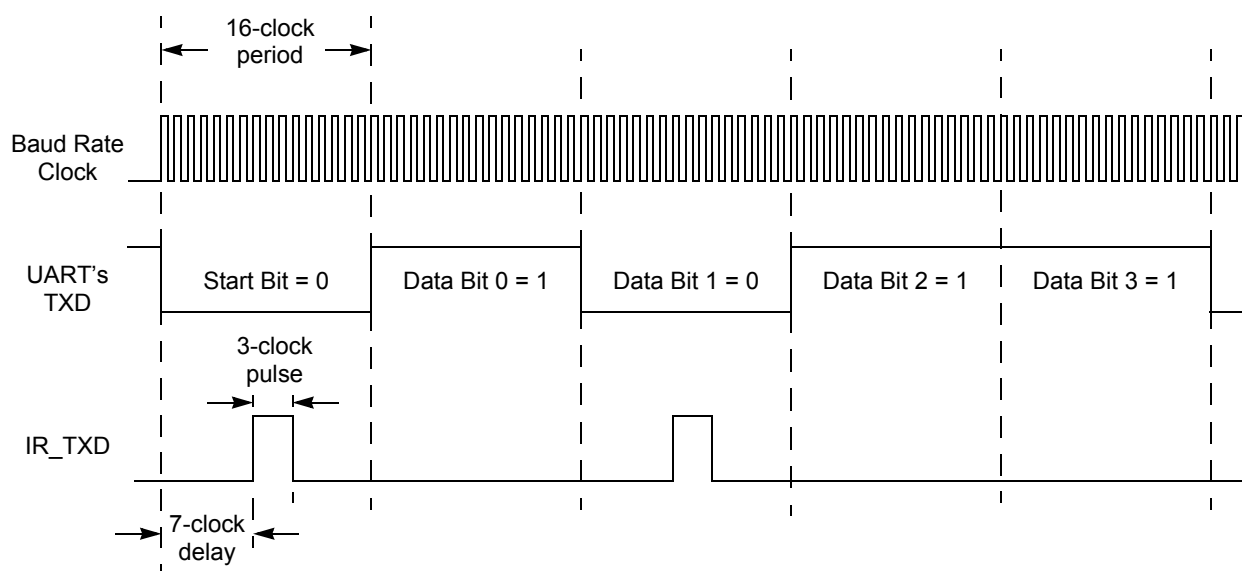


Figure 28. Infrared Data Transmission

## Receiving IrDA Data

Data received from the infrared transceiver using the `IR_RXD` signal through the RXD pin is decoded by the Infrared Endec and passed to the UART. The UART's baud rate clock is used by the Infrared Endec to generate the demodulated signal (`RXD`) that drives the UART. Each UART/Infrared data bit is 16 clocks wide. Figure 29 illustrates data reception. When the Infrared Endec is enabled, the UART's `RXD` signal is internal to the Z8 Encore! XP F1680 Series products while the `IR_RXD` signal is received through the RXD pin.

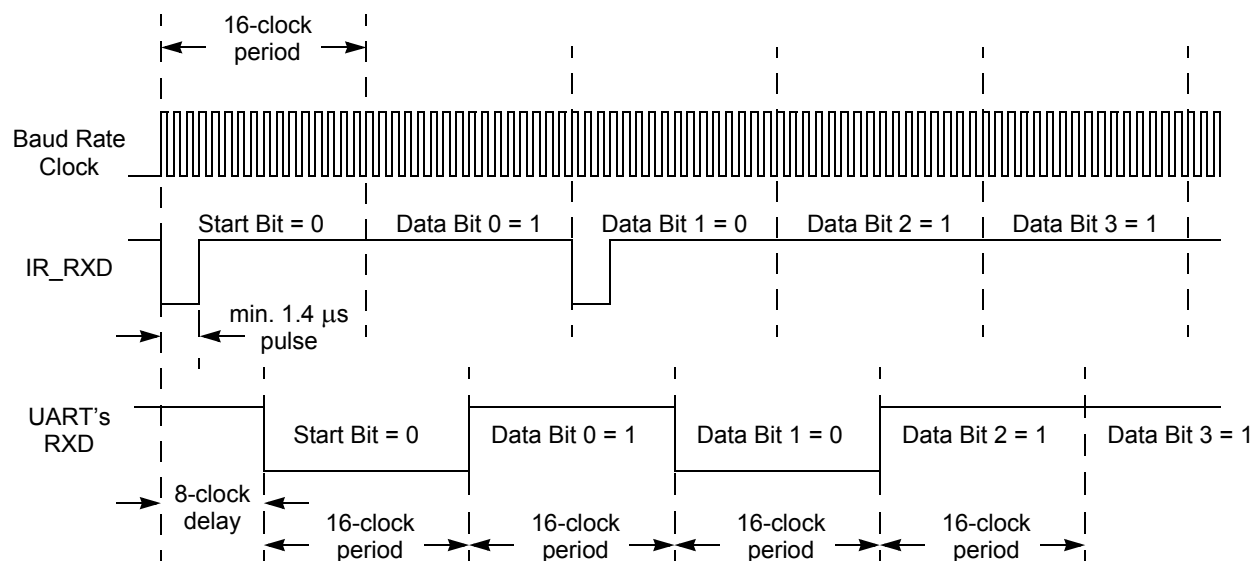


Figure 29. IrDA Data Reception

## Infrared Data Reception



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

## Endec Receiver Synchronization

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

The window remains open until the count again reaches 8 (in other words, 24 baud clock periods since the previous pulse is detected), giving the Endec a sampling window of minus 4 baud rate clocks to plus 8 baud rate clocks around the expected time of an incoming pulse. If an incoming pulse is detected inside this window this process is repeated. If the incoming data is a logical 1 (no pulse), the Endec returns to 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, allowing the Endec to tolerate jitter and baud rate errors in the incoming datastream. 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 Encoder/Decoder Control Register Definitions

All Infrared Endec configuration and status information is set by the UART control registers as defined beginning on page 159.



**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 Encoder/Decoder before enabling the GPIO Port alternate function for the corresponding pin of UART. See [Table 15](#) through [Table 17](#) on pages 52 through 57 for details.*

# Analog-to-Digital Converter

The Z8 Encore! includes an eight-channel Successive Approximation Register Analog-to-Digital converter (SAR ADC). The ADC converts an analog input signal to a 10-bit binary number. The features of the ADC include:

- Eight analog input sources multiplexed with general-purpose I/O ports
- Fast conversion time, less than 4.9  $\mu\text{s}$
- Programmable timing controls
- Interrupt on conversion complete
- Internal 1.6 V voltage reference generator
- Internal reference voltage available externally
- Ability to supply external reference voltage

## Architecture

The ADC architecture ([Figure 30](#) on page 182) consists of an 8-input multiplexer, sample-and-hold amplifier, and 10-bit SAR ADC. The ADC digitizes the signal on a selected channel and stores the digitized data in the ADC data registers. In environments with high electrical noise, an external RC filter must be added at the input pins to reduce high-frequency noise.

## Operation

The ADC converts the analog input,  $\text{ANA}_X$ , to a 10-bit digital representation. The equation for calculating the digital value is calculated by:

$$\text{ADC Output} = 1024 \times (\text{ANA}_X \div V_{\text{REF}})$$

Assuming zero gain and offset errors, any voltage outside the ADC input limits of  $\text{AV}_{\text{SS}}$  and  $V_{\text{REF}}$  returns all 0's or 1's, respectively.

A new conversion can be initiated by software write to the ADC Control Register's `START` bit. Initiating a new conversion stops any conversion currently in progress and begins a new conversion. To avoid disrupting a conversion already in progress, the `START` bit can be read to indicate ADC operation status (busy or available).

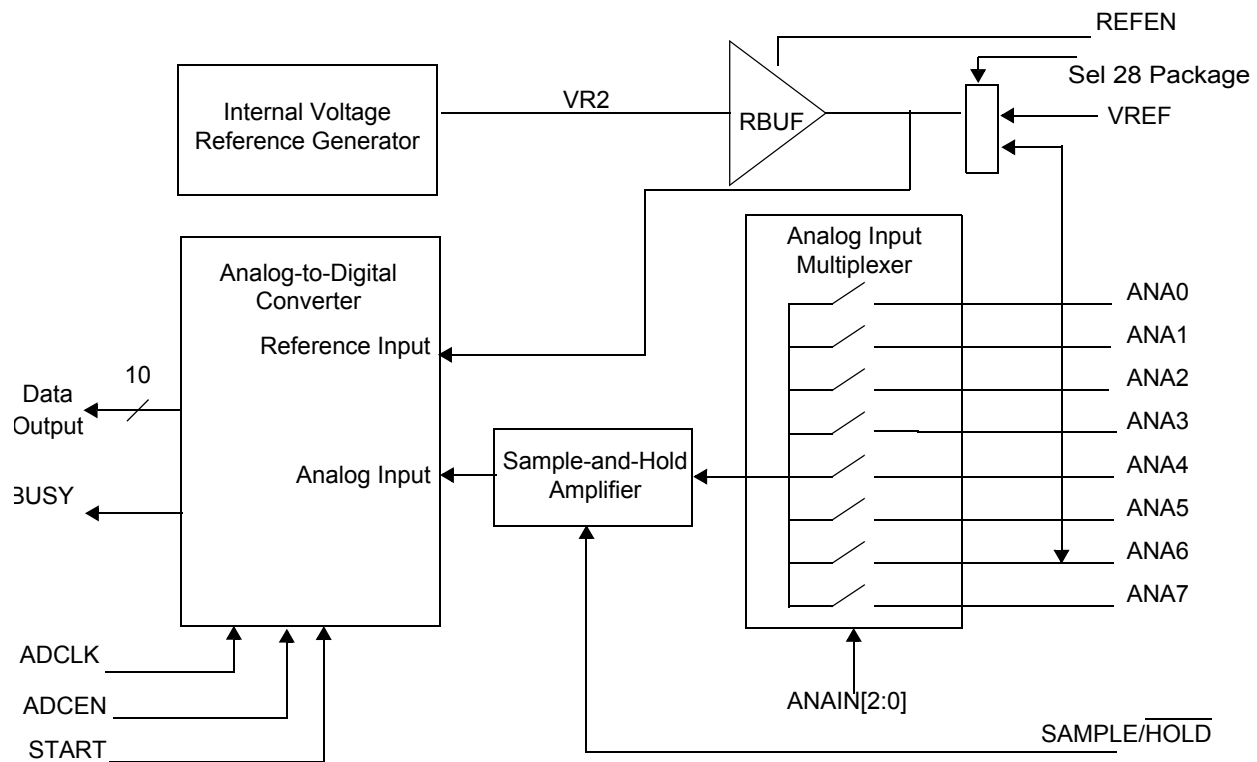


Figure 30. Analog-to-Digital Converter Block Diagram

## ADC Timing

Each ADC measurement consists of 3 phases:

1. Input sampling (programmable, minimum of 1.8  $\mu$ s).
2. Sample-and-hold amplifier settling (programmable, minimum of 0.5  $\mu$ s).
3. Conversion is 13 ADCLK cycles.

Figure 31 on page 183 illustrates the timing of an ADC conversion.

## ADC Interrupt

The ADC can generate an interrupt request when a conversion is completed. An interrupt request that is pending when the ADC is disabled is not automatically cleared.

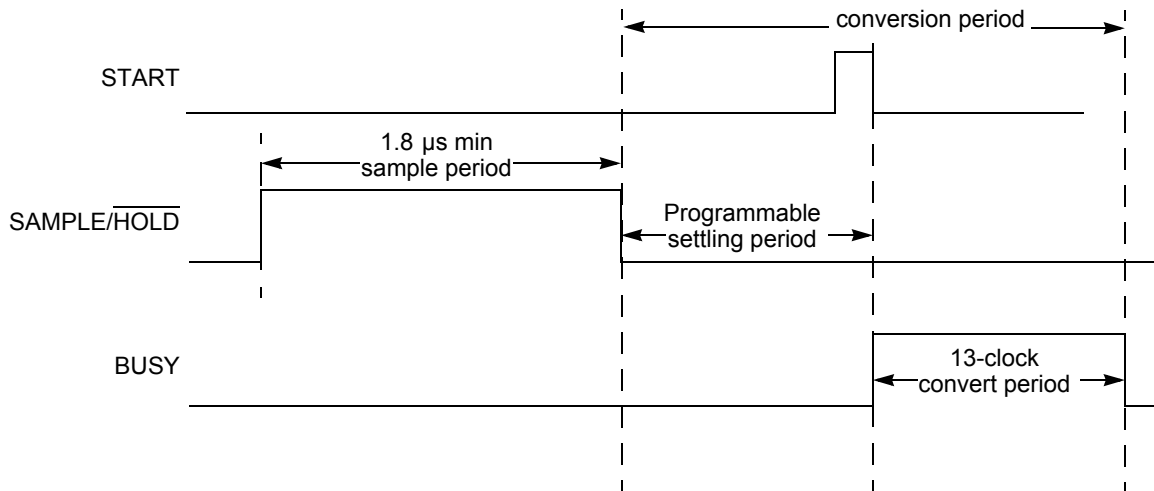


Figure 31. ADC Timing Diagram

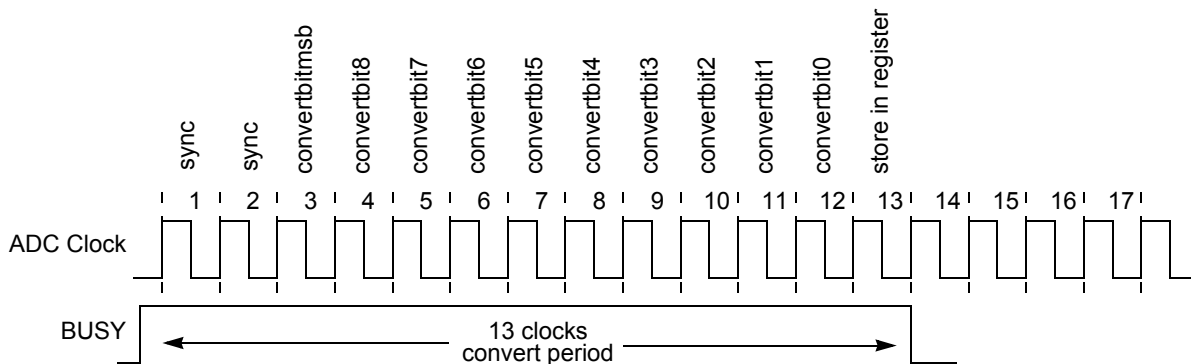


Figure 32. ADC Convert Timing

## Reference Buffer

The reference buffer, RBUF, supplies the reference voltage for the ADC. When enabled, the internal voltage reference generator supplies the ADC and this voltage is available on the  $V_{REF}$  pin. When RBUF is disabled, the ADC must have the reference voltage supplied externally through the  $V_{REF}$  pin. RBUF is controlled by the  $REFEN$  bit in the ADC Control Register.

## Internal Voltage Reference Generator

The Internal Voltage Reference Generator provides the voltage, VR2, for the RBUF. VR2 is 1.6 V.

## Calibration and Compensation

You can calibrate and store the values into Flash, or the user code can perform a manual offset calibration. There is no provision for manual gain calibration.

## ADC Control Register Definitions

The ADC Control Registers are described in the following paragraphs.

### ADC Control Register 0

The ADC Control Register 0 initiates the A/D conversion and provides ADC status information ([Table 101](#)).

**Table 101. ADC Control Register 0 (ADCCTL0)**

BITS	7	6	5	4	3	2	1	0
FIELD	START	INTREF_SEL	REFEN	ADCEN	ANAIN[3:0]			
RESET	0	0	0	0	0	0	0	0
R/W	R/W1	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F70h							

Bit Position	Value (H)	Description
[7] START	0	<b>ADC Start/Busy</b> Writing a 0 has no effect. Reading a 0 indicates the ADC is available to begin a conversion.
	1	Writing a 1 starts a conversion. Reading a 1 indicates that a conversion is currently in progress.
[6] INTREF_SEL	0	Select 1.6 V as internal reference.
	1	Select AVDD as internal reference.
[5] REFEN	0	Select external reference.
	1	Select internal reference.
[4] ADCEN	0	ADC is disabled.
	1	ADC is enabled for normal use. This bit cannot change with bit 7 (START) at the same time.

Bit Position	Value (H)	Description
[3:0] ANAIN	0000	<b>Analog Input Select</b> ANA0 input is selected for analog-to-digital conversion.
	0001	ANA1 input is selected for analog-to-digital conversion.
	0010	ANA2 input is selected for analog-to-digital conversion.
	0011	ANA3 input is selected for analog-to-digital conversion.
	0100	ANA4 input is selected for analog-to-digital conversion.
	0101	ANA5 input is selected for analog-to-digital conversion.
	0110	ANA6 input is selected for analog-to-digital conversion.
	0111	ANA7 input is selected for analog-to-digital conversion.
	1000	Hold LPO input nodes (ANA1 and ANA2) to ground.
	1001	Temperature Sensor.
	1100	Temperature Sensor output to ANA3 PAD.
	1101	vbg_chop signal output to ANA3 PAD.
	Others	Reserved.

## ADC Raw Data High Byte Register

The ADC Raw Data High Byte Register (see [Table 102](#) on page 185) contains the upper 8 bits of raw data from the ADC output. Access to the ADC Raw Data High Byte register is Read-only. This register is used for test only.

**Table 102. ADC Raw Data High Byte Register (ADCRD\_H)**

BITS	7	6	5	4	3	2	1	0
FIELD	ADCRDH							
RESET	X							
R/W	R							
ADDR	F71H							

Bit Position	Value (H)	Description
[7:0]	00–FF	<b>ADC Raw Data High Byte</b> The data in this register is the raw data coming from the SAR Block. It will change as the conversion is in progress. This register is used for testing only.

## ADC Data High Byte Register

The ADC Data High Byte Register (Table 103) contains the upper eight bits of the ADC output. Access to the ADC Data High Byte Register is Read-only. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

Table 103. ADC Data High Byte Register (ADCD\_H)

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

Bit Position	Value (H)	Description
[7:0]	00–FF	<b>ADC High Byte</b> The last conversion output is held in the data registers until the next ADC conversion has completed.

## ADC Data Low Bit Register

The ADC Data Low Bit Register (Table 104) contains the lower bits of the ADC output as well as an overflow status bit. Access to the ADC Data Low Bits Register is Read-Only. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

Table 104. ADC Data Low Bits Register (ADCD\_L)

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

Bit Position	Value (H)	Description
[7,6]	00–11b	<b>ADC Low Bit</b> These bits are the 2 least significant bits of the 10-bit ADC output. These bits are undefined after a Reset. The low bits are latched into this register whenever the ADC Data High Byte register is read.
[5:0] Reserved	0	Reserved—Must be 0.

## Sample Settling Time Register

The Sample Settling Time Register (Table 105) is used to program the length of time from the SAMPLE/HOLD signal to the START signal, when the conversion can begin. The number of clock cycles required for settling will vary from system to system depending on the system clock period used. The system designer should program this register to contain the number of clocks required to meet a 0.5  $\mu$ s minimum settling time.

Table 105. Sample Settling Time (ADCSST)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				SST			
RESET	0				1	1	1	1
R/W	R				R/W			
ADDR	F74H							
Bit Position	Value (H)	Description						
[7:4]	0h	Reserved—Must be 0.						
[3:0] SST	0h-Fh	Sample settling time in number of system clock periods to meet 0.5 μs minimum.						

## Sample Time Register

The Sample Time Register (Table 106) is used to program the length of active time for the sample after a conversion begins by setting the START bit in the ADC Control Register or initiated by the PWM. The number of system clock cycles required for sample time varies from system to system, depending on the clock period used. The system designer should program this register to contain the number of system clocks required to meet a 1.8  $\mu$ s minimum sample time.

Table 106. Sample Time (ADCST)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved		ST					
RESET	0		1	1	1	1	1	1
R/W	R/W		R/W					
ADDR	F75H							

Bit Position	Value (H)	Description
[7:6]	0	Reserved—Must be 0.
[5:0] ST	00-3F	Sample Hold time in number of system clock periods to meet 1.8 μs minimum.

## ADC Clock Prescale Register

The ADC Clock Prescale Register (Table 107) is used to provide a divided system clock to the ADC. When this register is programmed with 0h, the System Clock is used for the ADC Clock. DIV16 has the highest priority, DIV2 has the lowest priority.

**Table 107. ADC Clock Prescale Register (ADCCP)**

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				DIV16	DIV8	DIV4	DIV2
RESET	0				0	0	0	0
R/W	R/W							
ADDR	F76H							

Bit Position	Value (H)	Description
[0] DIV2	0	<b>DIV2</b> Clock is not divided.
	1	System Clock is divided by 2 for ADC Clock.
[1] DIV4	0	<b>DIV4</b> Clock is not divided.
	1	System Clock is divided by 4 for ADC Clock.
[2] DIV8	0	<b>DIV8</b> Clock is not divided.
	1	System Clock is divided by 8 for ADC Clock.
[3] DIV16	0	<b>DIV16</b> Clock is not divided.
	1	System Clock is divided by 16 for ADC Clock.
[7:4]	0h	Reserved—Must be 0.



**Caution:** *The maximum ADC clock at 2.7–3.6 V is 5 MHz. The maximum ADC clock at 1.8–2.7 V is 2.5 MHz. Set the Prescale Register correctly according to the different system clocks. See Table 198 on page 347 for details.*

## Low-Power Operational Amplifier

The low-power operational amplifier is a standard operational amplifier designed for current measurements. Each of the three ports of the amplifier is accessible from the package pins. The inverting input is commonly used to connect to the current source. The output node connects an external feedback network to the inverting input. The non-inverting output is required to apply a non-zero bias point. In a standard single-supply system, this bias point must be substantially above ground to measure positive input currents. The non-inverting input may also be used for offset correction.

► **Note:** *This is a voltage gain operational amplifier. Its transimpedance nature is determined by the feedback network.*

The low-power operational amplifier contains only one pin configuration; ANA0 is the output/feedback node, ANA1 is the inverting input and ANA2 is the non-inverting input.

To use the low-power operational amplifier, it must be enabled in the [Power Control Register Definitions](#) on page 47. The default state of the low-power operational amplifier is OFF. To use the low-power operational amplifier, the TRAM bit must be cleared, turning it ON ([Power Control Register 0](#) on page 47). When making normal ADC measurements on ANA0 (not transimpedance measurements), the TRAM bit must be OFF. Turning the TRAM bit ON interferes with normal ADC measurements. Finally, this bit enables the amplifier even in STOP mode. If the amplifier is not required in STOP mode, disable it. Failing to perform this results in STOP mode currents greater than specified.

As with other ADC measurements, any pins used for analog purposes must be configured as in the GPIO registers (see [Port A–E Alternate Function Subregisters](#) on page 64).

Standard transimpedance measurements are made on ANA0 as selected by the ANAIN[3:0] bits of [ADC Control Register 0](#) on page 184. It is also possible to make single-ended measurements on ANA1 and ANA2 when the amplifier is enabled which is often useful for determining offset conditions.



# Enhanced Serial Peripheral Interface

## Overview

The Enhanced Serial Peripheral Interface (ESPI) supports Serial Peripheral Interface (SPI) and other synchronous serial interface modes such as Inter IC Sound (I<sup>2</sup>S) and time division multiplexing (TDM). ESPI includes the following features:

- Full-duplex, synchronous, character-oriented communication.
- Four-wire interface ( $\overline{SS}$ , SCK, MOSI, and MISO).
- Data shift register is buffered to enable high throughput.
- Master mode transfer rates up to a maximum of one-half the system clock frequency.
- Slave mode transfer rates up to a maximum of one-eighth the system clock frequency.
- Error detection.
- Dedicated Programmable Baud Rate Generator.
- Data transfer control via polling, interrupt.

## Architecture

The ESPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (serial clock, transmit data, receive data and slave select). The ESPI block consists of a shift register, data buffer register, a Baud Rate (clock) Generator, control/status registers, and a control state machine. Transmit and receive transfers are in synch as there is a single shift register for both transmitting and receiving data. [Figure 33](#) on page 192 illustrates the block diagram of ESPI.

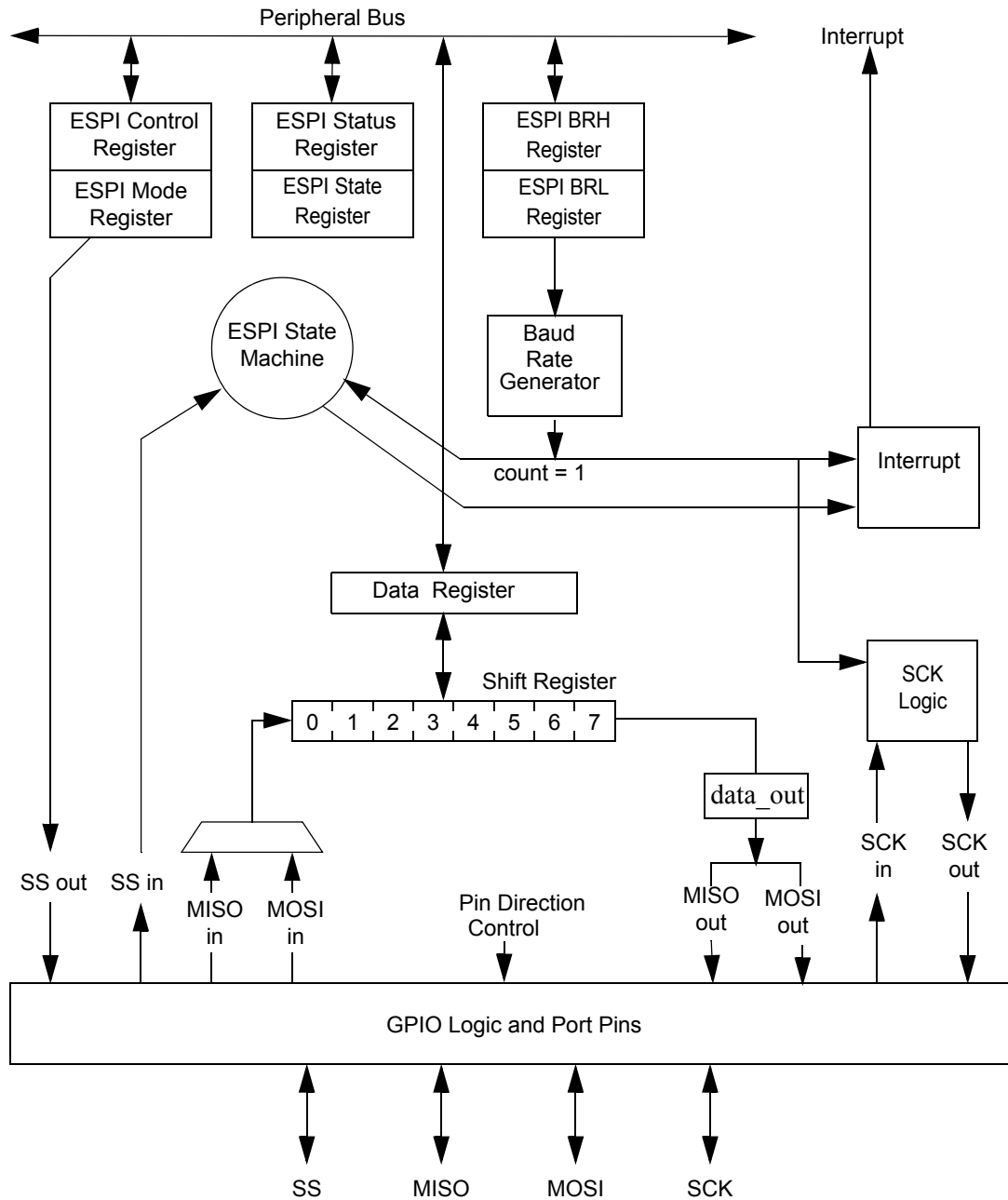


Figure 33. ESPI Block Diagram

## ESPI Signals

The four ESPI signals are:

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

The following paragraphs discuss these signals 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. Data is transferred either most significant bit first or least significant bit first as determined by the LSBF bit of the ESPI Mode register. The MISO pin of a Slave device is placed in a high-impedance state if the Slave is not selected. When the ESPI is not enabled, this signal is in a high-impedance state. The direction of this pin is controlled by the MMEN bit of the ESPI Control Register.

### 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. Data is transferred either as most significant bit first or least significant bit first as determined by the LSBF bit of the ESPI mode register. When the ESPI is not enabled, this signal is in a high-impedance state. The direction of this pin is controlled by the MMEN bit of the ESPI Control Register.

### Serial Clock

The Serial Clock (SCK) synchronizes data movement both in and out of the shift register via the MOSI and MISO pins. In MASTER mode (MMEN = 1), the ESPI's Baud Rate Generator creates the serial clock and drives it out on its SCK pin to the slave devices. In SLAVE mode, the SCK pin is an input. Slave devices ignore the SCK signal, unless their  $\overline{SS}$  pin is asserted.

The Master and Slave are each capable of exchanging a character of data during a sequence of NUMBITS clock cycles ([Table 113](#) on page 209). In both Master and Slave ESPI devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. SCK phase and polarity is determined by the PHASE and CLKPOL bits in the ESPI Control register.

## Slave Select

The Slave Select signal is a bidirectional framing signal with several modes of operation to support SPI and other synchronous serial interface protocols. The Slave Select mode is selected by the SSMD field of the ESPI Mode register. The direction of the  $\overline{SS}$  signal is controlled by the SSIO bit of the ESPI Mode register. The  $\overline{SS}$  signal is an input on slave devices and is an output on the active Master device. Slave devices ignore transactions on the bus unless their Slave Select input is asserted. In SPI MASTER mode, additional GPIO pins are required to provide Slave Selects if there is more than one slave device.

## ESPI Register Overview

The ESPI Control/Status Registers are summarized in [Table 108](#).

**Table 108. ESPI Registers**

Address	Even Address	Odd Address
XX0	Data	Transmit Data Command and Receive Data Buffer Control
XX2	Control	Mode
XX4	Status	State
XX6	Baud Rate High	Baud Rate Low

## Operation

During a transfer, data is sent and received simultaneously by both the Master and Slave devices. Separate signals are required for transmit data, receive data, and the serial clock. When a transfer occurs, a multi-bit (typically 8-bit) character is shifted out one data pin and a multi-bit character is simultaneously shifted in on second data pin. An 8-bit shift register in the Master and an 8-bit shift register in the Slave are connected as a circular buffer. The ESPI shift register is buffered to support back-to-back character transfers in high performance applications.

A transaction is initiated when the Data register is written in the Master device. The value from the Data register is transferred into the shift register and the I2C transaction begins. At the end of each character transfer, if the next transmit value has been written to the data register, the data and shift register values are swapped, which places the new transmit data into the shift register and the shift register contents (receive data) into the data register. At that point the Receive Data Register Not Empty signal is asserted (RDRNE bit set in the Status Register). Once software reads the receive data from the Data register, the Transmit Data Register Empty signal is asserted (TDRE bit set in the Status Register) to request the next transmit byte. To support back-to-back transfers without an intervening pause, the receive and transmit interrupts must be serviced when the current character is being transferred.

The Master sources the Serial Clock (SCK) and Slave Select signal ( $\overline{SS}$ ) during the transfer.

Internal data movement (by software) to/from the ESPI block is controlled by the Transmit Data Register Empty (TDRE) and Receive Data Register Not Empty (RDRNE) signals. These signals are Read-only bits in the ESPI Status register. When either the TDRE or RDRNE bits assert, an interrupt is sent to the interrupt controller. In many cases the software application is only moving information in one direction. In this case either the TDRE or RDRNE interrupts may be disabled to minimize software overhead.

Unidirectional data transfer is supported by setting the ESPIEN1,0 bits in the Control Register to 10 or 01.

## Throughput

In MASTER mode, the maximum SCK rate supported is one-half the system clock frequency. This is achieved by programming the value 0001H into the Baud Rate High/Low register pair. Though each character will be transferred at this rate it is unlikely that software interrupt routines could keep up with this rate. In SPI mode the transfer will automatically pause between characters until the current receive character is read and the next transmit data value is written.

In SLAVE mode, the transfer rate is controlled by the Master. As long as the TDRE and RDRNE interrupt are serviced before the next character transfer completes, the Slave will keep up with the Master. In SLAVE mode the baud rate must be restricted to a maximum of one-eighth of the system clock frequency to allow for synchronization of the SCK input to the internal system clock.

## ESPI Clock Phase and Polarity Control

The ESPI supports four combinations of serial clock phase and polarity using two bits in the ESPI Control register. The clock polarity bit, CLKPOL, selects an active High or active Low clock and has no effect on the transfer format. [Table 109](#) lists the ESPI Clock Phase and Polarity Operation parameters. The clock phase bit, PHASE, selects one of two fundamentally different transfer formats. The data is output a half-cycle before the receive clock edge which provides a half cycle of setup and hold time.

**Table 109. ESPI 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 when Phase Equals Zero

Figure 34 illustrates the timing diagram for an SPI type transfer, in which  $PHASE=0$ . For SPI transfers the clock only toggles during the character transfer. The two SCK waveforms show polarity with  $CLKPOL = 0$  and  $CLKPOL = 1$ . The diagram may be interpreted as either a Master or Slave timing diagram because the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the Master and the Slave.

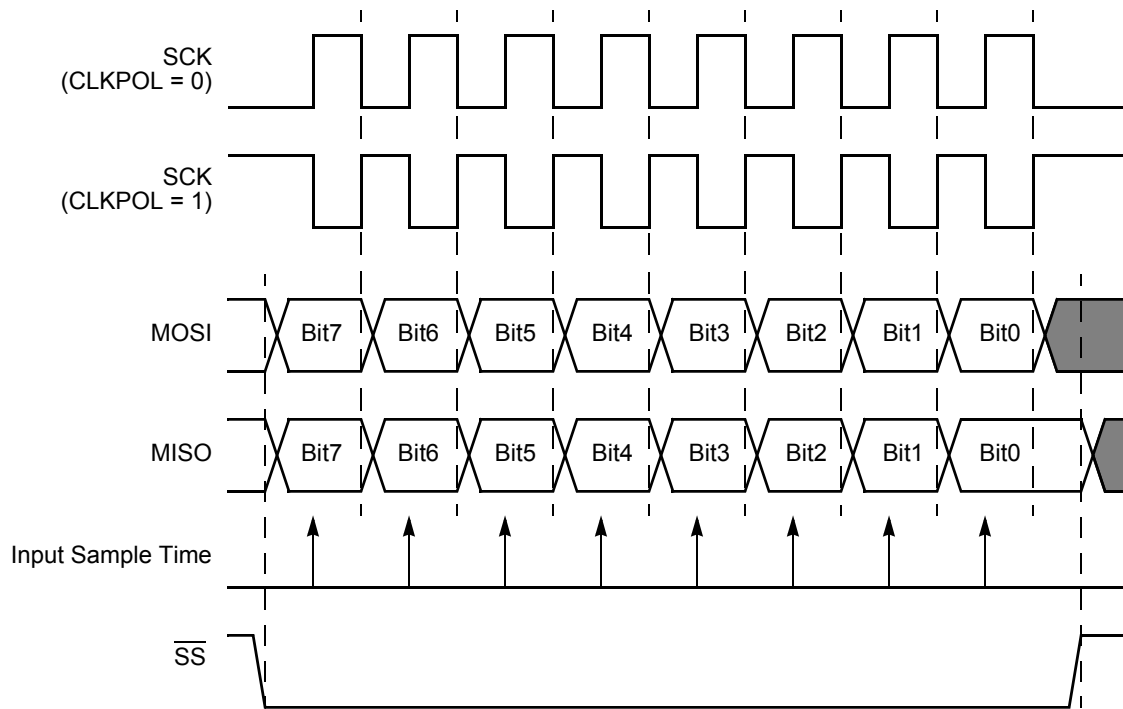


Figure 34. ESPI Timing when  $PHASE = 0$

### Transfer Format When Phase Equals One

Figure 35 on page 197 illustrates the timing diagram for an SPI type transfer in which  $PHASE$  is one. For SPI transfers the clock only toggles during the character transfer. Two waveforms are depicted for SCK, one for  $CLKPOL = 0$  and another for  $CLKPOL = 1$ .

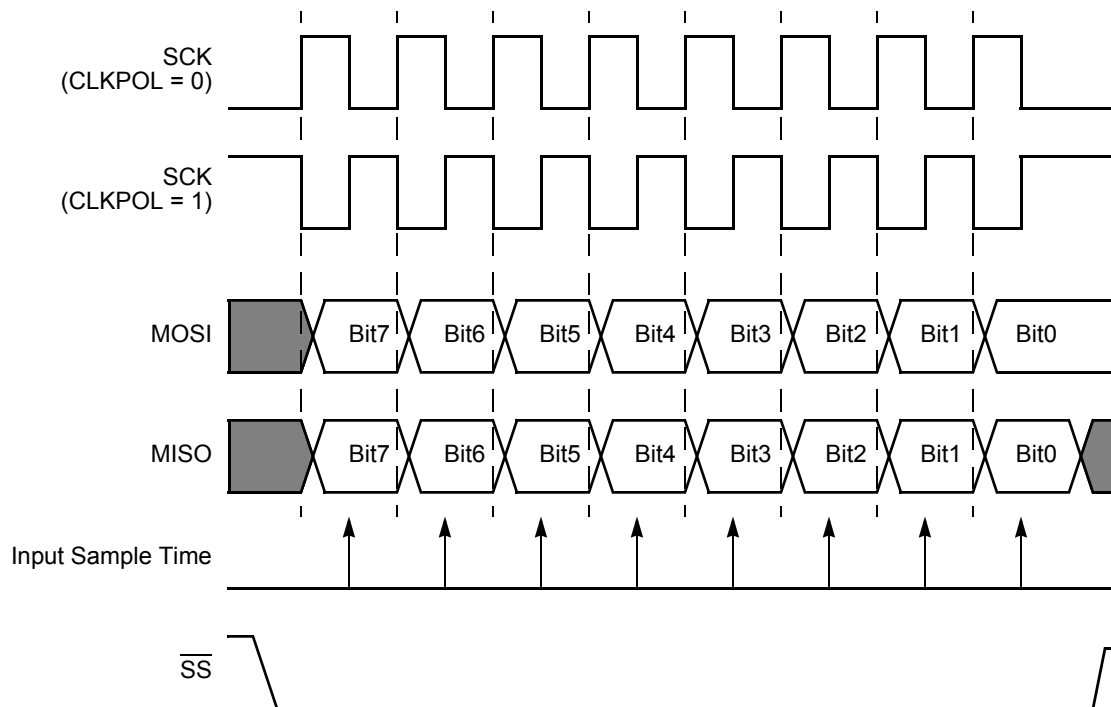


Figure 35. ESPI Timing when PHASE = 1

## Slave Select Modes of Operation

This section describes the different modes of data transfer supported by the ESPI block. The mode is selected by the Slave Select Mode (SSMD) field of the Mode Register.

### SPI Mode

This mode is selected by setting the SSMD field of the Mode Register to 00. In this mode software controls the assertion of the  $\overline{SS}$  signal directly via the SSV bit of the SPI Transmit Data Command register. Software can be used to control an SPI mode transaction. Prior to or simultaneously with writing the first transmit data byte; software sets the SSV bit. Software sets the SSV bit either by performing a byte write to the Transmit Data Command register prior to writing the first transmit character to the Data register or by performing a word write to the Data register address which loads the first transmit character and simultaneously sets the SSV bit.  $\overline{SS}$  will remain asserted when one or more characters are transferred. There are two mechanisms for deasserting  $\overline{SS}$  at the end of the transaction. One method used by software is to set the TEOF bit of the Transmit Data Command register, when the last TDRE interrupt is being serviced (set TEOF before or simultaneously with writing the last data byte). Once the last bit of the last character is

transmitted, the hardware will automatically deassert the SSV and TEOF bits. The second method is for software to directly clear the SSV bit after the transaction completes. If software clears the SSV bit directly it is not necessary for software to also set the TEOF bit on the last transmit byte. After writing the last transmit byte, the end of the transaction can be detected by waiting for the last RDRNE interrupt or monitoring the TFST bit in the ESPI Status register.

The transmit underrun and receive overrun errors will not occur in an SPI mode Master. If the RDRNE and TDRE requests have not been serviced before the current byte transfer completes, SCLK will be paused until the data register is read and written. The transmit underrun and receive overrun errors will occur in a Slave if the Slave's software does not keep up with the Master data rate. In this case the shift register in the Slave will be loaded with all 1s.

In the SPI mode, the SCK is active only for the data transfer with one SCK period per bit transferred. If the SPI bus has multiple Slaves, the Slave Select lines to all or all but one of the Slaves must be controlled independently by software using GPIO pins. [Figure 36](#) illustrates multiple character transfer in SPI mode.

► **Note:** When character  $n$  is transferred via the shift register, software responds to the receive request for character  $n-1$  and the transmit request for character  $n+1$ .

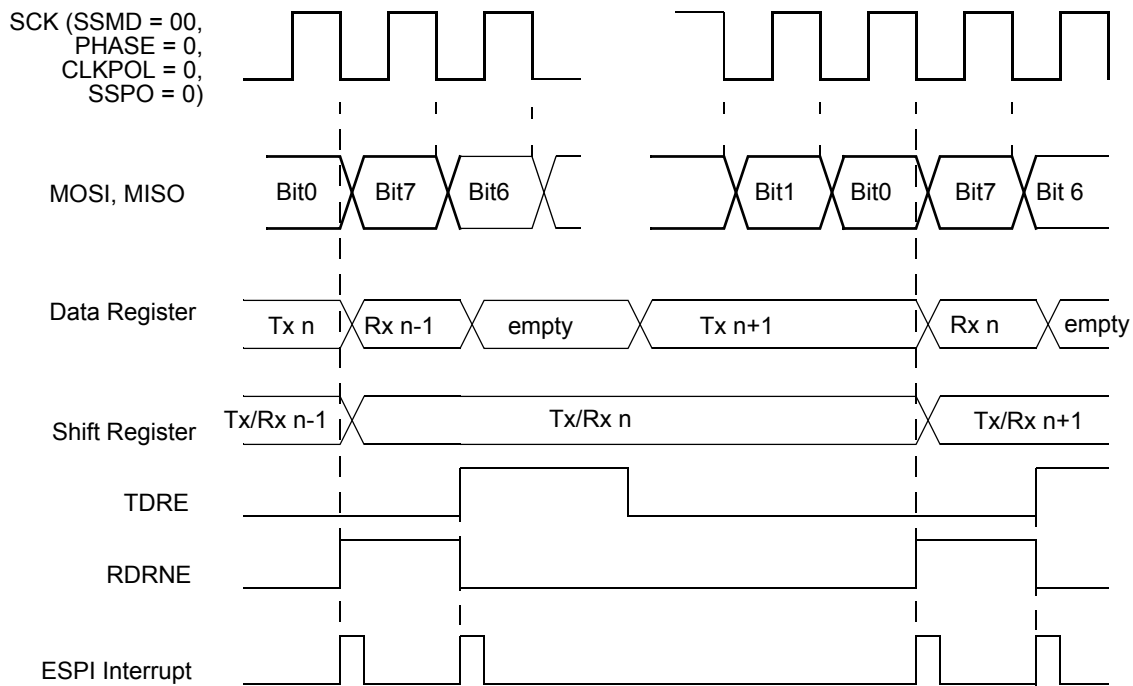


Figure 36. SPI Mode (SSMD = 00)

### Synchronous Frame Sync Pulse Mode

This mode is selected by setting the SSMD field of the Mode Register to 10. This mode is typically used for continuous transfer of fixed length frames where the frames are delineated by a pulse of duration one SCK period. The SSV bit in the ESPI Transmit Data Command register does not control the  $\overline{SS}$  pin directly in this mode. SSV must be set before or in sync with the first transmit data byte being written. The  $\overline{SS}$  signal will assert 1 SCK cycle before the first data bit and will stop after 1 SCK period. SCK is active from the initial assertion of  $\overline{SS}$  until the transaction end due to lack of transmit data.

The transaction is terminated by the Master when it no longer has data to send. If TDRE=1 at the end of a character, the  $\overline{SS}$  output will remain detached and SCK stops after the last bit is transferred. The TUND bit (transmit underrun) will assert in this case. Once the transaction has completed, hardware will clear the SSV bit. Figure 37 illustrates a frame with synchronous frame sync pulse mode.

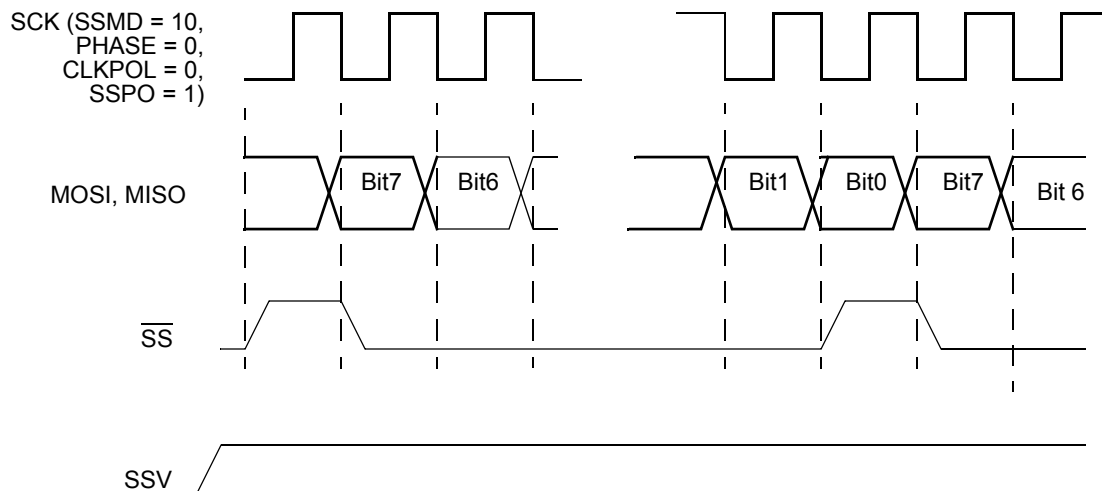


Figure 37. Synchronous Frame Sync Pulse mode (SSMD = 10)

### Synchronous Framing with $\overline{SS}$ Mode

This mode is selected by setting the SSMD field of the Mode Register to 11. Figure 38 on page 200 illustrates synchronous message framing mode with  $\overline{SS}$  alternating between consecutive frames. A frame consists of a fixed number of data bytes as defined by software. An example of this mode is the Inter-IC Sound (I<sup>2</sup>S) protocol which is used to transfer left/right channel audio data. The SSV indicates whether the corresponding bytes are left or right channel data. The SSV value must be updated when servicing the TDRE interrupt/request for the first byte in a left or write channel frame. This can be

accomplished by performing a word write when writing the first byte of the audio word, which will update both the ESPI Data and Transmit Data Command words or by doing a byte write to update SSV followed by a byte Write to the data register. The  $\overline{SS}$  signal will lead the data by one SCK period.

The transaction is terminated when the Master has no more data to transmit. After the last bit is transferred, SCLK will stop and  $\overline{SS}$  and SSV will return to their default states. A transmit underrun error will occur at this point.

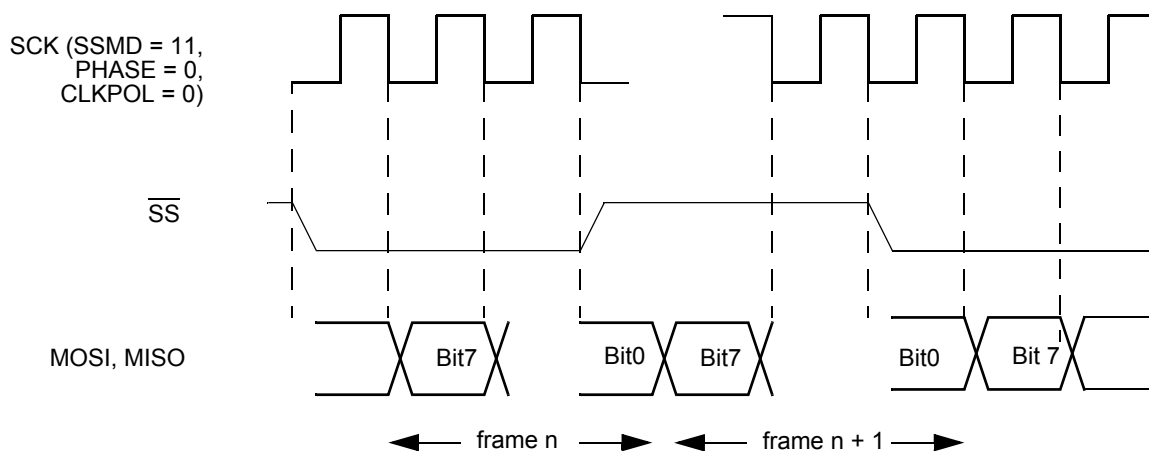


Figure 38. Synchronous Message Framing Mode (SSMD = 11), Multiple Frames

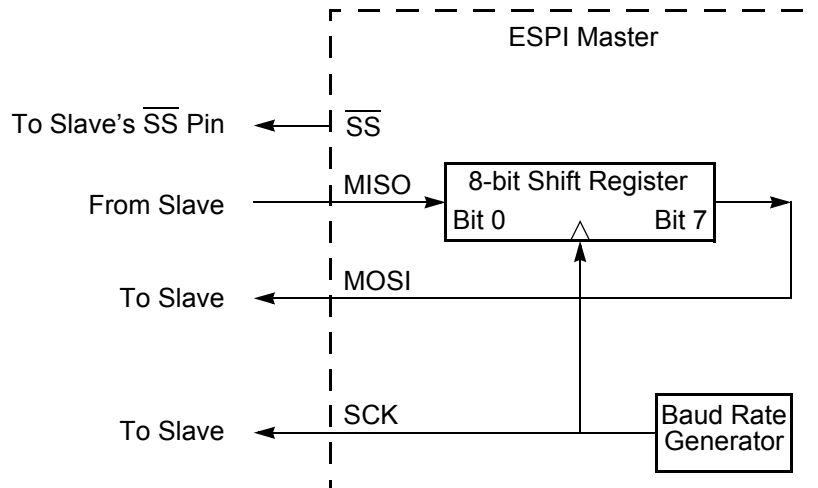
## SPI Protocol Configuration

This section describes in detail how to configure the ESPI block for the SPI protocol. In the SPI protocol the Master sources the SCK and asserts Slave Select signals to one or more Slaves. The Slave Select signals are typically active Low.

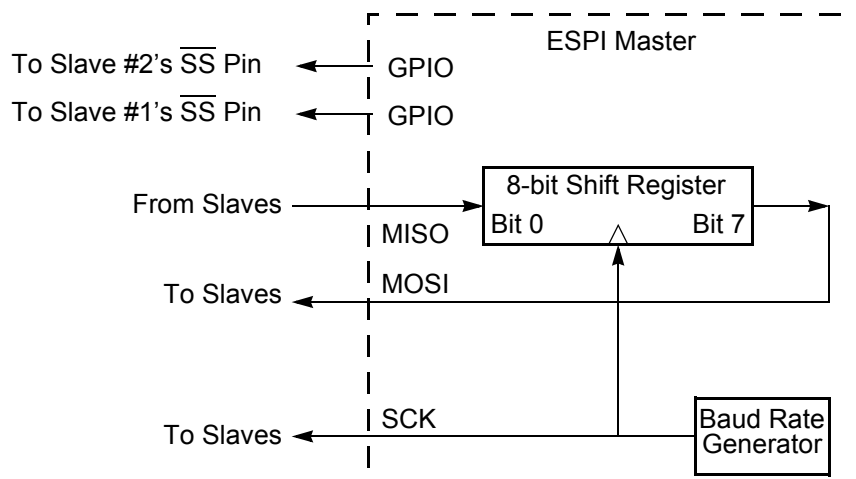
### SPI Master Operation

The ESPI block is configured for MASTER mode operation by setting the MMEN bit = 1 in the ESPICTL register. The SSMD field of the ESPI Mode register is set to 00 for SPI protocol mode. The PHASE, CLKPOL, and WOR bits in the ESPICTL register and the NUMBITS field in the ESPI Mode register must be set to be consistent with the Slave SPI devices. Typically for an SPI Master, LSBF = 0, SSIO = 1, and SSPO = 0.

The appropriate GPIO pins are configured for the ESPI alternate function on the MOSI, MISO and SCK pins. Typically the GPIO for the ESPI  $\overline{SS}$  pin is configured in an alternate function mode as well though the software can use any GPIO pin(s) to drive one or more Slave select lines. If the ESPI  $\overline{SS}$  signal is not used to drive a Slave select the SSIO bit should still be set to 1 in a single Master system. [Figure 39](#) and [Figure 40](#) on page 201 display the ESPI block configured as an SPI Master.



**Figure 39. ESPI Configured as an SPI Master in a Single Master, Single Slave System**



**Figure 40. ESPI Configured as an SPI Master in a Single Master, Multiple Slave System**

### Multi-Master SPI 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 be configured in open-drain mode to prevent bus contention. At any time, only one SPI device is configured as the Master and all other devices on the bus are configured as slaves. The Master asserts the  $\overline{SS}$  pin on the selected slave. Then, the active Master drives the clock and transmits data on the SCK and MOSI pins to the SCK and MOSI pins on the Slave (including those Slaves which are not enabled). The enabled slave drives data out its MISO pin to the MISO Master pin.

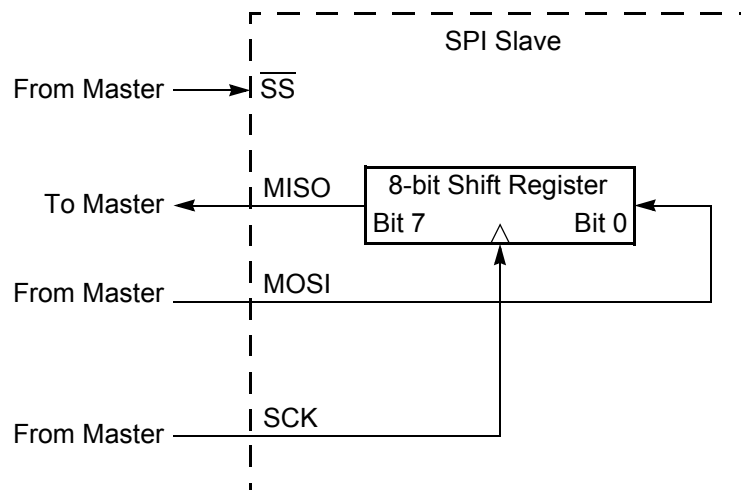
When the ESPI is configured as a Master in a Multi-Master SPI system, the  $\overline{SS}$  pin must be configured as an input. The  $\overline{SS}$  input signal on a device configured as a Master should remain High. If the  $\overline{SS}$  signal on the active Master goes Low (indicating another Master is accessing this device as a Slave), a Collision error flag is set in the ESPI Status register. The Slave select outputs on a Master in a Multi-Master system must come from GPIO pins.

### SPI Slave Operation

The ESPI block is configured for SLAVE mode operation by setting the MMEN bit = 0 in the ESPICTL register, and setting the SSIO bit = 0 in the ESPIMODE register. The SSMD field of the ESPI Mode register is set to 00 for SPI protocol mode. The PHASE, CLKPOL, and WOR bits in the ESPICTL register and the NUMBITS field in the ESPIMODE register must be set to be consistent with the other SPI devices. Typically for an SPI Slave, SSPO = 0.

If the Slave has data to send to the Master, the data must be written to the Data register before the transaction starts (first edge of SCK when  $\overline{SS}$  is asserted). If the Data register is not written prior to the Slave transaction, the MISO pin outputs all 1's.

Due to the delay resulting from synchronization of the  $\overline{SS}$  and SCK input signals to the internal system clock, the maximum SCK baud rate that can be supported in SLAVE mode is the system clock frequency divided by 4. This rate is controlled by the SPI Master. [Figure 41](#) illustrates the ESPI configuration in SPI SLAVE mode.



**Figure 41. ESPI Configured as an SPI Slave**

## Error Detection

Error events detected by the ESPI block are described in this section. Error events generate an ESPI interrupt and set a bit in the ESPI Status register. The error bits of the ESPI Status register are Read/Write 1 to clear.

### Transmit Underrun

A transmit underrun error occurs for a Master with SSMD = 10 or 11 when a character transfer completes and TDRE = 1. In these modes when a transmit underrun occurs the transfer will be aborted (SCK will halt and SSV will be deasserted). For a Master in SPI mode (SSMD = 00), a transmit underrun is not signaled since SCK will pause and wait for the data register to be written.

In SLAVE mode, a transmit underrun error occurs if TDRE = 1 at the start of a transfer. When a transmit underrun occurs in SLAVE mode, ESPI will transmit a character of all 1s.

A transmit underrun sets the TUND bit in the ESPI Status register to 1. Writing a 1 to TUND clears this error flag.

### Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one Master is trying to communicate simultaneously (a Multi-Master collision) in SPI mode. The mode fault is detected when the enabled Master's  $\overline{SS}$  input pin is asserted. For this to happen the Control and Mode registers must be configured with MMEN = 1, SSIO = 0 ( $\overline{SS}$  is an input), and  $\overline{SS}$  input = 0. A mode fault sets the COL bit in the ESPI Status register to 1. Writing a 1 to COL clears this error flag.

### Receive Overrun

A receive overrun error occurs when a transfer completes and the RDRNE bit is still set from the previous transfer. A receive overrun sets the ROVR bit in the ESPI Status register to 1. Writing a 1 to ROVR clears this error flag. The receive data register is not overwritten and will contain the data from the transfer which initially set the RDRNE bit. Subsequent received data is lost until the RDRNE bit is cleared.

In SPI MASTER mode, a receive overrun will not occur. Instead, the SCK will be paused until software responds to the previous RDRNE/TDRE requests.

### SLAVE Mode Abort

In SLAVE mode of operation, if the  $\overline{SS}$  pin deasserts before all bits in a character have been transferred, the transaction is aborted. When this condition occurs the ABT bit is set in the ESPI Status register. A Slave abort error resets the Slave control logic to idle state.

A Slave abort error is also asserted in SLAVE mode, if BRGCTL = 1 and a baud rate generator timeout occurs. When BRGCTL = 1 in Slave mode, the baud rate generator

functions as a Watchdog Timer monitoring the SCK signal. The BRG counter is reloaded every time a transition on SCK occurs while  $\overline{SS}$  is asserted. The Baud Rate Reload registers must be programmed with a value longer than the expected time between the  $\overline{SS}$  assertion and the first SCK edge, between SCK transitions while  $\overline{SS}$  is asserted, and between the last SCK edge and  $\overline{SS}$  deassertion. A timeout indicates the Master is stalled or disabled. Writing a 1 to ABT clears this error flag.

## ESPI Interrupts

ESPI has a single interrupt output which is asserted when any of the TDRE, TUND, COL, ABT, ROVR or RDRNE bits are set in the ESPI Status register. The interrupt is a pulse which is generated when any one of the source bits initially sets. The TDRE and RDRNE interrupts may be enabled/disabled via the Data Interrupt Request Enable (DIRQE) bit of the ESPI Control register.

A transmit interrupt is asserted by the TDRE status bit when the ESPI block is enabled and the DIRQE bit is set. The TDRE bit in the Status register is cleared automatically when the Data register is written or the ESPI block is disabled. Once the Data register is loaded into the shift register to start a new transfer, the TDRE bit will be set again, causing a new transmit interrupt. In SLAVE mode, if information is being received but not transmitted the transmit interrupts may be eliminated by selecting Receive Only mode (ESPIEN1,0 = 01). A Master cannot operate in Receive Only mode since a write to the ESPI (Transmit) Data register is still required to initiate the transfer of a character even if information is being received but not transmitted by the software application.

A receive interrupt is generated by the RDRNE status bit when the ESPI block is enabled, the DIRQE bit is set and a character transfer completes. At the end of the character transfer, the contents of the shift register are transferred into the data register, causing the RDRNE bit to assert. The RDRNE bit is cleared when the Data Buffer is read as empty. If information is being transmitted but not received by the software application, the receive interrupt can be eliminated by selecting Transmit Only mode (ESPIEN1,0 = 10) in either MASTER or SLAVE modes. When information is being sent and received under interrupt control, RDRNE and TDRE will both assert simultaneously at the end of a character transfer. Since the new receive data is in the Data register, the receive interrupt must be serviced before the transmit interrupt.

ESPI error interrupts occur if any of the TUND, COL, ABT, and ROVR bits in the ESPI Status register are set. These bits are cleared by writing a 1. If the ESPI is disabled (ESPIEN1, 0 = 00), an ESPI interrupt can be generated by a Baud Rate Generator timeout. This timer function must be enabled by setting the BRGCTL bit in the ESPICL register. This timer interrupt does not set any of the bits of the ESPI Status register.

## ESPI Baud Rate Generator

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

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

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

When the ESPI is disabled, the Baud Rate Generator can function as a basic 16-bit timer with interrupt on timeout. To configure the Baud Rate Generator as a timer with interrupt on timeout, follow the steps below:

1. Disable the ESPI by clearing the `ESPIEN1`, 0 bits in the ESPI Control register.
2. Load the desired 16-bit count value into the ESPI Baud Rate High and Low Byte registers.
3. Enable the Baud Rate Generator timer function and associated interrupt by setting the `BRGCTL` bit in the ESPI Control register to 1.

## ESPI Control Register Definitions

### ESPI Data Register

The ESPI Data register (see [Table 110](#) on page 206) addresses both the outgoing transmit data register and the incoming receive data register. Reads from the ESPI Data register return the contents of the receive data register. The receive data register is updated with the contents of the shift register at the end of each transfer. Writes to the ESPI Data register load the transmit data register unless `TDRE = 0`. Data is shifted out starting with bit 7. The last bit received resides in bit position 0.

With the ESPI configured as a Master, writing a data byte to this register initiates the data transmission. With the ESPI 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 `TDRE = 0`, writes to this register are ignored.

When the character length is less than 8 bits (as set by the `NUMBITS` field in the ESPI Mode register), the transmit character must be left justified in the ESPI 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 ESPI is configured for 4-bit characters, the transmit characters must be written to `ESPIDATA[7:4]` and the received characters are read from `ESPIDATA[3:0]`.

Table 110. ESPI Data Register (ESPIDATA)

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

**DATA—Data**

Transmit and/or receive data. Writes to the ESPIDATA register load the shift register.  
Reads from the ESPIDATA register return the value of the receive data register.

## ESPI Transmit Data Command and Receive Data Buffer Control Register

The ESPI Transmit Data Command and Receive Data Buffer Control register (Table 111) provides control of the SS pin when it is configured as an output (MASTER mode), clear receive data buffer function and flag. The CRDR, TEOF, and SSV bits can be controlled by a bus write to this register.

Table 111. ESPI Transmit Data Command and Receive Data Buffer Control Register (ESPITDCR)

BITS	7	6	5	4	3	2	1	0
FIELD	CRDR	RDFLAG					TEOF	SSV
RESET	0	00		0	0	0	0	0
R/W	R/W	R		R	R	R	R/W	R/W
ADDR	F61H							

**CRDR—Clear Receive Data Register**

Writing 1 to this bit is used to clear all data in receive data buffer.

**RDFLAG—Receive Data Buffer FLAG**

This bit is used to indicate how many bytes stored in receive buffer.

00 = 0 or 4 bytes (see RDRNE in ESPI Status Register).

01 = 1 byte.

02 = 2 bytes.

03 = 3 bytes.

**TEOF—Transmit End of Frame**

This bit is used in MASTER mode to indicate that the data in the transmit data register is the last byte of the transfer or frame. When the last byte has been sent  $\overline{SS}$  (and SSV) will change state and TEOF will automatically clear.

0 = The data in the transmit data register is not the last character in the message.

1 = The data in the transmit data register is the last character in the message.

**SSV—Slave Select Value**

When SSIO = 1, writes to this register will control the value output on the  $\overline{SS}$  pin. For more details, see SSMD field of the [ESPI Mode Register](#) on page 209.

**ESPI Control Register**

The ESPI Control register (see [Table 112](#)) configures the ESPI for transmit and receive operations.

**Table 112. ESPI Control Register (ESPICTL)**

BITS	7	6	5	4	3	2	1	0
FIELD	DIRQE	ESPIEN1	BRGCTL	PHASE	CLKPOL	WOR	MMEN	ESPIEN0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F62H							

**DIRQE—Data Interrupt Request Enable**

This bit is used to disable or enable data (TDRE and RDRNE) interrupts. Disabling the data interrupts is needed to control data transfer by polling. Error interrupts are not disabled. To block all ESPI interrupt sources, clear the ESPI interrupt enable bit in the Interrupt Controller.

0 = TDRE and RDRNE assertions do not cause an interrupt.

Use this setting if controlling data transfer by software polling of TDRE and RDRNE. The TUND, COL, ABT, and ROVR bits will cause an interrupt.

1 = TDRE and RDRNE assertions will cause an interrupt.

TUND, COL, ABT, and ROVR will also cause interrupts. Use this setting when controlling data transfer via interrupt handlers.

**ESPIEN1, ESPIEN0—ESPI Enable and Direction Control**

00 = ESPI block is disabled.

BRG may be used as a general purpose timer by setting BRGCTL = 1.

01 = Receive Only Mode.

Use this setting in SLAVE mode if software application is receiving data but not sending. TDRE will not assert. Transmitted data will be all 1's. Not valid in MASTER mode since Master must source data to drive the transfer.

10 = Transmit Only Mode

Use this setting in MASTER or SLAVE mode when the software application is sending data but not receiving. RDRNE will not assert.

11 = Transmit/Receive Mode

Use this setting if the software application is both sending and receiving information. Both TDRE and RDRNE will be active.

#### **BRGCTL—Baud Rate Generator Control**

The function of this bit depends upon ESPIEN1,0. When ESPIEN1,0 = 00, this bit allows enabling the BRG to provide periodic interrupts.

##### **If the ESPI is disabled**

0 = The Baud Rate Generator timer function is disabled.

Reading the Baud Rate High and Low registers returns the BRG Reload value.

1 = The Baud Rate Generator timer function and timeout interrupt is enabled.

Reading the Baud Rate High and Low registers returns the BRG Counter value.

##### **If the ESPI is enabled**

0 = Reading the Baud Rate High and Low registers returns the BRG Reload value.

If MMEN = 1, the BRG is enabled to generate SCK. If MMEN = 0, the BRG is disabled.

1 = Reading the Baud Rate High and Low registers returns the BRG Counter value.

If MMEN = 1, the BRG is enabled to generate SCK. If MMEN = 0 the BRG is enabled to provide a Slave SCK timeout. See the Slave Abort error description.



**Caution:** *If reading the counter one byte at a time while the BRG is counting keep in mind that the values will not be in sync. It is recommended to read the counter using word (2-byte) reads.*

#### **PHASE—Phase Select**

Sets the phase relationship of the data to the clock. For more information on operation of the PHASE bit, see [ESPI Clock Phase and Polarity Control](#) on page 195.

#### **CLKPOL—Clock Polarity**

0 = SCK idles Low (0).

1 = SCK idles High (1).

#### **WOR—Wire-OR (Open-Drain) Mode Enabled**

0 = ESPI signal pins not configured for open-drain.

1 = All four ESPI signal pins (SCK,  $\overline{SS}$ , MISO, and MOSI) configured for open-drain function. This setting is typically used for multi-Master and/or Multi-Slave configurations.

#### **MMEN—ESPI Master Mode Enable**

This bit controls the data I/O pin selection and SCK direction

0 = Data out on MISO, data in on MOSI (used in SPI SLAVE mode), SCK is an input.

1 = Data out on MOSI, data in on MISO (used in SPI MASTER mode), SCK is an output.

## ESPI Mode Register

The ESPI Mode register (see [Table 113](#)) configures the character bit width and mode of the ESPI I/O pins.

**Table 113. ESPI Mode Register (ESPIMODE)**

BITS	7	6	5	4	3	2	1	0
FIELD	SSMD			NUMBITS[2:0]			SSIO	SSPO
RESET	000			0	0	0	0	0
R/W	R/W			R/W	R/W	R/W	R/W	R/W
ADDR	F63H							

### SSMD—Slave Select Mode

This field selects the behavior of  $\overline{SS}$  as a framing signal. For a detailed description of these modes, see [Slave Select](#) on page 194.

#### 000 = SPI Mode

When SSIO = 1, the  $\overline{SS}$  pin is driven directly from the SSV bit in the Transmit Data Command register. The Master software should set SSV (or a GPIO output if the  $\overline{SS}$  pin is not connected to the desired Slave) to the asserted state prior to or on the same clock cycle that the transmit data register is written with the initial byte. At the end of a frame (after the last RDRNE event), SSV will be automatically deasserted by hardware. In this mode, SCK is active only for data transfer (one clock cycle per bit transferred).

#### 001 = Loopback Mode

When ESPI is configured as Master (MMEN = 1), the outputs are deasserted and data is looped from shift register out to shift register in. When ESPI is configured as a Slave (MMEN = 0) and  $\overline{SS}$  asserts, MISO (Slave output) is tied to MOSI (Slave input) to provide an asynchronous remote loop back (echo) function.

#### 010 = I2S Mode (Synchronous Framing with SSV)

In this mode, the value from SSV will be output by the Master on the  $\overline{SS}$  pin with one SCK period before the data and will remain in that state until the start of the next frame. Typically this mode is used to send back to back frames with  $\overline{SS}$  alternating on each frame. A frame boundary is indicated in the Master when SSV changes. A frame boundary is detected in the Slave by  $\overline{SS}$  changing state. The  $\overline{SS}$  framing signal will lead the frame by one SCK period. In this mode SCK will run continuously, starting with the initial  $\overline{SS}$  assertion. Frames will run back-to-back as long as software continues to provide data. An example of this mode is the I<sup>2</sup>S protocol (Inter IC Sound) which is used to carry left and right channel audio data with the  $\overline{SS}$  signal indicating which channel is being sent. In SLAVE mode, the change in state of  $\overline{SS}$  (Low to High or High to Low) triggers the start of a transaction on the next SCK cycle.

#### NUMBITS[2:0]—Number of Data Bits Per Character to Transfer

This field contains the number of bits to shift for each character transfer. For information on valid bit positions when the character length is less than 8 bits, see description of [ESPI Data Register](#) on page 205.

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

This bit controls the direction of the  $\overline{SS}$  pin. In single MASTER mode, SSIO is set to 1 unless a separate GPIO pin is being used to provide the  $\overline{SS}$  output function. In the SPI Slave or multi-Master configuration, SSIO is set to 0.

0 =  $\overline{SS}$  pin configured as an input (SPI Slave and multi-Master modes).

1 =  $\overline{SS}$  pin configured as an output (SPI single Master mode).

#### SSPO—Slave Select Polarity

This bit controls the polarity of the  $\overline{SS}$  pin.

0 =  $\overline{SS}$  is active Low. (SSV = 1 corresponds to  $\overline{SS}$  = 0).

1 =  $\overline{SS}$  is active High. (SSV = 1 corresponds to  $\overline{SS}$  = 1).

## ESPI Status Register

The ESPI Status register ([Table 114](#)) indicates the current state of the ESPI. All bits revert to their Reset state if the ESPI is disabled.

**Table 114. ESPI Status Register (ESPISTAT)**

BITS	7	6	5	4	3	2	1	0
FIELD	TDRE	TUND	COL	ABT	ROVR	RDRNE	TFST	SLAS
RESET	1	0	0	0	0	0	0	1
R/W	R	R/W*	R/W*	R/W*	R/W*	R	R	R
ADDR	F64H							
Note: R/W* = Read access. Write a 1 to clear the bit to 0.								

#### TDRE—Transmit Data Register Empty

0 = Transmit data register is full or ESPI is disabled.

1 = Transmit data register is empty. A write to the ESPI (Transmit) Data register clears this bit.

**TUND—Transmit Underrun**

0 = A Transmit Underrun error has not occurred.

1 = A Transmit Underrun error has occurred.

**COL—Collision**

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

1 = A multi-Master collision (mode fault) has occurred.

**ABT—Slave mode transaction abort**

This bit is set if the ESPI 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 ESPIMODE register. This bit can also be set in SLAVE mode by an SCK monitor timeout (MMEN = 0, BRGCTL = 1).

0 = A SLAVE mode transaction abort has not occurred.

1 = A SLAVE mode transaction abort has occurred.

**ROVR—Receive Overrun**

0 = A Receive Overrun error has not occurred.

1 = A Receive Overrun error has occurred.

**RDRNE—Receive Data Register Not Empty**

0 = Receive Data register is empty.

1 = Receive Data register is not empty.

**TFST—Transfer Status**

0 = No data transfer is currently in progress.

1 = Data transfer is currently in progress.

**SLAS—Slave Select**

Reading this bit returns the current value of the  $\overline{SS}$  pin.

0 = The  $\overline{SS}$  pin input is Low.

1 = The  $\overline{SS}$  pin input is High.

## ESPI State Register

The ESPI State register (Table 115) lets you observe the ESPI clock, data and internal state.

**Table 115. ESPI State Register (ESPISTATE)**

BITS	7	6	5	4	3	2	1	0
FIELD	SCKI	SDI	ESPISTATE					
RESET	0	0	0					
R/W	R	R	R					
ADDR	F65H							

**SCKI—Serial Clock Input**

This bit reflects the state of the serial clock pin.

0 = The SCK input pin is Low.

1 = The SCK input pin is High.

**SDI—Serial Data Input**

This bit reflects the state of the serial data input (MOSI or MISO depending on the MMEN bit).

0 = The serial data input pin is Low.

1 = The serial data input pin is High.

**ESPISTATE—ESPI State Machine**

Indicates the current state of the internal ESPI State Machine. This information is intended for manufacturing test. The state values may change in future hardware revisions and are not intended to be used by a software driver. [Table 116](#) defines the valid states.

**Table 116. ESPISTATE values**

<b>ESPISTATE Value</b>	<b>Description</b>
00_0000	Idle
00_0001	Slave Wait For SCK
01_0001	Master Ready
10_1110	Bit 7 Receive
10_1111	Bit 7 Transmit
10_1100	Bit 6 Receive
10_1101	Bit 6 Transmit
10_1010	Bit 5 Receive
10_1011	Bit 5 Transmit
10_1000	Bit 4 Receive
10_1001	Bit 4 Transmit
10_0110	Bit 3 Receive
10_0111	Bit 3 Transmit
10_0100	Bit 2 Receive
10_0101	Bit 2 Transmit
10_0010	Bit 1 Receive
10_0011	Bit 1 Transmit
10_0000	Bit 0 Receive
10_0001	Bit 0 Transmit

## ESPI Baud Rate High and Low Byte Registers

The ESPI Baud Rate High and Low Byte registers (see [Table 117](#) and [Table 118](#)) combine to form a 16-bit reload value, BRG[15:0], for the ESPI Baud Rate Generator. The ESPI baud rate is calculated using the following equation:

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

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

**Table 117. ESPI Baud Rate High Byte Register (ESPIBRH)**

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

**BRH = ESPI Baud Rate High Byte**

Most significant byte, BRG[15:8], of the ESPI Baud Rate Generator's Reload value.

**Table 118. ESPI Baud Rate Low Byte Register (ESPIBRL)**

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

**BRL = ESPI Baud Rate Low Byte**

Least significant byte, BRG[7:0], of the ESPI Baud Rate Generator's Reload value.



# I<sup>2</sup>C Master/Slave Controller

The I<sup>2</sup>C Master/Slave Controller ensures that the Z8 Encore! XP F1680 Series devices are bus-compatible with the I<sup>2</sup>C protocol. The I<sup>2</sup>C bus consists of the serial data signal (SDA) and a serial clock signal (SCL) bidirectional lines. I<sup>2</sup>C controller includes the following features:

- Operates in MASTER/SLAVE or SLAVE ONLY modes.
- Supports arbitration in a multimaster environment (MASTER/SLAVE mode).
- Supports data rates up to 400 Kbps.
- 7-bit or 10-bit slave address recognition (interrupt only on address match).
- Optional general call address recognition.
- Optional digital filter on receive SDA, SCL lines.
- Optional interactive receive mode allows software interpretation of each received address and/or data byte before acknowledging.
- Unrestricted number of data bytes per transfer.
- Baud Rate Generator can be used as a general-purpose timer with an interrupt, if the I<sup>2</sup>C controller is disabled.

## Architecture

[Figure 42](#) on page 216 illustrates the architecture of the I<sup>2</sup>C controller.

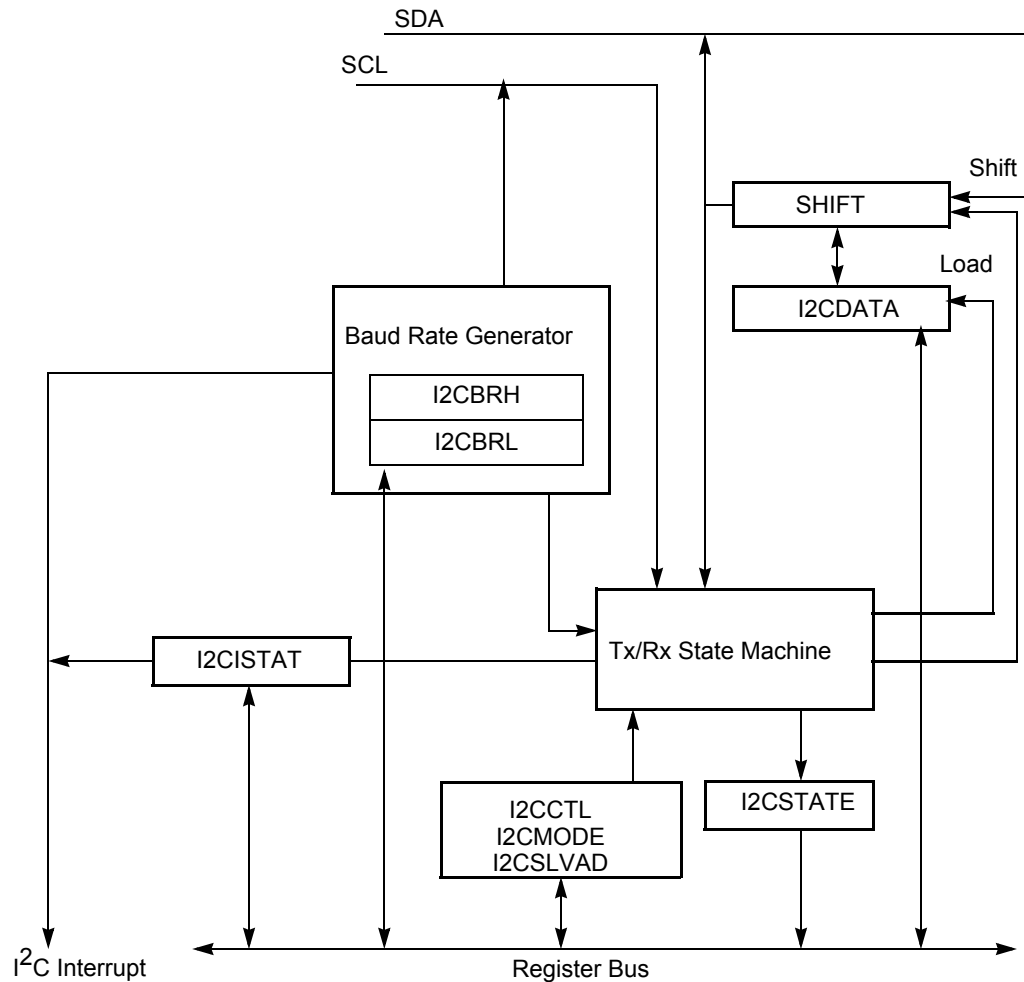


Figure 42. I²C Controller Block Diagram

## I²C Master/Slave Controller Registers

Table 119 summarizes the I²C Master/Slave controller's software-accessible registers.

Table 119. I²C Master/Slave Controller Registers

Name	Abbreviation	Description
I2C Data	I2CDATA	Transmit/receive data register.
I2C Interrupt Status	I2CISTAT	Interrupt status register.

**Table 119. I<sup>2</sup>C Master/Slave Controller Registers (Continued)**

Name	Abbreviation	Description
I <sup>2</sup> C Control	I2CCTL	Control register—basic control functions.
I <sup>2</sup> C Baud Rate High	I2CBRH	High byte of baud rate generator initialization value.
I <sup>2</sup> C Baud Rate Low	I2CBRL	Low byte of baud rate generator initialization value.
I <sup>2</sup> C State	I2CSTATE	State register.
I <sup>2</sup> C Mode	I2CMODE	Selects MASTER or SLAVE modes, 7-bit or 10-bit addressing; configure address recognition, define slave address bits [9:8].
I <sup>2</sup> C Slave Address	I2CSLVAD	Defines slave address bits [7:0].

## Operation

The I<sup>2</sup>C Master/Slave Controller operates in MASTER/SLAVE mode, SLAVE ONLY mode, or with master arbitration. In MASTER/SLAVE mode, it can be used as the only Master on the bus or as one of the several masters on the bus, with arbitration. In a Multi-Master environment, the controller switches from MASTER to SLAVE mode on losing arbitration.

Though slave operation is fully supported in MASTER/SLAVE mode, if a device is intended to operate only as a slave, then SLAVE ONLY mode can be selected. In SLAVE ONLY mode, the device will not initiate a transaction, even if the software inadvertently sets the START bit.

## SDA and SCL Signals

The I<sup>2</sup>C circuit sends all addresses, Data, and Acknowledge signals over the SDA line, with most-significant bit first. SCL is the clock for the I<sup>2</sup>C bus. 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 is responsible for driving the SCL clock signal. During the Low period of the clock, a slave can hold the SCL signal Low to suspend the transaction if it is not ready to proceed. 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 I<sup>2</sup>C master continues the transaction. All data is transferred in bytes; there is no limit to the amount of data transferred in one operation. When transmitting address, data, or an Acknowledge, the SDA signal changes in the middle of the Low period of SCL. When receiving address, Data, or an Acknowledge; the SDA signal is sampled in the middle of the High period of SCL.

A low-pass digital filter can be applied to the SDA and SCL receive signals by setting the Filter Enable (**FILTEN**) bit in the I<sup>2</sup>C Control Register. When the filter is enabled, any glitch that is less than a system clock period in width will be rejected. This filter should be enabled when running in I<sup>2</sup>C FAST mode (400 Kbps), and can also be used at lower data rates.

## I<sup>2</sup>C Interrupts

The I<sup>2</sup>C controller contains multiple interrupt sources that are combined into one interrupt request signal to the interrupt controller. If the I<sup>2</sup>C controller is enabled, the source of the interrupt is determined by which bits are set in the I2CISTAT Register. If the I<sup>2</sup>C controller is disabled, the BRG controller is used to generate general-purpose timer interrupts.

Each interrupt source, other than the baud rate generator interrupt, features an associated bit in the I2CISTAT Register that clears automatically when software reads the register or performs another task, such as reading/writing the data register.

### Transmit Interrupts

Transmit interrupts (**TDRE** bit = 1 in I2CISTAT) occur under the following conditions, both of which must be true:

- The transmit data register is empty and the **TXI** bit = 1 in the I<sup>2</sup>C Control Register.
- The I<sup>2</sup>C controller is enabled with one of the following:
  - The first bit of a 10-bit address is shifted out.
  - The first bit of the final byte of an address is shifted out and the **RD** bit is deasserted.
  - The first bit of a data byte is shifted out.

Writing to the I<sup>2</sup>C Data Register always clears the **TRDE** bit to 0.

### Receive Interrupts

Receive interrupts (**RDRF** bit = 1 in I2CISTAT) occur when a byte of data has been received by the I<sup>2</sup>C controller. The **RDRF** bit is cleared by reading from the I<sup>2</sup>C Data Register. If the **RDRF** interrupt is not serviced prior to the completion of the next Receive byte, the I<sup>2</sup>C controller holds SCL Low during the final data bit of the next byte until **RDRF** is cleared, to prevent receive overruns. A receive interrupt does not occur when a Slave receives an address byte or for data bytes following a slave address that do not match. An exception is if the Interactive Receive Mode (**IRM**) bit is set in the I2CMODE Register, in which case Receive interrupts occur for all Receive address and data bytes in SLAVE mode.

### Slave Address Match Interrupts

Slave address match interrupts (**SAM** bit = 1 in I2CISTAT) occur when the I<sup>2</sup>C controller is in SLAVE mode and an address received matches the unique slave

address. The General Call Address (0000\_0000) and STARTBYTE (0000\_0001) are recognized if the GCE bit = 1 in the I2CMODE Register. The software checks the RD bit in the I2CISTAT Register to determine if the transaction is a Read or Write transaction. The General Call Address and STARTBYTE address are also distinguished by the RD bit. The General Call Address (GCA) bit of the I2CISTAT Register indicates whether the address match occurred on the unique slave address or the General Call/STARTBYTE address. The SAM bit clears automatically when the I2CISTAT Register is read.

If configured via the MODE[1:0] field of the I<sup>2</sup>C Mode Register for 7-bit slave addressing, the most significant 7 bits of the first byte of the transaction are compared against the SLA[6:0] bits of the Slave Address Register. If configured for 10-bit slave addressing, the first byte of the transaction is compared against {11110, SLA[9:8], R/W} and the second byte is compared against SLA[7:0].

### Arbitration Lost Interrupts

Arbitration Lost interrupts (ARBLST bit = 1 in I2CISTAT) occur when the I<sup>2</sup>C controller is in MASTER mode and loses arbitration (outputs 1 on SDA and receives 0 on SDA). The I<sup>2</sup>C controller switches to SLAVE mode when this instance occurs. This bit clears automatically when the I2CISTAT Register is read.

### Stop/Restart Interrupts

A Stop/Restart event interrupt (SPRS bit = 1 in I2CISTAT) occurs when the I<sup>2</sup>C controller is in SLAVE mode and a STOP or RESTART condition is received, indicating the end of the transaction. The RSTR bit in the I2C State Register indicates whether the bit is set due to a STOP or RESTART condition. When a restart occurs, a new transaction by the same master is expected to follow. This bit is cleared automatically when the I2CISTAT Register is read. The Stop/Restart interrupt occurs only on a selected (address match) slave.

### Not Acknowledge Interrupts

Not Acknowledge interrupts (NCKI bit = 1 in I2CISTAT) occur in MASTER mode when Not Acknowledge is received or sent by the I<sup>2</sup>C controller and the START or STOP bit is not set in the I<sup>2</sup>C Control Register. In MASTER mode, the Not Acknowledge interrupt clears by setting the START or STOP bit. When this interrupt occurs in MASTER mode, the I<sup>2</sup>C controller waits till it is cleared before performing any action. In SLAVE mode, the Not Acknowledge interrupt occurs when a Not Acknowledge is received in response to data sent. The NCKI bit clears in SLAVE mode when software reads the I2CISTAT Register.

### General Purpose Timer Interrupt from Baud Rate Generator

If the I<sup>2</sup>C 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 baud rate

generator (BRG) counts down to 1. The baud rate generator reloads and continues counting, providing a periodic interrupt. None of the bits in the I2CISTAT Register are set, allowing the BRG in the I<sup>2</sup>C controller to be used as a general-purpose timer when the I<sup>2</sup>C controller is disabled.

## Start and Stop Conditions

The Master generates the START and STOP conditions to start or end a transaction. To start a transaction, the I<sup>2</sup>C controller generates a START condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the I<sup>2</sup>C controller generates a STOP condition by creating a Low-to-High transition of the SDA signal while the SCL signal is High. These START and STOP events occur when the START and STOP bits in the I<sup>2</sup>C Control Register are written by software to begin or end a transaction. Any byte transfer currently under way including the Acknowledge phase finishes before the START or STOP condition occurs.

## Software Control of I<sup>2</sup>C Transactions

The I<sup>2</sup>C controller is configured via the I<sup>2</sup>C Control and I<sup>2</sup>C Mode registers. The MODE[1:0] field of the I<sup>2</sup>C Mode Register allows the configuration of the I<sup>2</sup>C controller for MASTER/SLAVE or SLAVE ONLY mode and configures the slave for 7-bit or 10-bit addressing recognition.

MASTER/SLAVE mode can be used for:

- MASTER ONLY operation in a Single Master/One or More Slave I<sup>2</sup>C system.
- MASTER/SLAVE in a Multimaster/multislave I<sup>2</sup>C system.
- SLAVE ONLY operation in an I<sup>2</sup>C system.

In SLAVE ONLY mode, the START bit of the I<sup>2</sup>C Control Register is ignored (software cannot initiate a master transaction by accident), and operation to SLAVE ONLY mode is restricted thereby preventing accidental operation in MASTER mode. The software controls I<sup>2</sup>C transactions by enabling the I<sup>2</sup>C controller interrupt in the interrupt controller or by polling the I<sup>2</sup>C Status Register.

To use interrupts, the I<sup>2</sup>C interrupt must be enabled in the interrupt controller and followed by executing an EI instruction. The TXI bit in the I<sup>2</sup>C Control Register must be set to enable transmit interrupts. An I<sup>2</sup>C interrupt service routine then checks the I<sup>2</sup>C Status Register to determine the cause of the interrupt.

To control transactions by polling, the TDRE, RDRE, SAM, ARBLST, SPRS, and NCKI interrupt bits in the I<sup>2</sup>C Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

## Master Transactions

The following sections describe Master Read and Write transactions to both 7-bit and 10-bit slaves.

### Master Arbitration

If a Master loses arbitration during the address byte it releases the SDA line, switches to SLAVE mode and monitors the address to determine if it is selected as a Slave. If a Master loses arbitration during the transmission of a data byte, it releases the SDA line and waits for the next STOP or START condition.

The Master detects a loss of arbitration when a 1 is transmitted but a 0 is received from the bus in the same bit-time. This loss occurs if more than one Master is simultaneously accessing the bus. Loss of arbitration occurs during the address phase (two or more Masters accessing different slaves) or during the data phase, when the masters are attempting to Write different data to the same Slave.

When a Master loses arbitration, the software is informed by means of the Arbitration Lost interrupt. The software can repeat the same transaction at a later time.

A special case can occur when a Slave transaction starts just before the software attempts to start a new master transaction by setting the START bit. In this case, the state machine enters its Slave states before the START bit is set, and as a result the I<sup>2</sup>C controller will not arbitrate. If a Slave address match occurs and the I<sup>2</sup>C controller receives/transmits data, the START bit is cleared and an Arbitration Lost interrupt is asserted. The software can minimize the chance of this instance occurring by checking the BUSY bit in the I2CSTATE Register before initiating a Master transaction. If a slave address match does not occur, the Arbitration Lost interrupt will not occur, and the START bit will not be cleared. The I<sup>2</sup>C controller will initiate the master transaction after the I<sup>2</sup>C bus is no longer busy.

### Master Address-Only Transactions

It is sometimes preferable to perform an address-only transaction to determine if a particular slave device is able to respond. This transaction can be performed by monitoring the ACKV bit in the I2CSTATE Register after the address has been written to the I2CDATA Register and the START bit has been set. After the ACKV bit is set, the ACK bit in the I2CSTATE Register determines if the slave is able to communicate. The STOP bit must be set in the I2CCTL Register to terminate the transaction without transferring data. For a 10-bit slave address, if the first address byte is acknowledged, the second address byte should also be sent to determine if the preferred Slave is responding.

Another approach is to set both the STOP and START bits (for sending a 7-bit address). After both bits have been cleared (7-bit address has been sent and transaction is complete), the ACK bit can be read to determine if the Slave has acknowledged. For a 10-bit Slave, set

the `STOP` bit after the second `TDRE` interrupt (which indicates that the second address byte is being sent).

Master Transaction Diagrams

In the following transaction diagrams, the shaded regions indicate the data that is transferred from the Master to the Slave, and the unshaded regions indicate the data that is transferred from the Slave to the Master. The transaction field labels are defined as follows:

- S        Start
- W        Write
- A        Acknowledge
- $\overline{A}$     Not Acknowledge
- P        Stop

Master Write Transaction with a 7-Bit Address

Figure 43 illustrates the data transfer format from a Master to a 7-bit addressed slave.

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

Figure 43. Data Transfer Format—Master Write Transaction with a 7-Bit Address

The procedure for a Master transmit operation to a 7-bit addressed slave is as follows:

1. The software initializes the `MODE` field in the `I2C Mode Register` for `MASTER/SLAVE` mode with either a 7-bit or 10-bit slave address. The `MODE` field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the `IEN` bit in the `I2C Control Register`.
2. The 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. The software responds to the `TDRE` bit by writing a 7-bit slave address plus the Write bit (which is cleared to 0) to the `I2C Data Register`.
5. The software sets the `START` bit of the `I2C Control Register`.
6. The `I2C` controller sends a `START` condition to the `I2C` slave.

7. The I<sup>2</sup>C controller loads the I<sup>2</sup>C Shift Register with the contents of the I<sup>2</sup>C Data Register.
8. After one bit of the address has been shifted out by the SDA signal, the transmit interrupt asserts.
9. The software responds by writing the transmit data into the I<sup>2</sup>C Data Register.
10. The I<sup>2</sup>C controller shifts the remainder of the address and the Write bit out via the SDA signal.
11. The I<sup>2</sup>C slave sends an Acknowledge (by pulling the SDA signal Low) during the next high period of SCL. The I<sup>2</sup>C controller sets the ACK bit in the I<sup>2</sup>C Status Register.

If the slave does not acknowledge the address byte, the I<sup>2</sup>C controller sets the NCKI bit in the I<sup>2</sup>C Status Register, sets the ACKV bit, and clears the ACK bit in the I<sup>2</sup>C State Register. The software responds to the Not Acknowledge interrupt by setting the STOP bit and clearing the TXI bit. The I<sup>2</sup>C controller flushes the Transmit Data Register, sends a STOP condition on the bus, and clears the STOP and NCKI bits. The transaction is complete, and the following steps can be ignored.

12. The I<sup>2</sup>C controller loads the contents of the I<sup>2</sup>C Shift Register with the contents of the I<sup>2</sup>C Data Register.
13. The I<sup>2</sup>C controller shifts the data out via the SDA signal. After the first bit is sent, the transmit interrupt asserts.
14. If more bytes remain to be sent, return to [Step 9](#).
15. When there is no more data to be sent, the software responds by setting the STOP bit of the I<sup>2</sup>C Control Register (or the START bit to initiate a new transaction).
16. If no additional transaction is queued by the master, the software can clear the TXI bit of the I<sup>2</sup>C Control Register.
17. The I<sup>2</sup>C controller completes transmission of the data on the SDA signal.
18. The I<sup>2</sup>C controller sends a STOP condition to the I<sup>2</sup>C bus.

► **Note:** *If the slave terminates the transaction early by responding with a Not Acknowledge during the transfer, the I<sup>2</sup>C controller asserts the NCKI interrupt and halts. The software must terminate the transaction by setting either the STOP bit (end transaction) or the START bit (end this transaction, start a new one). In this case, it is not necessary for software to set the FLUSH bit of the I2CCTL Register to flush the data that was previously written but not transmitted. The I<sup>2</sup>C controller hardware automatically flushes transmit data in the not acknowledge case.*

### Master Write Transaction with a 10-Bit Address

[Figure 44](#) illustrates the data transfer format from a Master to a 10-bit addressed slave.

S	Slave Address 1st Byte	W = 0	A	Slave Address 2nd Byte	A	Data	A	Data	A/ $\bar{A}$	F/S
---	---------------------------	-------	---	---------------------------	---	------	---	------	--------------	-----

**Figure 44. Data Transfer Format—Master Write Transaction with a 10-Bit Address**

The first 7 bits transmitted in the first byte are 11110XX. The 2 XX bits are the two most significant bits of the 10-bit address. The lowest bit of the first byte transferred is the Read/Write control bit (which is cleared to 0). The transmit operation is performed in the same manner as 7-bit addressing.

The procedure for a master transmit operation to a 10-bit addressed slave is as follows:

1. The software initializes the `MODE` field in the I<sup>2</sup>C Mode Register for MASTER/SLAVE mode with 7- or 10-bit addressing (the I<sup>2</sup>C bus protocol allows the mixing of slave address types). The `MODE` field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the `IEN` bit in the I<sup>2</sup>C Control Register.
2. The software asserts the `TXI` bit of the I<sup>2</sup>C Control Register to enable transmit interrupts.
3. The I<sup>2</sup>C interrupt asserts because the I<sup>2</sup>C Data Register is empty.
4. The software responds to the `TDRE` interrupt by writing the first Slave Address byte (11110XX0). The least-significant bit must be 0 for the write operation.
5. The software asserts the `START` bit of the I<sup>2</sup>C Control Register.
6. The I<sup>2</sup>C controller sends a `START` condition to the I<sup>2</sup>C Slave.
7. The I<sup>2</sup>C controller loads the I<sup>2</sup>C Shift Register with the contents of the I<sup>2</sup>C Data Register.
8. After one bit of the address is shifted out by the SDA signal, the transmit interrupt asserts.
9. The software responds by writing the second byte of address into the contents of the I<sup>2</sup>C Data Register.
10. The I<sup>2</sup>C controller shifts the remainder of the first byte of the address and the Write bit out via the SDA signal.
11. The I<sup>2</sup>C slave sends an Acknowledge by pulling the SDA signal Low during the next high period of SCL. The I<sup>2</sup>C controller sets the `ACK` bit in the I<sup>2</sup>C Status Register.

If the slave does not acknowledge the first address byte, the I<sup>2</sup>C controller sets the `NCKI` bit in the I<sup>2</sup>C Status Register, sets the `ACKV` bit, and clears the `ACK` bit in the I<sup>2</sup>C State Register. The software responds to the Not Acknowledge interrupt by setting the `STOP` bit and clearing the `TXI` bit. The I<sup>2</sup>C controller flushes the second address byte from the data register, sends a `STOP` condition on the bus, and clears

the `STOP` and `NCKI` bits. The transaction is complete, and the following steps can be ignored.

12. The I<sup>2</sup>C controller loads the I<sup>2</sup>C Shift Register with the contents of the I<sup>2</sup>C Data Register (2nd address byte).
13. The I<sup>2</sup>C controller shifts the second address byte out via the SDA signal. After the first bit has been sent, the transmit interrupt asserts.
14. The software responds by writing the data to be written out to the I<sup>2</sup>C Control Register.
15. The I<sup>2</sup>C controller shifts out the remainder of the second byte of the slave address (or ensuring data bytes, if looping) via the SDA signal.
16. The I<sup>2</sup>C slave sends an Acknowledge by pulling the SDA signal Low during the next high period of SCL. The I<sup>2</sup>C controller sets the `ACK` bit in the I<sup>2</sup>C Status Register. If the slave does not acknowledge, see the second paragraph of [Step 11](#).
17. The I<sup>2</sup>C controller shifts the data out by the SDA signal. After the first bit is sent, the transmit interrupt asserts.
18. If more bytes remain to be sent, return to [Step 14](#).
19. The software responds by asserting the `STOP` bit of the I<sup>2</sup>C Control Register.
20. The I<sup>2</sup>C controller completes transmission of the data on the SDA signal.
21. The I<sup>2</sup>C controller sends a `STOP` condition to the I<sup>2</sup>C bus.

► **Note:** *If the slave responds with a Not Acknowledge during the transfer, the I<sup>2</sup>C controller asserts the `NCKI` bit, sets the `ACKV` bit, clears the `ACK` bit in the I<sup>2</sup>C State Register, and halts. The software terminates the transaction by setting either the `STOP` bit (end transaction) or the `START` bit (end this transaction, start a new one). The Transmit Data Register is flushed automatically.*

### Master Read Transaction with a 7-Bit Address

[Figure 45](#) illustrates the data transfer format for a Read operation to a 7-bit addressed slave.

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

**Figure 45. Data Transfer Format—Master Read Transaction with a 7-Bit Address**

The procedure for a Master Read operation to a 7-bit addressed slave is as follows:

1. The software initializes the `MODE` field in the I<sup>2</sup>C Mode Register for MASTER/SLAVE mode with 7- or 10-bit addressing (the I<sup>2</sup>C bus protocol allows the mixing of slave address types). The `MODE` field selects the address width for

this mode when addressed as a slave (but not for the remote slave). The software asserts the `IEN` bit in the `I2C Control Register`.

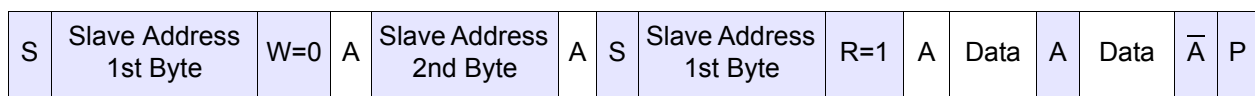
2. The software writes the `I2C Data Register` with a 7-bit slave address, plus the Read bit (which is set to 1).
3. The software asserts the `START` bit of the `I2C Control Register`.
4. If this operation is a single-byte transfer, the software asserts the `NAK` bit of the `I2C Control Register` so that after the first byte of data has been read by the `I2C controller`, a Not Acknowledge instruction is sent to the `I2C slave`.
5. The `I2C controller` sends a `START` condition.
6. The `I2C controller` sends the address and Read bit out via the `SDA` signal.
7. The `I2C slave` acknowledges the address by pulling the `SDA` signal Low during the next high period of `SCL`.

If the slave does not acknowledge the address byte, the `I2C controller` sets the `NCKI` bit in the `I2C Status Register`, sets the `ACKV` bit, and clears the `ACK` bit in the `I2C State Register`. The software responds to the Not Acknowledge interrupt by setting the `STOP` bit and clearing the `TXI` bit. The `I2C controller` flushes the `Transmit Data Register`, sends a `STOP` condition on the bus, and clears the `STOP` and `NCKI` bits. The transaction is complete, and the following steps can be ignored.

8. The `I2C controller` shifts in the first byte of data from the `I2C slave` on the `SDA` signal.
9. The `I2C controller` asserts the receive interrupt.
10. The software responds by reading the `I2C Data Register`. If the next data byte is to be the final byte, the software must set the `NAK` bit of the `I2C Control Register`.
11. The `I2C controller` sends a Not Acknowledge to the `I2C slave` if the next byte is the final byte; otherwise, it sends an Acknowledge.
12. If there are more bytes to transfer, the `I2C controller` returns to [Step 7](#).
13. A NAK interrupt (`NCKI` bit in `I2CISTAT`) is generated by the `I2C controller`.
14. The software responds by setting the `STOP` bit of the `I2C Control Register`.
15. A `STOP` condition is sent to the `I2C slave`.

### Master Read Transaction with a 10-Bit Address

[Figure 46](#) illustrates the read transaction format for a 10-bit addressed Slave.



**Figure 46. Data Transfer Format—Master Read Transaction with a 10-Bit Address**

The first 7 bits transmitted in the first byte are 11110XX. The two XX bits are the two most-significant bits of the 10-bit address. The lowest bit of the first byte transferred is the write control bit.

The data transfer procedure for a Read operation to a 10-bit addressed slave is as follows:

1. The software initializes the `MODE` field in the I<sup>2</sup>C Mode Register for MASTER/SLAVE mode with 7- or 10-bit addressing (the I<sup>2</sup>C bus protocol allows the mixing of slave address types). The `MODE` field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the `IEN` bit in the I<sup>2</sup>C Control Register.
2. The software writes 11110b, followed by the two most-significant address bits and a 0 (write) to the I<sup>2</sup>C Data Register.
3. The software asserts the `START` bit of the I<sup>2</sup>C Control Register.
4. The I<sup>2</sup>C controller sends a `START` condition.
5. The I<sup>2</sup>C controller loads the I<sup>2</sup>C Shift Register with the contents of the I<sup>2</sup>C Data Register.
6. After the first bit has been shifted out, a transmit interrupt is asserted.
7. The software responds by writing the least significant eight bits of address to the I<sup>2</sup>C Data Register.
8. The I<sup>2</sup>C controller completes shifting of the first address byte.
9. The I<sup>2</sup>C slave sends an Acknowledge by pulling the SDA signal Low during the next high period of SCL.  
  
If the slave does not acknowledge the address byte, the I<sup>2</sup>C controller sets the `NCKI` bit in the I<sup>2</sup>C Status Register, sets the `ACKV` bit and clears the `ACK` bit in the I<sup>2</sup>C State Register. The software responds to the Not Acknowledge interrupt by setting the `STOP` bit and clearing the `TXI` bit. The I<sup>2</sup>C controller flushes the Transmit Data Register, sends the `STOP` condition on the bus and clears the `STOP` and `NCKI` bits. The transaction is complete, and the following steps can be ignored.
10. The I<sup>2</sup>C controller loads the I<sup>2</sup>C Shift Register with the contents of the I<sup>2</sup>C Data Register (the lower byte of the 10-bit address).
11. The I<sup>2</sup>C controller shifts out the next eight bits of the address. After the first bit shifts, the I<sup>2</sup>C controller generates a transmit interrupt.
12. The software responds by setting the `START` bit of the I<sup>2</sup>C Control Register to generate a repeated `START` condition.
13. The software writes 11110b, followed by the 2-bit slave address and a 1 (Read) to the I<sup>2</sup>C Data Register.
14. If the user chooses to read only one byte, the software responds by setting the `NAK` bit of the I<sup>2</sup>C Control Register.

15. After the I<sup>2</sup>C controller shifts out the address bits listed in [Step 9](#) (the second address transfer), the I<sup>2</sup>C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.

If the slave does not acknowledge the address byte, the I<sup>2</sup>C controller sets the NCKI bit in the I<sup>2</sup>C Status Register, sets the ACKV bit, and clears the ACK bit in the I<sup>2</sup>C State Register. The software responds to the Not Acknowledge interrupt by setting the STOP bit and clearing the TXI bit. The I<sup>2</sup>C controller flushes the Transmit Data Register, sends the STOP condition on the bus, and clears the STOP and NCKI bits. The transaction is complete, and the following steps can be ignored.

16. The I<sup>2</sup>C controller sends a repeated START condition.
17. The I<sup>2</sup>C controller loads the I<sup>2</sup>C Shift Register with the contents of the I<sup>2</sup>C Data Register (the third address transfer).
18. The I<sup>2</sup>C controller sends 11110b, followed by the two most-significant bits of the slave read address and a 1 (Read).
19. The I<sup>2</sup>C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.
20. The I<sup>2</sup>C controller shifts in a byte of data from the slave.
21. The I<sup>2</sup>C controller asserts the Receive interrupt.
22. The software responds by reading the I<sup>2</sup>C Data Register. If the next data byte is to be the final byte, the software must set the NAK bit of the I<sup>2</sup>C Control Register.
23. The I<sup>2</sup>C controller sends an Acknowledge or Not Acknowledge to the I<sup>2</sup>C Slave, based on the value of the NAK bit.
24. If there are more bytes to transfer, the I<sup>2</sup>C controller returns to [Step 18](#).
25. The I<sup>2</sup>C controller generates a NAK interrupt (the NCKI bit in the I2CISTAT Register).
26. The software responds by setting the STOP bit of the I<sup>2</sup>C Control Register.
27. A STOP condition is sent to the I<sup>2</sup>C Slave.

## Slave Transactions

The following sections describe Read and Write transactions to the I<sup>2</sup>C controller configured for 7-bit and 10-bit Slave modes.

### Slave Address Recognition

The following Slave address recognition options are supported:

### Slave 7-Bit Address Recognition Mode

If  $IRM = 0$  during the address phase and the controller is configured for MASTER/SLAVE or SLAVE 7-bit address mode, the hardware detects a match to the 7-bit slave address defined in the I2CSLVAD Register and generates the slave address match interrupt (the  $SAM$  bit = 1 in the I2CISTAT Register). The I<sup>2</sup>C controller automatically responds during the Acknowledge phase with the value in the  $NAK$  bit of the I2CCTL Register.

### Slave 10-Bit Address Recognition Mode

If  $IRM = 0$  during the address phase and the controller is configured for MASTER/SLAVE or SLAVE 10-bit address mode, the hardware detects a match to the 10-bit slave address defined in the I2CMODE and I2CSLVAD registers and generates the slave address match interrupt (the  $SAM$  bit = 1 in the I2CISTAT Register). The I<sup>2</sup>C controller automatically responds during the Acknowledge phase with the value in the  $NAK$  bit of the I2CCTL Register.

### General Call and Start Byte Address Recognition

If  $GCE = 1$  and  $IRM = 0$  during the address phase, and the controller is configured for MASTER/SLAVE or SLAVE in either 7- or 10-bit address modes, the hardware detects a match to the General Call Address or the START byte and generates the slave address match interrupt. A General Call Address is a 7-bit address of all 0's with the  $R/\overline{W}$  bit = 0. A START byte is a 7-bit address of all 0's with the  $R/\overline{W}$  bit = 1. The  $SAM$  and  $GCA$  bits are set in the I2CISTAT Register. The  $RD$  bit in the I2CISTAT Register distinguishes a General Call Address from a START byte which is cleared to 0 for a General Call Address). For a General Call Address, the I<sup>2</sup>C controller automatically responds during the address acknowledge phase with the value in the  $NAK$  bit of the I2CCTL Register. If the software is set to process the data bytes associated with the  $GCA$  bit, the  $IRM$  bit can optionally be set following the  $SAM$  interrupt to allow the software to examine each received data byte before deciding to set or clear the  $NAK$  bit. A START byte will not be acknowledged—a requirement of the I<sup>2</sup>C specification.

### Software Address Recognition

To disable hardware address recognition, the  $IRM$  bit must be set to 1 prior to the reception of the address byte(s). When  $IRM = 1$ , each received byte generates a receive interrupt ( $RDRF = 1$  in the I2CISTAT Register). The software must examine each byte and determine whether to set or clear the  $NAK$  bit. The slave holds SCL Low during the Acknowledge phase until the software responds by writing to the I2CCTL Register. The value written to the  $NAK$  bit is used by the controller to drive the I<sup>2</sup>C bus, then releasing the SCL. The  $SAM$  and  $GCA$  bits are not set when  $IRM = 1$  during the address phase, but the  $RD$  bit is updated based on the first address byte.

## Slave Transaction Diagrams

In the following transaction diagrams, the shaded regions indicate data transferred from the Master to the Slave, and the unshaded regions indicate the data transferred from the Slave to the Master. The transaction field labels are defined as follows:

S	Start
W	Write
A	Acknowledge
$\bar{A}$	Not Acknowledge
P	Stop

### Slave Receive Transaction with 7-Bit Address

The data transfer format for writing data from a Master to a Slave in 7-bit address mode is illustrated in Figure 47. The procedure that follows describes the I<sup>2</sup>C Master/Slave Controller operating as a slave in 7-bit addressing mode and receiving data from the bus master.

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

**Figure 47. Data Transfer Format—Slave Receive Transaction with 7-Bit Address**

- The software configures the controller for operation as a slave in 7-bit addressing mode, as follows:
  - Initialize the **MODE** field in the I<sup>2</sup>C Mode Register for either SLAVE ONLY mode or MASTER/SLAVE mode with 7-bit addressing.
  - Optionally set the **GCE** bit.
  - Initialize the **SLA[6:0]** bits in the I<sup>2</sup>C Slave Address Register.
  - Set **IEN** = 1 in the I<sup>2</sup>C Control Register. Set **NAK** = 0 in the I<sup>2</sup>C Control Register.
- The bus master initiates a transfer, sending the address byte. In SLAVE mode, the I<sup>2</sup>C controller recognizes its own address and detects that **R/ $\bar{W}$**  bit = 0 (written from the master to the slave). The I<sup>2</sup>C controller acknowledges indicating it is available to accept the transaction. The **SAM** bit in the I2CISTAT Register is set to 1, causing an interrupt. The **RD** bit in the I2CISTAT Register is cleared to 0, indicating a Write to the slave. The I<sup>2</sup>C controller holds the SCL signal Low waiting for the software to load the first data byte.
- The software responds to the interrupt by reading the I2CISTAT Register (which clears the **SAM** bit). After seeing the **SAM** bit to 1, the software checks the **RD** bit. Because **RD** = 0, no immediate action is required until the first byte of data is received.

If software is only able to accept a single byte, it sets the `NAK` bit in the `I2CCTL` Register at this time.

4. The Master detects the Acknowledge and sends the byte of data.
5. The I<sup>2</sup>C controller receives the data byte and responds with Acknowledge or Not Acknowledge depending on the state of the `NAK` bit in the `I2CCTL` Register. The I<sup>2</sup>C controller generates the receive data interrupt by setting the `RDRF` bit in the `I2CISTAT` Register.
6. The software responds by reading the `I2CISTAT` Register, finding the `RDRF` bit = 1 and reading the `I2CDATA` Register clearing the `RDRF` bit. If software can accept only one more data byte it sets the `NAK` bit in the `I2CCTL` Register.
7. The master and slave loops through [Step 4](#) to [Step 6](#) until the master detects a Not Acknowledge instruction or runs out of data to send.
8. The master sends the `STOP` or `RESTART` signal on the bus. Either of these signals can cause the I<sup>2</sup>C controller to assert a `STOP` interrupt (the `STOP` bit = 1 in the `I2CISTAT` Register). Because the slave received data from the master, the software takes no action in response to the `STOP` interrupt other than reading the `I2CISTAT` Register to clear the `STOP` bit in the `I2CISTAT` Register.

### Slave Receive Transaction with 10-Bit Address

The data transfer format for writing data from a master to a slave with 10-bit addressing is illustrated in [Figure 48](#). The procedure that follows describes the I<sup>2</sup>C Master/Slave Controller operating as a slave in 10-bit addressing mode and receiving data from the bus master.

S	Slave Address 1st Byte	W=0	A	Slave Address 2nd Byte	A	Data	A	Data	A/ $\bar{A}$	P/S
---	---------------------------	-----	---	---------------------------	---	------	---	------	--------------	-----

**Figure 48. Data Transfer Format—Slave Receive Transaction with 10-Bit Address**

1. The software configures the controller for operation as a slave in 10-bit addressing mode, as follows:
  - (a) Initialize the `MODE` field in the `I2CMODE` Register for either `SLAVE ONLY` mode or `MASTER/SLAVE` mode with 10-bit addressing.
  - (b) Optionally set the `GCE` bit.
  - (c) Initialize the `SLA[7:0]` bits in the `I2CSLVAD` Register and the `SLA[9:8]` bits in the `I2CMODE` Register.
  - (d) Set `IEN` = 1 in the `I2CCTL` Register. Set `NAK` = 0 in the I<sup>2</sup>C Control Register.

2. The Master initiates a transfer, sending the first address byte. The I<sup>2</sup>C controller recognizes the start of a 10-bit address with a match to `SLA[9:8]` and detects `R/W` bit = 0 (a Write from the master to the slave). The I<sup>2</sup>C controller acknowledges, indicating it is available to accept the transaction.
3. The Master sends the second address byte. The SLAVE mode I<sup>2</sup>C controller detects an address match between the second address byte and `SLA[7:0]`. The `SAM` bit in the `I2CISTAT` Register is set to 1, thereby causing an interrupt. The `RD` bit is cleared to 0, indicating a Write to the slave. The I<sup>2</sup>C controller acknowledges, indicating it is available to accept the data.
4. The software responds to the interrupt by reading the `I2CISTAT` Register, which clears the `SAM` bit. Because `RD` = 0, no immediate action is taken by the software until the first byte of data is received. If the software is only able to accept a single byte, it sets the `NAK` bit in the `I2CCTL` Register.
5. The Master detects the Acknowledge and sends the first byte of data.
6. The I<sup>2</sup>C controller receives the first byte and responds with Acknowledge or Not Acknowledge, depending on the state of the `NAK` bit in the `I2CCTL` Register. The I<sup>2</sup>C controller generates the receive data interrupt by setting the `RDRF` bit in the `I2CISTAT` Register.
7. The software responds by reading the `I2CISTAT` Register, finding the `RDRF` bit = 1, and then reading the `I2CDATA` Register, which clears the `RDRF` bit. If the software can accept only one more data byte, it sets the `NAK` bit in the `I2CCTL` Register.
8. The Master and Slave loops through [Step 5](#) to [Step 7](#) until the Master detects a Not Acknowledge instruction or runs out of data to send.
9. The Master sends the `STOP` or `RESTART` signal on the bus. Either of these signals can cause the I<sup>2</sup>C controller to assert the `STOP` interrupt (the `STOP` bit = 1 in the `I2CISTAT` Register). Because the slave received data from the master, the software takes no action in response to the `STOP` interrupt other than reading the `I2CISTAT` Register to clear the `STOP` bit.

### Slave Transmit Transaction With 7-bit Address

The data transfer format for a master reading data from a slave in 7-bit address mode is illustrated in [Figure 49](#). The procedure that follows describes the I<sup>2</sup>C Master/Slave Controller operating as a slave in 7-bit addressing mode and transmitting data to the bus master.

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

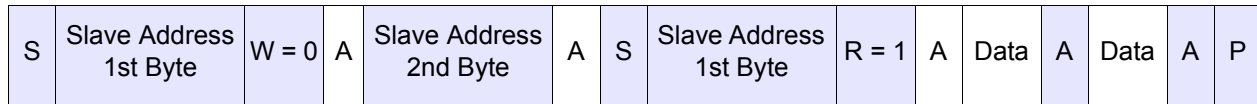
**Figure 49. Data Transfer Format—Slave Transmit Transaction with 7-bit Address**

1. The software configures the controller for operation as a slave in 7-bit addressing mode, as follows:
  - (a) Initialize the `MODE` field in the I<sup>2</sup>C Mode Register for either SLAVE ONLY mode or MASTER/SLAVE mode with 7-bit addressing.
  - (b) Optionally set the `GCE` bit.
  - (c) Initialize the `SLA[6:0]` bits in the I<sup>2</sup>C Slave Address Register.
  - (d) Set `IEN` = 1 in the I<sup>2</sup>C Control Register. Set `NAK` = 0 in the I<sup>2</sup>C Control Register.
2. The Master initiates a transfer by sending the address byte. The SLAVE mode I<sup>2</sup>C controller finds an address match and detects that the `R/W` bit = 1 (read by the master from the slave). The I<sup>2</sup>C controller acknowledges, indicating that it is ready to accept the transaction. The `SAM` bit in the I2CISTAT Register is set to 1, causing an interrupt. The `RD` bit is set to 1, indicating a Read from the slave.
3. The software responds to the interrupt by reading the I2CISTAT Register, thereby clearing the `SAM` bit. Because `RD` = 1, the software responds by loading the first data byte into the I2CDATA Register. The software sets the `TXI` bit in the I2CCTL Register to enable transmit interrupts. When the master initiates the data transfer, the I<sup>2</sup>C controller holds SCL Low until the software has written the first data byte to the I2CDATA Register.
4. SCL is released and the first data byte is shifted out.
5. After the first bit of the first data byte has been transferred, the I<sup>2</sup>C controller sets the `TDRE` bit, which asserts the transmit data interrupt.
6. The software responds to the transmit data interrupt (`TDRE` = 1) by loading the next data byte into the I2CDATA Register, which clears `TDRE`.
7. After the data byte has been received by the master, the master transmits an Acknowledge instruction (or Not Acknowledge instruction if this byte is the final data byte).
8. The bus cycles through [Step 5](#) to [Step 7](#) until the final byte has been transferred. If the software has not yet loaded the next data byte when the master brings SCL Low to transfer the most significant data bit, the slave I<sup>2</sup>C controller holds SCL Low until the data register has been written. When a Not Acknowledge instruction is received by the slave, the I<sup>2</sup>C controller sets the `NCKI` bit in the I2CISTAT Register causing the Not Acknowledge interrupt to be generated.
9. The software responds to the Not Acknowledge interrupt by clearing the `TXI` bit in the I2CCTL Register and by asserting the `FLUSH` bit of the I2CCTL Register to *empty* the data register.
10. When the Master has completed the final acknowledge cycle, it asserts a `STOP` or `RESTART` condition on the bus.
11. The Slave I<sup>2</sup>C controller asserts the `STOP/RESTART` interrupt (set `SPRS` bit in I2CISTAT Register).

12. The software responds to the STOP/RESTART interrupt by reading the I2CISTAT Register, which clears the SPRS bit.

### Slave Transmit Transaction With 10-Bit Address

The data transfer format for a master reading data from a slave with 10-bit addressing is illustrated in Figure 50. The following procedure describes the I<sup>2</sup>C Master/Slave Controller operating as a slave in 10-bit addressing mode, transmitting data to the bus master.



**Figure 50. Data Transfer Format—Slave Transmit Transaction with 10-Bit Address**

1. The software configures the controller for operation as a slave in 10-bit addressing mode.
  - (a) Initialize the MODE field in the I<sup>2</sup>C Mode Register for either SLAVE ONLY mode or MASTER/SLAVE mode with 10-bit addressing.
  - (b) Optionally set the GCE bit.
  - (c) Initialize the SLA[7:0] bits in the I2CSLVAD Register and SLA[9:8] in the I2C MODE Register.
  - (d) Set IEN = 1, and NAK = 0 in the I<sup>2</sup>C Control Register.
2. The Master initiates a transfer by sending the first address byte. The SLAVE mode I<sup>2</sup>C controller recognizes the start of a 10-bit address with a match to SLA[9:8] and detects R/W bit = 0 (a Write from the master to the slave). The I<sup>2</sup>C controller acknowledges indicating it is available to accept the transaction.
3. The Master sends the second address byte. The SLAVE mode I<sup>2</sup>C controller compares the second address byte with the value in SLA[7:0]. If there is a match, the SAM bit in the I2CISTAT Register is set = 1, causing a slave address match interrupt. The RD bit is set = 0, indicating a write to the slave. If a match occurs, the I<sup>2</sup>C controller acknowledges on the I<sup>2</sup>C bus, indicating it is available to accept the data.
4. The software responds to the slave address match interrupt by reading the I2CISTAT Register, which clears the SAM bit. Because the RD bit = 0, no further action is required.
5. The Master sees the Acknowledge and sends a RESTART instruction, followed by the first address byte with R/W set to 1. The SLAVE mode I<sup>2</sup>C controller recognizes the RESTART instruction followed by the first address byte with a match to SLA[9:8], and detects R/W = 1 (the master reads from the slave). The slave I<sup>2</sup>C controller sets the SAM bit in the I2CISTAT Register which causes the slave address match interrupt. The RD bit is set = 1. The SLAVE mode I<sup>2</sup>C controller acknowledges on the bus.

6. The software responds to the interrupt by reading the I2CISTAT Register clearing the SAM bit. The software loads the initial data byte into the I2CDATA Register and sets the TXI bit in the I2CCTL Register.
7. The Master starts the data transfer by asserting SCL Low. After the I<sup>2</sup>C controller has data available to transmit, the SCL is released and the master proceeds to shift the first data byte.
8. After the first bit of the first data byte has been transferred, the I<sup>2</sup>C controller sets the TDRE bit which asserts the transmit data interrupt.
9. The software responds to the transmit data interrupt by loading the next data byte into the I2CDATA Register.
10. The I<sup>2</sup>C Master shifts in the remainder of the data byte. The Master transmits the Acknowledge (or Not Acknowledge, if this byte is the final data byte).
11. The bus cycles through [Step 7](#) to [Step 10](#) until the final byte is transferred. If the software has not yet loaded the next data byte when the master brings SCL Low to transfer the most significant data bit, the slave I<sup>2</sup>C controller holds SCL Low until the data register is written.  
  
When a Not Acknowledge is received by the slave, the I<sup>2</sup>C controller sets the NCKI bit in the I2CISTAT Register, causing the NAK interrupt to be generated.
12. The software responds to the NAK interrupt by clearing the TXI bit in the I2CCTL Register and by asserting the FLUSH bit of the I2CCTL Register.
13. When the Master has completed the Acknowledge cycle of the last transfer, it asserts a STOP or RESTART condition on the bus.
14. The Slave I<sup>2</sup>C controller asserts the STOP/RESTART interrupt (sets the SPRS bit in the I2CISTAT Register).
15. The software responds to the STOP interrupt by reading the I2CISTAT Register and clearing the SPRS bit.

## I<sup>2</sup>C Control Register Definitions

### I<sup>2</sup>C Data Register

The I<sup>2</sup>C Data Register listed in [Table 120](#) contains the data that is to be loaded into the Shift Register to transmit onto the I<sup>2</sup>C bus. This register also contains data that is loaded from the Shift Register after it is received from the I<sup>2</sup>C bus. The I<sup>2</sup>C Shift Register is not accessible in the Register File address space, but is used only to buffer incoming and outgoing data.

Writes by the software to the I2CDATA Register are blocked if a slave Write transaction is underway (the I<sup>2</sup>C controller is in SLAVE mode, and data is being received).

Table 120. I<sup>2</sup>C Data Register (I2CDATA = F50H)

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

Bit Position	Value	Description
[7:0] DATA	I <sup>2</sup> C Data Byte	

## I<sup>2</sup>C Interrupt Status Register

The Read-only I<sup>2</sup>C Interrupt Status Register (see Table 121) indicates the cause of any current I<sup>2</sup>C interrupt and provides status of the I<sup>2</sup>C controller. When an interrupt occurs, one or more of the TDRE, RDRF, SAM, ARBLST, SPRS or NCKI bits is set. The GCA and RD bits do not generate an interrupt but rather provide status associated with the SAM bit interrupt.

Table 121. I<sup>2</sup>C Interrupt Status Register (I2CISTAT = F51H)

BITS	7	6	5	4	3	2	1	0
FIELD	TDRE	RDRF	SAM	GCA	RD	ARBLST	SPRS	NCKI
RESET	1	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
ADDR	F51H							

### TDRE—Transmit Data Register Empty

When the I<sup>2</sup>C controller is enabled, this bit is 1 when the I<sup>2</sup>C Data Register is empty. When set, this bit causes the I<sup>2</sup>C controller to generate an interrupt, except when the I<sup>2</sup>C controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit clears by writing to the I2CDATA Register.

### RDRF—Receive Data Register Full

This bit is set = 1 when the I<sup>2</sup>C controller is enabled and the I<sup>2</sup>C controller has received a byte of data. When asserted, this bit causes the I<sup>2</sup>C controller to generate an interrupt. This bit clears by reading the I2CDATA Register.

### SAM—Slave Address Match

This bit is set = 1 if the I<sup>2</sup>C controller is enabled in SLAVE mode and an address is received that matches the unique slave address or General Call Address (if enabled by the GCE bit in the I<sup>2</sup>C Mode Register). In 10-bit addressing mode, this bit is not set until a

match is achieved on both address bytes. When this bit is set, the RD and GCA bits are also valid. This bit clears by reading the I2CISTAT Register.

#### **GCA—General Call Address**

This bit is set in SLAVE mode when the General Call Address or Start byte is recognized (in either 7 or 10 bit SLAVE mode). The GCE bit in the I<sup>2</sup>C Mode Register must be set to enable recognition of the General Call Address and Start byte. This bit clears when IEN = 0 and is updated following the first address byte of each SLAVE mode transaction. A General Call Address is distinguished from a Start byte by the value of the RD bit (RD = 0 for General Call Address, 1 for Start byte).

#### **RD—Read**

This bit indicates the direction of transfer of the data. It is set when the Master is reading data from the Slave. This bit matches the least-significant bit of the address byte after the START condition occurs (for both MASTER and SLAVE modes). This bit clears when IEN = 0 and is updated following the first address byte of each transaction.

#### **ARBLST—Arbitration Lost**

This bit is set when the I<sup>2</sup>C controller is enabled in MASTER mode and loses arbitration (outputs a 1 on SDA and receives a 0 on SDA). The ARBLST bit clears when the I2CISTAT Register is read.

#### **SPRS—STOP/RESTART condition Interrupt**

This bit is set when the I<sup>2</sup>C controller is enabled in SLAVE mode and detects a STOP or RESTART condition during a transaction directed to this slave. This bit clears when the I2CISTAT Register is read. Read the RSTR bit of the I2CSTATE Register to determine whether the interrupt was caused by a STOP or RESTART condition.

#### **NCKI—NAK Interrupt**

In MASTER mode, this bit is set when a Not Acknowledge condition is received or sent and neither the START nor the STOP bit is active. In MASTER mode, this bit can only be cleared by setting the START or STOP bits.

In SLAVE mode, this bit is set when a Not Acknowledge condition is received (Master reading data from Slave), indicating the master is finished reading. A STOP or RESTART condition follows. In SLAVE mode this bit clears when the I2CISTAT Register is read.

## **I<sup>2</sup>C Control Register**

The I<sup>2</sup>C Control Register (see [Table 122](#)) enables and configures I<sup>2</sup>C operation.

► **Note:** The R/W1 bit may be set (written to 1), when IEN = 1, but cannot be cleared (written 1 or 0) anyway.

**Table 122. I<sup>2</sup>C Control Register (I2CCTL)**

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

#### **IEN—I<sup>2</sup>C Enable**

This bit enables the I<sup>2</sup>C controller.

#### **START—Send START condition**

When set, this bit causes the I<sup>2</sup>C controller (when configured as the master) to send a START condition. After it is asserted, this bit is cleared by the I<sup>2</sup>C controller after it sends the START condition or by deasserting the IEN bit. If this bit is 1, it cannot be cleared by writing to the bit. After this bit is set, a START condition is sent if there is data in the I2CDATA or I<sup>2</sup>C Shift Register. If there is no data in one of these registers, the I<sup>2</sup>C controller waits until data is loaded. If this bit is set while the I<sup>2</sup>C controller is shifting out data, it generates a RESTART condition after the byte shifts and the Acknowledge phase completes. If the STOP bit is also set, it waits till the STOP condition is sent before the START condition. If START is set while a SLAVE mode transaction is underway to this device, the START bit will be cleared and ARBLST bit in the Interrupt Status Register will be set.

#### **STOP—Send STOP condition**

When set, this bit causes the I<sup>2</sup>C controller (when configured as the master) to send the STOP condition after the byte in the I<sup>2</sup>C Shift Register has completed transmission or after a byte is received in a receive operation. When set, this bit is reset by the I<sup>2</sup>C controller after a STOP condition has been sent or by deasserting the IEN bit. If this bit is 1, it cannot be cleared to 0 by writing to the register. If STOP is set while a SLAVE mode transaction is underway, the STOP bit is cleared by hardware.

#### **BIRQ—Baud Rate Generator Interrupt Request**

This bit is ignored when the I<sup>2</sup>C controller is enabled. If this bit is set = 1 when the I<sup>2</sup>C controller is disabled (IEN = 0), the baud rate generator is used as an additional timer causing an interrupt to occur every time the baud rate generator counts down to one. The baud rate generator runs continuously in this mode, generating periodic interrupts.

#### **TXI—Enable TDRE interrupts**

This bit enables interrupts when the I<sup>2</sup>C Data Register is empty.

#### NAK—Send NAK

Setting this bit sends a Not Acknowledge condition after the next byte of data has been received. It is automatically deasserted after the Not Acknowledge is sent or the `1EN` bit is cleared. If this bit is 1, it cannot be cleared to 0 by writing to the register.

#### FLUSH—Flush Data

Setting this bit clears the I<sup>2</sup>C Data Register and sets the `TDRE` bit to 1. This bit allows flushing of the I<sup>2</sup>C Data Register when an NAK condition is received after the next data byte is written to the I<sup>2</sup>C Data Register. Reading this bit always returns 0.

#### FILTEN—I<sup>2</sup>C Signal Filter Enable

Setting this bit enables low-pass digital filters on the SDA and SCL input signals. This function provides the spike suppression filter required in I<sup>2</sup>C Fast Mode. 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.

### I<sup>2</sup>C Baud Rate High and Low Byte Registers

The I<sup>2</sup>C Baud Rate High and Low Byte registers (Table 123 and Table 124 on page 239) combine to form a 16-bit reload value, `BRG[15:0]`, for the I<sup>2</sup>C Baud Rate Generator. The I<sup>2</sup>C baud rate is calculated using the following equation.

$$\text{I}^2\text{C Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{4 \times \text{BRG}[15:0]}$$

► **Note:** If `BRG = 0000H`, then use `10000H` in the equation.

Table 123. I<sup>2</sup>C Baud Rate High Byte Register (`I2CBRH = 53H`)

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

Bit Position	Value	Description
[7:0]	I <sup>2</sup> C Baud Rate High Byte	
BRH	The most significant byte, <code>BRG[15:8]</code> , of the I <sup>2</sup> C Baud Rate Generator's reload value.	

► **Note:** If the **DIAG** bit in the **I<sup>2</sup>C Mode Register** is set to 1, a read of the **I2CBRH** Register returns the current value of the **I<sup>2</sup>C Baud Rate Counter[15:8]**.

**Table 124. I<sup>2</sup>C Baud Rate Low Byte Register (I2CBRL = F54H)**

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

Bit Position	Value	Description
[7:0]	I <sup>2</sup> C Baud Rate Low Byte	
BRL	The least significant byte, BRG[7:0], of the I <sup>2</sup> C Baud Rate Generator's reload value.	

► **Note:** If the **DIAG** bit in the **I<sup>2</sup>C Mode Register** is set to 1, a read of the **I2CBRL** Register returns the current value of the **I<sup>2</sup>C Baud Rate Counter[7:0]**.

## I<sup>2</sup>C State Register

The Read-only I<sup>2</sup>C State Register provides information on the state of the I<sup>2</sup>C bus and the I<sup>2</sup>C bus controller. When the **DIAG** bit of the I<sup>2</sup>C Mode Register is cleared, this register provides information on the internal state of the I<sup>2</sup>C controller and I<sup>2</sup>C bus as listed in [Table 126](#) on page 241.

When the **DIAG** bit of the I<sup>2</sup>C Mode Register is set, this register returns the value of the I<sup>2</sup>C controller state machine as listed in [Table 125](#).

**Table 125. I<sup>2</sup>C State Register (I2CSTATE)—Description when DIAG = 1**

BITS	7	6	5	4	3	2	1	0
FIELD	I2CSTATE_H				I2CSTATE_L			
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
ADDR	F55H							

### I2CSTATE\_H—I<sup>2</sup>C State

This field defines the current state of the I<sup>2</sup>C controller. It is the most significant nibble of the internal state machine. [Table 127](#) on page 242 defines the states for this field.

### I2CSTATE\_L—Least significant nibble of the I<sup>2</sup>C state machine

This field defines the substates for the states defined by I2CSTATE\_H. [Table 128](#) on page 243 defines the values for this field.

Table 126. I<sup>2</sup>C State Register (I2CSTATE)—Description when DIAG = 0

BITS	7	6	5	4	3	2	1	0
FIELD	ACKV	ACK	AS	DS	10B	RSTR	SCLOUT	BUSY
RESET	0	0	0	0	0	0	1	0
R/W	R	R	R	R	R	R	R	R
ADDR	F55H							

**ACKV—ACK Valid**

This bit is set, if sending data (Master or Slave) and the ACK bit in this register is valid for the byte just transmitted. This bit can be monitored if it is appropriate for software to verify the ACK value before writing the next byte to be sent. To operate in this mode, the data register must not be written when TDRE asserts; instead, the software waits for ACKV to assert. This bit clears when transmission of the next byte begins or the transaction is ended by a STOP or RESTART condition.

**ACK—Acknowledge**

This bit indicates the status of the Acknowledge for the last byte transmitted or received. This bit is set for an Acknowledge and cleared for a Not Acknowledge condition.

**AS—Address State**

This bit is active High while the address is being transferred on the I<sup>2</sup>C bus.

**DS—Data State**

This bit is active high while the data is being transferred on the I<sup>2</sup>C bus.

**10B**—This bit indicates whether a 7-bit or 10-bit address is being transmitted when operating as a Master. 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 address has been sent.

**RSTR—RESTART**

This bit is updated each time a STOP or RESTART interrupt occurs (SPRS bit set in I2CISTAT Register).

0 = STOP condition.

1 = RESTART condition.

**SCLOUT—Serial Clock Output**

Current value of Serial Clock being output onto the bus. The actual values of the SCL and SDA signals on the I<sup>2</sup>C bus can be observed via the GPIO Input Register.

**BUSY—I<sup>2</sup>C Bus Busy**

0 = No activity on the I<sup>2</sup>C Bus.

1 = A transaction is underway on the I<sup>2</sup>C bus.

Table 127. I2CSTATE\_H

State Encoding	State Name	State Description
0000	Idle	I <sup>2</sup> C bus is idle or I <sup>2</sup> C controller is disabled.
0001	Slave Start	I <sup>2</sup> C controller has received a START condition.
0010	Slave Bystander	Address did not match; ignore remainder of transaction.
0011	Slave Wait	Waiting for STOP or RESTART condition after sending a Not Acknowledge instruction.
0100	Master Stop2	Master completing STOP condition (SCL = 1, SDA = 1).
0101	Master Start/Restart	MASTER mode sending START condition (SCL = 1, SDA = 0).
0110	Master Stop1	Master initiating STOP condition (SCL = 1, SDA = 0).
0111	Master Wait	Master received a Not Acknowledge instruction, waiting for software to assert STOP or START control bits.
1000	Slave Transmit Data	Nine substates, one for each data bit and one for the Acknowledge.
1001	Slave Receive Data	Nine substates, one for each data bit and one for the Acknowledge.
1010	Slave Receive Addr1	Slave receiving first address byte (7- and 10-bit addressing) Nine substates, one for each address bit and one for the Acknowledge.
1011	Slave Receive Addr2	Slave Receiving second address byte (10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.
1100	Master Transmit Data	Nine substates, one for each data bit and one for the Acknowledge.
1101	Master Receive Data	Nine substates, one for each data bit and one for the Acknowledge.
1110	Master Transmit Addr1	Master sending first address byte (7- and 10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.
1111	Master Transmit Addr2	Master sending second address byte (10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.

Table 128. I2CSTATE\_L

State I2CSTATE_H	Substate I2CSTATE_L	Substate Name	State Description
0000–0100	0000	—	There are no substates for these I2CSTATE_H values.
0110–0111	0000	—	There are no substates for these I2CSTATE_H values.
0101	0000	Master Start	Initiating a new transaction
	0001	Master Restart	Master is ending one transaction and starting a new one without letting the bus go idle.
1000–1111	0111	Send/Receive bit 7	Sending/Receiving most significant bit
	0110	Send/Receive bit 6	
	0101	Send/Receive bit 5	
	0100	Send/Receive bit 4	
	0011	Send/Receive bit 3	
	0010	Send/Receive bit 2	
	0001	Send/Receive bit 1	
	0000	Send/Receive bit 0	Sending/Receiving least significant bit
	1000	Send/Receive Acknowledge	Sending/Receiving Acknowledge

## I<sup>2</sup>C Mode Register

The I<sup>2</sup>C Mode Register (see [Table 129](#)) provides control over master versus slave operating mode, slave address and diagnostic modes.

Table 129. I<sup>2</sup>C Mode Register (I2C Mode = F56H)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	MODE[1:0]		IRM	GCE	SLA[9:8]		DIAG
RESET	0	0		0	0	0		0
R/W	R	R/W		R/W	R/W	R/W		R/W
ADDR	F56H							

**MODE**—Selects the I<sup>2</sup>C controller operational mode

00 = MASTER/SLAVE capable (supports multi-master arbitration) with 7-bit

Slave address.

01 = MASTER/SLAVE capable (supports multi-master arbitration)  
with 10-bit slave address.

10 = SLAVE ONLY capable with 7-bit address.

11 = SLAVE ONLY capable with 10-bit address.

#### **IRM—Interactive Receive Mode**

Valid in SLAVE mode when software needs to interpret each received byte before acknowledging. This bit is useful for processing the data bytes following a General Call Address or if software wants to disable hardware address recognition.

0 = Acknowledge occurs automatically and is determined by the value of the NAK bit of the I2CCTL Register.

1 = A receive interrupt is generated for each byte received (address or data). The SCL is held Low during the Acknowledge cycle until software writes to the I2CCTL Register. The value written to the NAK bit of the I2CCTL Register is output on SDA. This value allows software to Acknowledge or Not Acknowledge after interpreting the associated address/data byte.

#### **GCE—General Call Address Enable**

Enables reception of messages beginning with the General Call Address or START byte.

0 = Do not accept a message with the General Call Address or START byte.

1 = Do accept a message with the General Call Address or START byte. When an address match occurs, the GCA and RD bits in the I<sup>2</sup>C Status Register indicates whether the address matched the General Call Address/START byte or not. Following the General Call Address byte, the software may set the IRM bit that allows software to examine the following data byte(s) before acknowledging.

#### **SLA[9:8]—Slave Address Bits 9 and 8.**

Initialize with the appropriate slave address value when using 10-bit slave addressing.

These bits are ignored when using 7-bit slave addressing.

#### **DIAG—Diagnostic Mode**

Selects read back value of the Baud Rate Reload and State registers.

0 = Reading the Baud Rate registers returns the Baud Rate register values. Reading the State register returns I<sup>2</sup>C controller state information.

1 = Reading the Baud Rate registers returns the current value of the baud rate counter. Reading the State register returns additional state information.

## **I<sup>2</sup>C Slave Address Register**

The I<sup>2</sup>C Slave Address Register (see [Table 130](#)) provides control over the lower order address bits used in 7 and 10 bit slave address recognition.

Table 130. I<sup>2</sup>C Slave Address Register (I2CSLVAD = 57H)

BITS	7	6	5	4	3	2	1	0
FIELD	SLA[7:0]							
RESET	00H							
R/W	R/W							
ADDR	F57H							

Bit Position	Value	Description
[7:0] SLA[7:0]	00H	<b>Slave Address Bits 7:0</b> Initialize with the appropriate Slave address value. When using 7-bit Slave addressing, SLA[9:7] are ignored.



# Comparator

## Overview

The Z8 Encore! XP F1680 Series devices feature two same general purpose comparators that compares two analog input signals. For each comparator, a GPIO (C0INP/C1INP) pin provides the positive comparator input, the negative input (C0INN/C1INN) can be taken from either an external GPIO pin or an internal reference. The output of each comparator is available as an interrupt source or can be routed to an external pin using the GPIO multiplex. Features for each comparator include:

- Two inputs which are connected using the GPIO multiplex (MUX).
- One input can be connected to a programmable internal reference.
- One input can be connected to the on-chip temperature sensor.
- Output can enable/disable Timer operation.
- Output can be either an interrupt source or an output to an external pin.
- Operation in STOP Mode.

## Operation

One of the comparator inputs may be connected to an internal reference which is a user selectable reference that is user programmable with 200 mV resolution.

The comparator may be powered down to save supply current or to continue to operate in STOP Mode. For details, see [Power Control Register 0](#) on page 47. In STOP Mode, the comparator interrupt (if enabled) automatically initiates a Stop Mode Recovery and generates an interrupt request. In the [Reset Status Register](#) on page 42, the Stop bit is set to 1. Also, the Comparator request bit in [Interrupt Request 1 Register](#) on page 76 is set. Following completion of the Stop Mode Recovery if interrupts are enabled, the CPU responds to the interrupt request by fetching the comparator interrupt vector.



**Caution:** *Because of the propagation delay of the comparator, it is not recommended to enable the comparator without first disabling interrupts, and then waiting for the comparator output to settle. Doing so can result in spurious interrupts after comparator enabling. The following example shows how to safely enable the comparator:*

```
di
ldx CMP0,r0
nop
nop      ; wait for output to settle
ldx IRQ0,#0 ; clear any spurious interrupts pending
ei
```

## Comparator Control Register Definitions

### Comparator 0 Control Register

The Comparator 0 Control Register (CMP0) as described in [Table 131](#) configures the comparator 0 inputs and sets the value of the internal voltage reference.

**Table 131. Comparator 0 Control Register (CMP0)**

BITS	7	6	5	4	3	2	1	0
FIELD	INPSEL	INNSEL	REFLVL				TIMTRG	
RESET	0	0	0	1	0	1	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F90H							

**INPSEL—Signal Select for Positive Input**

0 = GPIO pin used as positive comparator 0 input.

1 = Temperature sensor used as positive comparator 0 input.

**INNSEL—Signal Select for Negative Input**

0 = Internal reference disabled, GPIO pin used as negative comparator 0 input.

1 = Internal reference enabled as negative comparator 0 input.

**REFLVL—Comparator 0 Internal Reference Voltage Level**

(this reference is independent of the ADC voltage reference)

0000 = 0.0 V

0001 = 0.2 V

0010 = 0.4 V

0011 = 0.6 V

0100 = 0.8 V

0101 = 1.0 V (Default)

0110 = 1.2 V

0111 = 1.4 V

1000 = 1.6 V

1001 = 1.8 V

1010–1111 = Reserved

**TIMTRG—Timer Trigger to Toggle (enable/disable) timer operation**

00 = Disable Timer Trigger.

01 = Comparator 0 output works as Timer 0 Trigger.

10 = Comparator 0 output works as Timer 1 Trigger.

11 = Comparator 0 output works as Timer 2 Trigger.

## Comparator 1 Control Register

The Comparator 1 Control Register (CMP1) as described in [Table 132](#), configures the comparator 1 inputs and sets the value of the internal voltage reference.

**Table 132. Comparator 1 Control Register (CMP1)**

BITS	7	6	5	4	3	2	1	0
FIELD	INPSEL	INNSEL	REFLVL				TIMTRG	
RESET	0	0	0	1	0	1	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F90H							

### INPSEL—Signal Select for Positive Input

0 = GPIO pin used as positive comparator 1 input.

1 = Temperature sensor used as positive comparator 1 input.

### INNSEL—Signal Select for Negative Input

0 = Internal reference disabled, GPIO pin used as negative comparator 1 input.

1 = Internal reference enabled as negative comparator 1 input.

### REFLVL—Comparator 1 Internal Reference Voltage Level

(this reference is independent of the ADC voltage reference)

0000 = 0.0 V

0001 = 0.2 V

0010 = 0.4 V

0011 = 0.6 V

0100 = 0.8 V

0101 = 1.0 V (Default)

0110 = 1.2 V

0111 = 1.4 V

1000 = 1.6 V

1001 = 1.8 V

1010–1111 = Reserved

### TIMTRG—Timer Trigger to Toggle (enable/disable) timer operation

00 = Disable Timer Trigger.

01 = Comparator 1 output works as Timer 0 Trigger.

10 = Comparator 1 output works as Timer 1 Trigger.

11 = Comparator 1 output works as Timer 2 Trigger.



# Temperature Sensor

## Overview

The on-chip Temperature Sensor allows you to measure temperature on the die to an accuracy of roughly  $\pm 7^\circ\text{C}$  over a range of  $-40^\circ\text{C}$  to  $+105^\circ\text{C}$ . Over a reduced range, the accuracy is significantly better. This block is a moderately accurate temperature sensor for low-power applications where high accuracy is not required. Uncalibrated accuracy is significantly worse, therefore the temperature sensor is not recommended for untrimmed use:

- On-chip temperature sensor.
- $\pm 7^\circ\text{C}$  full-range accuracy for calibrated version.
- $\pm 1.5^\circ\text{C}$  accuracy over the range of  $20^\circ\text{C}$  to  $30^\circ\text{C}$ .
- Flash recalibration capability.

## Operation

The on-chip temperature sensor is a PTAT (proportional to absolute temperature) topology which has provision for zero-point calibration. A pair of Flash option bytes contain the calibration data. The temperature sensor can be disabled by a bit in the [Power Control Register 0](#) on page 47 to reduce power consumption.

The temperature sensor can be directly read by the ADC to determine the absolute value of its output. The temperature sensor output is also available as an input to the comparator for threshold type measurement determination. The accuracy of the sensor when used with the comparator is substantially less than when measured by the ADC. Maximum accuracy can be obtained by customer re-trimming the sensor using an external reference and a high-precision external reference in the target application.

During normal operation, the die undergoes heating that will cause a mismatch between the ambient temperature and that measured by the sensor. For best results, the XP device should be placed into STOP mode for sufficient time such that the die and ambient temperatures converge (this time will be dependent on the thermal design of the system). The temperature sensor should be measured immediately after recovery from STOP mode.

The following equation defines the relationship between the ADC reading and the die temperature:

**If bit 2 of TEMPCALH calibration option byte is zero, then**

$$T = (25/128) * (\text{ADC} + \{\text{TEMPCALH\_bit1}, \text{TEMPCALH\_bit0}, \text{TEMPCALL}\}) - 77$$

(where T is the temperature in  $^\circ\text{C}$ ; ADC is the 10-bit compensated ADC value).

**If bit 2 of TEMPCALH calibration option byte is one, then**

$$T = (25/128) * (ADC - \{TEMPCALH\_bit1, TEMPCALH\_bit0, TEMPCALL\}) - 77$$

(where T is the temperature in °C; ADC is the 10-bit compensated ADC value).

Here, TEMPCALH and TEMPCALL are a pair of Flash option bits containing the calibration data. For more details, see TEMPCALH and TEMPCALL in [Flash Option Bits](#) on page 267.

► **Note:** *This is just a temporary test result of 1680XP version A, the coefficient in the formula may change according to the test result of version B.*

## Calibration

The temperature sensor undergoes calibration during the manufacturing process and is maximally accurate only at 30 °C. Accuracy decreases as measured temperatures move further from the calibration point.

Because this sensor is an on-chip sensor, it is recommended that the user accounts for the difference between ambient and die temperatures when inferring ambient temperature conditions.

# Flash Memory

## Overview

The products in the Z8 Encore! XP F1680 Series feature either 24 KB (24576), 16 KB (16384), and 8 KB (8192) of non-volatile Flash memory with read/write/erase capability. The Flash memory can be programmed and erased in-circuit by either user code or through the On-Chip Debugger.

The Flash memory array is arranged in pages with 512 bytes per page. The 512-byte page is the minimum Flash block size that can be erased. Each page is divided into 4 rows of 128 bytes.

For program/data protection, the Flash memory is also divided into sectors. In the Z8 Encore! XP F1680 Series, the Flash memory is divided into 8 sectors which can be protected from programming and erase operation on a per sector basis.

The first 2 bytes of the Flash Program Memory are used as Flash Option bits. For more information on their operation, see [Flash Option Bits](#) on page 267.

[Table 133](#) describes the Flash memory configuration for each device in the Z8 Encore! XP F1680 Series. [Figure 51](#) on page 254 through [Figure 53](#) on page 256 illustrate the Flash memory arrangement.

**Table 133. Z8 Encore! XP F1680 Series Flash Memory Configurations**

Part Number	Flash Size in KB (Bytes)	Flash Pages	Program Memory Addresses	Flash Sector Size (bytes)	Number of Sectors	Pages per Sector
Z8F2480	24 (24576)	48	0000H–5FFFH	3072	8	6
Z8F1680	16 (16384)	32	0000H–3FFFH	2048	8	4
Z8F0880	8 (8192)	16	0000H–1FFFH	1024	8	2

## Flash Information Area

The Flash Information Area is separate from Program Memory and is mapped to the address range FE00H to FFFFH. Not all these addresses are user accessible. Factory trim values for the analog peripherals are stored here. Factory calibration data for the Temperature Sensor is also stored here.

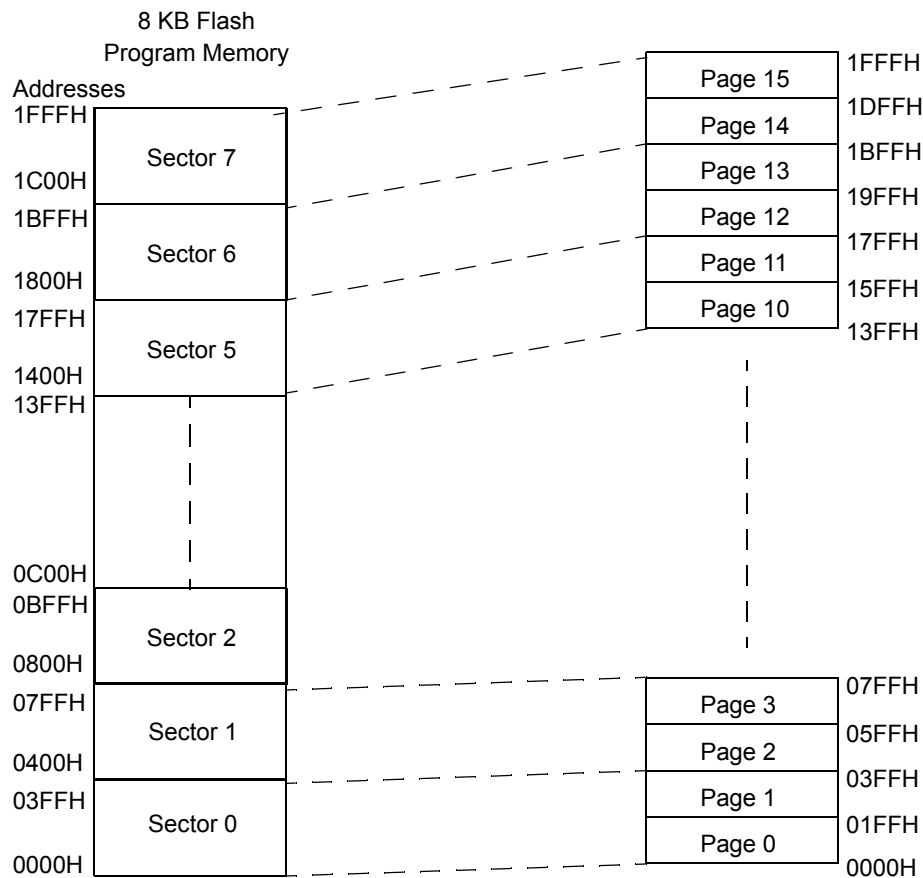


Figure 51. 8 KB Flash Memory Arrangement

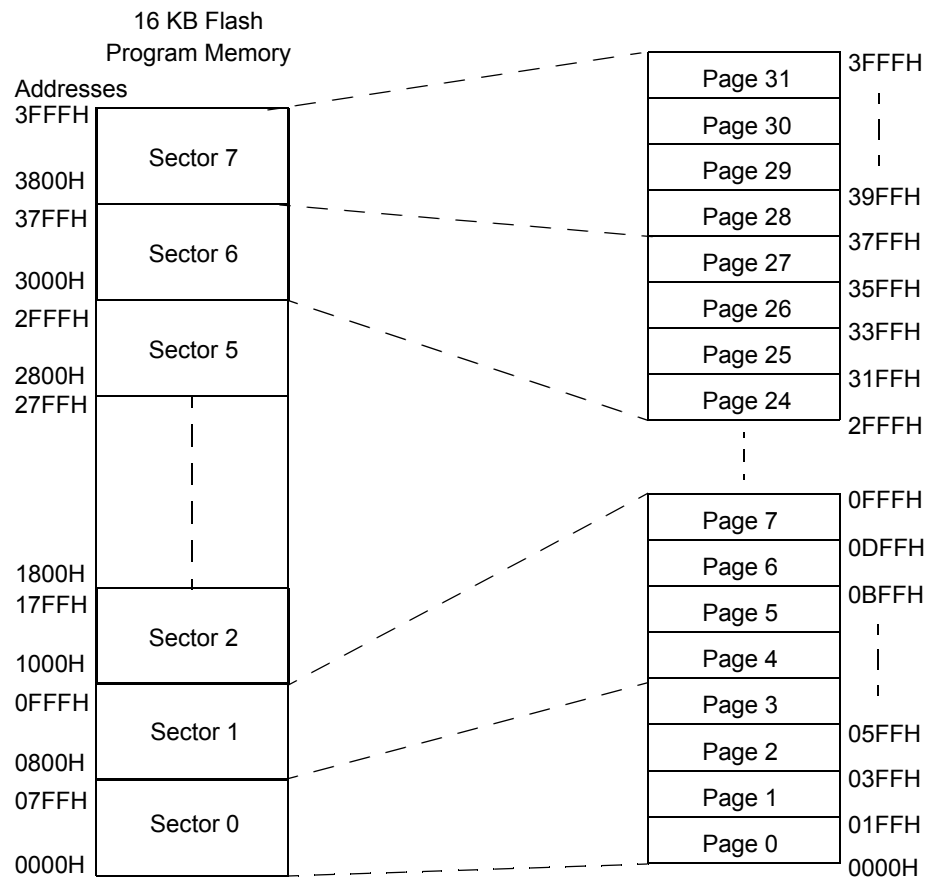


Figure 52. 16 KB Flash Memory Arrangement

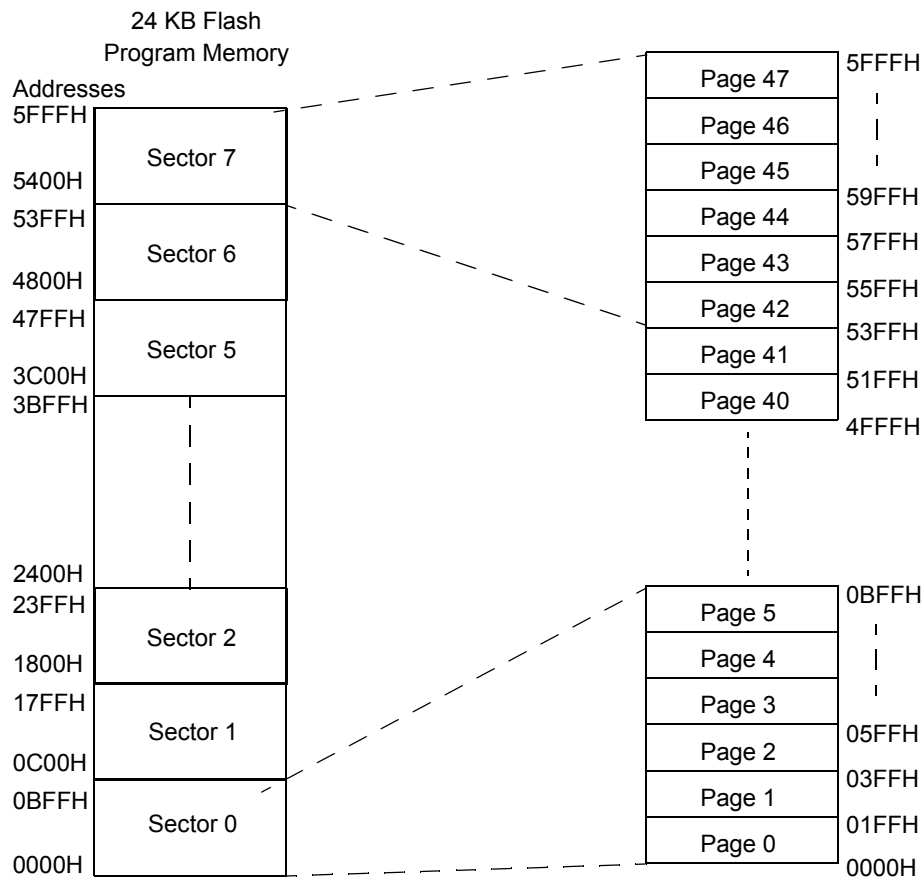


Figure 53. 24 KB Flash Memory Arrangement

## Operation

The Flash Controller programs and erases Flash memory. The Flash Controller provides the proper Flash controls and timing for byte programming, Page Erase, and Mass Erase of Flash memory.

The Flash Controller contains several protection mechanisms to prevent accidental programming or erasure. These mechanisms operate on the page, sector, and full-memory levels.

The Flow Chart in [Figure 54](#) on page 257 illustrates basic Flash Controller operation. The following sections provide details about the various operations (Lock, Unlock, Byte Programming, Page Protect, Page Unprotect, Page Select Page Erase, and Mass Erase) listed in [Figure 54](#) on page 257.

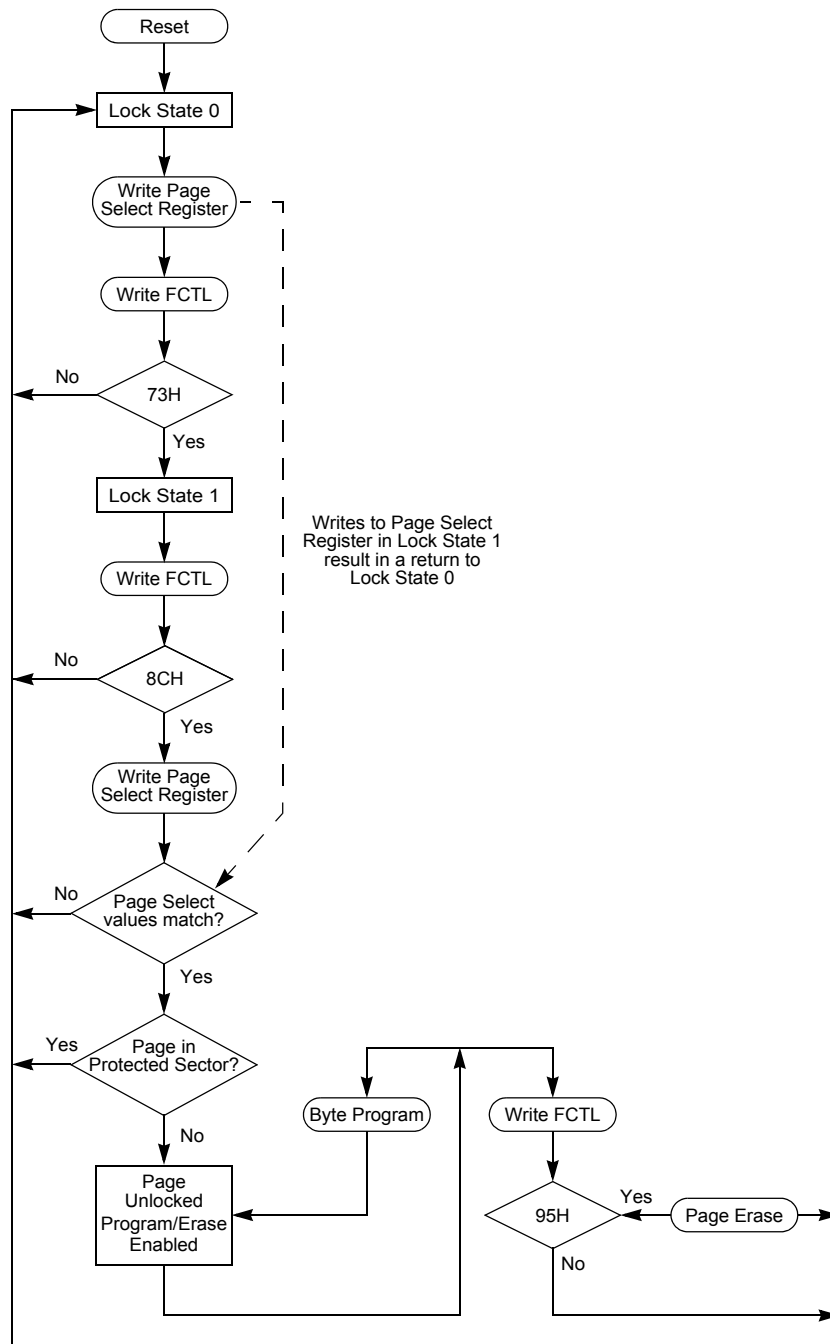


Figure 54. Flash Controller Operation Flowchart

## Flash Operation Timing Using Flash Frequency Registers

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

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

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



**Caution:** *Flash programming and erasure are not supported for system clock frequencies below 32 kHz (32768 Hz) or above 20 MHz. The Flash Frequency High and Low Byte registers must be loaded with the correct value to ensure operation of the Z8 Encore! XP F1680 Series devices.*

## Flash Code Protection Against External Access

The user code contained within the Flash memory can be protected against external access with the On-Chip Debugger. Programming the `FRP` Flash Option Bit prevents reading of the user code with the On-Chip Debugger. For more details, see [Flash Option Bits](#) on page 267 and [On-Chip Debugger](#) on page 285.

## Flash Code Protection Against Accidental Program and Erasure

The Z8 Encore! XP F1680 Series provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by a combination of the Flash Option bits, the register locking mechanism, the page select redundancy, and the sector level protection control of the Flash Controller.

### Flash Code Protection Using the Flash Option Bits

The `FWP` Flash Option Bit provides Flash Program Memory protection as listed in [Table 134](#). For more details, see [Flash Option Bits](#) on page 267.

**Table 134. Flash Code Protection Using the Flash Option Bit**

FWP	Flash Code Protection Description
0	Programming and erasing disabled for all of Flash Program Memory. In user code programming, Page Erase, and Mass Erase are all disabled. Mass Erase is available through the On-Chip Debugger.
1	Programming, Page Erase, and Mass Erase are enabled for all of Flash Program Memory.

## Flash Code Protection Using the Flash Controller

At Reset, the Flash Controller locks to prevent accidental program or erasure of the Flash memory. To program or erase the Flash memory, first write the Page Select Register with the target page. Unlock the Flash Controller by making two consecutive writes to the Flash Control register with the values 73H and 8CH, sequentially. The Page Select Register must be rewritten with the same page previously stored there. If the two Page Select writes do not match, the controller reverts to a locked state. If the two writes match, the selected page becomes active. For details, see [Figure 54](#) on page 257.

After unlocking a specific page, you can either enable Page Program or Erase. Writing the value 95H causes a Page Erase, only if the active page resides in a sector that is not protected. Any other value written to the Flash Control register locks the Flash Controller. Mass Erase is not allowed in the user code but only in through the Debug Port.

After unlocking a specific page, you can also write to any byte on that page. After a byte is written, the page remains unlocked, allowing for subsequent writes to other bytes on the same page. Further writes to the Flash Control Register cause the active page to revert to a locked state.

## Sector Based Flash Protection

The final protection mechanism is implemented on a per-sector basis. The Flash memories of Z8 Encore! devices are divided into a maximum number of 8 sectors. A sector is 1/8 of the total size of the Flash memory unless this value is smaller than the page size, in which case the sector and page sizes are equal. On the Z8 Encore! XP F1680 Series devices, the sector size is 3 KB, 2 KB or 1 KB depending on available on-chip Flash size of 24 KB, 16 KB, and 8 KB.

The Sector Protect Register controls the protection state of each Flash sector. This register is shared with the Page Select Register. This is accessed by unlocking the Flash controller and writing the command byte 5EH. The next write to the Page Select Register targets the Sector Protect Register.

The Sector Protect Register is initialized to 0 on Reset, putting each sector into an unprotected state. When a bit in the Sector Protect Register is written to 1, the corresponding sector can no longer be written or erased. After a bit of the Sector Protect Register has been set, it can not be cleared except by powering down the device.

## Byte Programming

The Flash memory is enabled for byte programming after unlocking the Flash Controller and successfully enabling either Mass Erase or Page Erase. When the Flash Controller is unlocked and Mass Erase is successfully enabled, all Program Memory locations are available for byte programming. In contrast, when the Flash Controller is unlocked and Page Erase is successfully enabled, only the locations of the selected page are available for byte programming. An erased Flash byte contains all 1's (FFH). The programming operation can only be used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from 0 to 1 requires execution of either the Page Erase or Mass Erase commands.

Byte Programming can be accomplished using the On-Chip Debugger's Write Memory command or eZ8 CPU execution of the LDC or LDCI instructions. For a description of the LDC and LDCI instructions, refer to *eZ8 CPU User Manual (UM0077)* available at [www.zilog.com](http://www.zilog.com). While the Flash Controller programs the Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. To exit programming mode and lock the Flash, write any value to the Flash Control register, except the Mass Erase or Page Erase commands.



**Caution:** *The byte at each address of the Flash memory cannot be programmed (any bits written to 0) more than twice before an erase cycle occurs.*

## Page Erase

The 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 Flash Page Select register identifies the page to be erased. Only a page residing in an unprotected sector can be erased. With the Flash Controller unlocked and the active page set, writing the value `95h` to the Flash Control register initiates the Page Erase operation. 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. If the Page Erase operation is performed using the OCD, poll the Flash Status register to determine when the Page Erase operation is complete. When the Page Erase is complete, the Flash Controller returns to its locked state.

## Mass Erase

The Flash memory can also be Mass Erased using the Flash Controller, but only by using the On-Chip Debugger. Mass Erasing the Flash memory sets all bytes to the value `FFH`. With the Flash Controller unlocked and the Mass Erase successfully enabled, writing the value `63H` to the Flash Control register initiates the Mass Erase operation. While the Flash Controller executes the Mass Erase operation, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. Using the On-Chip Debugger, poll the Flash Status register to determine when the Mass Erase operation is complete. When the Mass Erase is complete, the Flash Controller returns to its locked state.

## Flash Controller Bypass

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

Row programming is recommended for gang programming applications and large volume customers who do not require in-circuit initial programming of the Flash memory. Mass Erase and Page Erase operations are also supported when the Flash Controller is bypassed.

For more information on bypassing the Flash Controller, refer to *Third-Party Flash Programming Support for Z8 Encore!<sup>®</sup>* available at [www.zilog.com](http://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 On-Chip Debugger:

- 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 can be written when the Flash Controller is unlocked.
- The Mass Erase command is enabled through the Flash Control register.



**Caution:** *For security reasons, Flash controller allows only a single page to be opened for write/erase. When writing multiple Flash pages, the Flash controller must go through the unlock sequence again to select another page.*

## Flash Control Register Definitions

### Flash Control Register

The Flash Controller must be unlocked using the Flash Control register (Table 135) before programming or erasing the Flash memory. Writing the sequence 73H 8CH, sequentially, to the Flash Control register unlocks the Flash Controller. When the Flash Controller is unlocked, the Flash memory can be enabled for Mass Erase or Page Erase by writing the appropriate enable command to the FCTL. Switching between PRAM Modes (Normal and Remap) can also be done by writing to the FCTL, when the Flash controller is unlocked. Page Erase applies only to the active page selected in Flash Page Select register. Mass Erase is enabled only through the On-Chip Debugger. Writing an invalid value or an invalid sequence returns the Flash Controller to its locked state. The Write-only Flash Control Register shares its Register File address with the Read-only Flash Status Register.

**Table 135. Flash Control Register (FCTL)**

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

**FCMD—Flash Command**

73H = First unlock command.

8CH = Second unlock command.

95H = Page Erase command (must be third command in sequence to initiate Page Erase).

63H = Mass Erase command (must be third command in sequence to initiate Mass Erase).

5EH = Enable Flash Sector Protect Register Access

## Flash Status Register

The Flash Status register ([Table 136](#)) 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 136. Flash Status Register (FSTAT)**

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

**Program\_status—Indicate the fail or success after Flash Write/Erase.**

00, 10 = Success.

11 = Fail due to low power.

01 = Reserved.

**FSTAT—Flash Controller Status.**

000000 = Flash Controller locked.

000001 = First unlock command received (73H written).

000010 = Second unlock command received (8CH written).

000011 = Flash Controller unlocked.

000100 = Sector protect register selected.

001xxx = Program operation in progress.  
010xxx = Page erase operation in progress.  
100xxx = Mass erase operation in progress.

## Flash Page Select Register

The Flash Page Select register (see [Table 137](#) on page 263) shares address space with the Flash Sector Protect Register. Unless the Flash controller is unlocked and written with 5EH, writes to this address target the Flash Page Select Register.

The register is used to select one of the Flash memory pages to be programmed or erased. Each Flash Page contains 512 bytes of Flash memory. During a Page Erase operation, all Flash memory having addresses with the most significant 7-bits given by FPS[6:0] are chosen for program/erase operation.

**Table 137. Flash Page Select Register (FPS)**

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

### INFO\_EN—Information Area Enable

0 = Information Area is not selected

1 = Information Area is selected. The Information Area is mapped into the Program Memory address space at addresses FE00H through FFFFH.

### PAGE—Page Select

This 7-bit field identifies the Flash memory page for Page Erase and page unlocking.

Program Memory Address[15:9] = PAGE[6:0]. For the Z8F2480 devices, the upper 1 bit must always be 0. For the Z8F1680 devices, the upper 2 bits must always be 0. For the Z8F0880 devices, the upper 3 bits must always be 0.

## Flash Sector Protect Register

The Flash Sector Protect register is shared with the Flash Page Select Register. When the [Flash Control Register](#) on page 261 is written with 73H followed by 5EH, the next write to this address targets the Flash Sector Protect Register. In all other cases, it targets the Flash Page Select Register.

This register selects one of the eight available Flash memory sectors to be protected. The reset state of each Sector Protect bit is an unprotected state. After a sector is protected by

setting its corresponding register bit, it cannot be unprotected (the register bit cannot be cleared) without powering down the device.

**Table 138. Flash Sector Protect Register (FPROT)**

BITS	7	6	5	4	3	2	1	0
FIELD	SPROT7	SPROT6	SPROT5	SPROT4	SPROT3	SPROT2	SPROT1	SPROT0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF9H							

**SPROT7–SPROT0—Sector Protection**

Each bit corresponds to a 3 KB Flash sector for Z8F2480 devices. For the Z8F1680 devices each bit corresponds to a 2 KB Flash Sector. For the Z8F0880 devices each bit corresponds to a 1 KB Flash Sector.

## Flash Frequency High and Low Byte Registers

The Flash Frequency High and Low Byte registers combine to form a 16-bit value, FFREQ, to control timing for Flash program and erase operations. The 16-bit binary Flash Frequency value must contain the system clock frequency (in kHz) and 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 32 kHz or above 20 MHz. The Flash Frequency High and Low Byte registers must be loaded with the correct value to ensure proper operation of the device.*

**Table 139. Flash Frequency High Byte Register (FFREQH)**

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

**FFREQH—Flash Frequency High Byte**

High byte of the 16-bit Flash Frequency value.



Table 140. Flash Frequency Low Byte Register (FFREQ\_L)

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

**FFREQ\_L—Flash Frequency Low Byte**  
Low byte of the 16-bit Flash Frequency value.



# Flash Option Bits

## Overview

Programmable Flash Option Bits allow user configuration of certain aspects of Z8 Encore! XP F1680 Series operation. The feature configuration data is stored in the Flash Program Memory and read during Reset. The features available for control through the Flash Option Bits are:

- Watchdog Timer timeout response selection—interrupt or System Reset.
- Watchdog Timer enabled at Reset.
- The ability to prevent unwanted read access to user code in Program Memory.
- The ability to prevent accidental programming and erasure of all or a portion of the user code in Program Memory.
- VBO configuration—always enabled or disabled during STOP mode to reduce STOP mode power consumption.
- VBO voltage threshold selection.
- Oscillator mode selection for high, medium, and low-power crystal oscillators, or external RC oscillator.
- Factory trimming information for the IPO and Temperature Sensor.

## Operation

### Option Bit Configuration by Reset

Each time the Flash Option bits are programmed or erased, the device must be Reset for the change to take effect. During any Reset operation (System Reset or Stop Mode Recovery), the Flash Option bits are automatically read from the Flash Program Memory and written to Option Configuration registers. The Option Configuration registers control operation of the devices within the Z8 Encore! XP F1680 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.

### Option Bit Types

#### User Option Bits

The user option bits are contained in the first two bytes of Program Memory. User access to these bits has been provided because these locations contain application-specific device

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

### Trim Option Bits

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

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

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

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

### Calibration Option Bits

The calibration option bits are also contained in the information page. These bits are factory programmed values intended for use in software correcting the device's analog performance. To read these values, the user code must employ the LDC instruction to access the information area of the address space as defined in [Flash Information Area](#) on page 23.

The following code example shows how to read the calibration data from the Flash Information Area.

```
; get value at info address 60 (FE60h)
ldx FPS, #80 ; enable access to flash info page
ld R0, #FE
ld R1, #60
ldc R2, @RR0 ; R2 now contains the calibration value
```

## Flash Option Bit Control Register Definitions

### Trim Bit Address Register

This register contains the target address for an access to the trim option bits ([Table 141](#)). Trim Bit Address (00H–1FH) maps to the Information Area address (20H to 3FH) as listed in [Table 142](#).

**Table 141. Trim Bit Address Register (TRMADR)**

BITS	7	6	5	4	3	2	1	0
FIELD	TRMADR—Trim Bit Address (00H to 1FH)							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF6H							

**Table 142. Trim Bit Address Map**

Trim Bit Address	Information Area Address
00H	20H
01H	21H
02H	22H
03H	23H
:	:
1FH	3FH

### Trim Bit Data Register

This register contains the read or write data for access to the trim option bits ([Table 143](#)).

**Table 143. Trim Bit Data Register (TRMDR)**

BITS	7	6	5	4	3	2	1	0
FIELD	TRMDR—Trim Bit Data							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF7H							

## Flash Option Bit Address Space

The first two bytes of Flash Program Memory at addresses 0000H and 0001H are reserved for the user-programmable Flash Option bits.

### Flash Program Memory Address 0000H

Table 144. Flash Option Bits at Program Memory Address 0000H

BITS	7	6	5	4	3	2	1	0
FIELD	WDT_RES	WDT_AO	OSC_SEL[1:0]		VBO_AO	FRP	PRAM_M	FWP
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Program Memory 0000H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

#### WDT\_RES—Watchdog Timer Reset

0 = Watchdog Timer timeout generates an interrupt request. Interrupts must be globally enabled for the eZ8 CPU to acknowledge the interrupt request.

1 = Watchdog Timer timeout causes a System 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 cannot 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 or external clock input (<4 MHz).

01 = Reserved.

10 = Medium power for use with medium frequency crystals or ceramic resonators (1 MHz to 8.0 MHz).

11 = Maximum power for use with high frequency crystals (8.0 MHz to 20.0 MHz). This setting is 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. And it is controlled by VBO/LVD control bit of Power Control Register in ACTIVE and HALT mode.

1 = Voltage Brownout Protection is always enabled including during STOP mode. And it cannot be disabled by VBO/LVD control bit of Power Control Register in any mode. This setting is default for unprogrammed (erased) Flash.

#### FRP—Flash Read Protect

0 = User program code is inaccessible. Limited control features are available through the On-Chip Debugger.

1 = User program code is accessible. All On-Chip Debugger commands are enabled. This setting is default for unprogrammed (erased) Flash.

#### PRAM\_M—On-chip Program RAM Mode select

0 = Program RAM is used as on-chip Register RAM, and it begins at the first available Register File address space. See [Register Map](#) on page 25.

1 = Program RAM is used as on-chip Program RAM, and it begins at address E000H in the Program Memory address space. This setting is default for unprogrammed (erased) Flash.

#### FWP—Flash Write Protect

This Option Bit provides Flash Program Memory protection:

0 = Programming and erasure disabled for all of Flash Program Memory.

Programming, Page Erase and Mass Erase through User Code are disabled.

Mass Erase is available using the On-Chip Debugger.

1 = Programming, Page Erase and Mass Erase are enabled for all of Flash Program Memory.

## Flash Program Memory Address 0001H

Table 145. Flash Option Bits at Program Memory Address 0001H

BITS	7	6	5	4	3	2	1	0
FIELD	EXTLTMG		FLASH_WR_PRO_EN	EXTL_AO	Reserved		X2_Mode	X2TL_AO
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Program Memory 0001H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

#### EXTLTMG—External Crystal Reset Timing

00 = 500 Internal Precision Oscillator Cycles.

01 = 1000 Internal Precision Oscillator Cycles.

10 = 5000 Internal Precision Oscillator Cycles.

11 = 10,000 Internal Precision Oscillator Cycles.

**FLASH\_WR\_PRO\_EN—Flash write operation protect**

0 = Flash write protect disable

1 = Flash write protect with internal LVD

**EXTL\_AO—External Crystal Always ON**

0 = Crystal oscillator is enabled during Reset, resulting in longer reset timing.

1 = Crystal oscillator is disabled during Reset, resulting in shorter reset timing.

► **Note:** *This bit determines the state of the external crystal oscillator at Reset. Its selection as system clock must be done in the Oscillator Control Register (OSCCTL0).*

**X2\_Mode—Secondary Crystal Mode Select**

0 = External clock input

1 = External 32 kHz watch crystal

**X2TL\_AO—Secondary Crystal Always ON**

0 = Secondary Crystal Oscillator is enabled during reset.

1 = Secondary Crystal Oscillator is disabled during reset.

► **Note:** *This bit determines state of the Secondary Crystal Oscillator at Reset. Its selection as peripheral clock must be done in the Peripheral Control (for example, Timer Control2) register.*

## Trim Bit Address Space

All available Trim bit addresses and their functions are listed in [Table 146](#). See [Table 147](#) through [Table 156](#) for details.

**Table 146. Trim Bit Address Description**

Address	Function
00H	Temperature Sensor Trim0
01H	Temperature Sensor Trim1
02H	Internal Precision Oscillator
03H	VBO and LVD
04H	ADC and Comparator 0/1
05H	ADC Reference Voltage
06H	20-M Oscillator and 32-K Oscillator
07H	Reserved
08H	Reserved

## Trim Bit Address 0000H

Table 147. Trim Option Bits at Address 0000H (TTEMP0)

BITS	7	6	5	4	3	2	1	0
FIELD	TS_FINE				Reserved	TS_ULTRAFINE		
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory 0020H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

### TS\_FINE—Temperature Sensor Fine Control Trim Bits

Contains fine control offset trimming bits for Temperature Sensor.

**Reserved**—Must be 1.

### TS\_ULTRAFINE—Temperature Sensor Ultra Fine Control Trim Bits

Contains ultra fine control offset trimming bits for Temperature Sensor.

## Trim Bit Address 0001H

Table 148. Trim Option Bits at 0001H (TTEMP1)

BITS	7	6	5	4	3	2	1	0
FIELD	TS_NEG			TS_COARSE				
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory 0021H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

### TS\_NEG—Temperature Sensor Negative Control Trim Bits

Negative control offset trimming bits for Temperature Sensor.

### TS\_COARSE—Temperature Sensor Coarse Control Trim Bits

Contains coarse control offset trimming bits for Temperature Sensor.

## Trim Bit Address 0002H

Table 149. Trim Option Bits at 0002H (TIPO)

BITS	7	6	5	4	3	2	1	0
FIELD	IPO_TRIM							
RESET	U							
R/W	R/W							
ADDR	Information Page Memory 0022H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

**IPO\_TRIM—Internal Precision Oscillator Trim Byte**  
Contains trimming bits for Internal Precision Oscillator.

## Trim Bit Address 0003H

Table 150. Trim Option Bits at Address 0003H (TLVD\_VBO)

BITS	7	6	5	4	3	2	1	0
FIELD	VBO_TRIM			LVD_TRIM				
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory 0023H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

### LVD\_TRIM—Low Voltage Detect Trim

This trimming affects the low-voltage detection threshold. Each LSB represents a 50 mV change in the threshold level. Alternatively, the low voltage threshold may be computed from the options bit value by the following equation:

$$\text{LVD\_LVL} = 3.2 \text{ V} - \text{LVD\_TRIM} * 0.05 \text{ V}$$

LVD_TRIM	LVD Threshold (V)			Description
	Minimum	Typical	Maximum	
00000		3.20		Maximum LVD threshold
00001		3.15		
00010		3.10		
00011		3.05		

LVD_TRIM	LVD Threshold (V)			Description
	Minimum	Typical	Maximum	
00100		3.00		
00101		2.95		
00110		2.90		
00111		2.85		
01000		2.80		
01001		2.75		
01010		2.70		
01011		2.65		
01100		2.60		
01101		2.55		
01110		2.50		
01111		2.45		
10000		2.40		
10001		2.35		
10010		2.30		
10011		2.25		
10100		2.20		
10101		2.15		
10110		2.10		
10111		2.05		
11000		2.00		
11001		1.95		
11010		1.90		
11011		1.85		
11100		1.80		
11101		1.75		
11110		1.70		
11111		1.65		Minimum LVD threshold, Default on Reset

## Trim Bit Address 0004H

Table 151. Trim Option Bits at 0004H (TCOMP\_ADC)

BITS	7	6	5	4	3	2	1	0
FIELD	TEMPST	COMP1_OPT	Reserved	COMP0_OPT	ADC_OPT			
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory 0024H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

**TEMPST**—Temperature Sensor test control bit. The default is 1.

**COMP1\_OPT**—Comparator 1 trim bit. COMP1\_OPT(bit 6) is for comparator's "hys" input. See Table 152 for the truth table of this pin.

**COMP0\_OPT**—Comparator 0 trim bit. COMP0\_OPT(bit 4) is for comparator's "hys" input. See Table 152 for the truth table of this pin.

Table 152. Truth Table of HYS

HYS	Hysteresis input
1	Enable
0	Disable

**ADC\_OPT**—ADC Trim Values

Contains factory trimmed values for the ADC block.

## Trim Bit Address 0005H

Table 153. Trim Option Bits at 0005H (TVREF)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved			VREF_TRIM				
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory 0025H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

Reserved—Must be 1.

**VREF\_TRIM**—5 bits trimming signal for internal ADC reference voltage generator. Used to change ADC reference voltage to 2.0V.

## Trim Bit Address 0006H

**Table 154. Trim Option Bits at 0006H (TBG)**

BITS	7	6	5	4	3	2	1	0
FIELD	X1_TRIM				X0_TRIM			
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory 0026H							

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

X1\_TRIM—4 bits trimming signal for 20 M crystal oscillator.

X0\_TRIM—4 bits trimming signal for 32 K second crystal oscillator.

## Trim Bit Address 0007H

**Table 155. Trim Option Bits at 0007H (TFilter0)**

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory 0027H							

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

## Trim Bit Address 0008H

Table 156. Trim Option Bits at 0008H (TFilter1)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory 0028H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

## Zilog Calibration Bits

### Temperature Sensor Calibration Bits

#### TEMPCALH—Temperature Sensor Calibration High Byte

Table 157. Temperature Sensor Calibration High Byte at FE60H (TEMPCALH)

BITS	7	6	5	4	3	2	1	0
FIELD	TEMPCALH							
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory FE60H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

#### TEMPCALH—Temperature Sensor Calibration High Byte

The bits[7:3] of TEMPCALH are not useful. The bit 2 of TEMPCALH indicates whether the calibration data is added to or subtracted from the measured ADC data. If bit 2 is zero, the calibration data is added; if bit 2 is one, the calibration data is subtracted. The bit 1 and bit 0 of TEMPCALH are the high two bits of 10-bit calibration data.

**TEMPCALL—Temperature Sensor Calibration Low Byte**

**Table 158. Temperature Sensor Calibration Low Byte at FE61H (TEMPCALL)**

BITS	7	6	5	4	3	2	1	0
FIELD	TEMPCALL							
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Information Page Memory FE61H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

**TEMPCALL—Temperature Sensor Calibration Low Byte**

TEMPCALL is the low eight bits of 10-bit calibration data. The whole 10-bit calibration data is {TEMPCALH[1:0], TEMPCALL}.



# Non-Volatile Data Storage

## Overview

The Z8 Encore! XP F1680 Series devices contain a Non-Volatile Data Storage (NVDS) element of up to 256 bytes. This memory can perform over 100,000 write cycles.

## Operation

The NVDS is implemented by special purpose Zilog<sup>®</sup> software stored in areas of Program memory not accessible to you. These special-purpose routines use the Flash memory to store the data. The routines incorporate a dynamic addressing scheme to maximize the Write/Erase endurance of the Flash.

► **Note:** *Different members of the Z8 Encore! XP F1680 Series feature multiple NVDS array sizes. For more details, see [Table 1](#) on page 2.*

## NVDS Code Interface

Two routines are required to access the NVDS: a write routine and a read routine. Both of these routines are accessed with a CALL instruction to a pre-defined address outside the program memory accessible to you. Both the NVDS address and data are single-byte values. In order NOT to disturb the user code, these routines save the working register set before using it, so 16 bytes of stack space is needed to preserve the site. After finishing the call to these routines, the working register set of the user code is recovered.

During both read and write accesses to the NVDS, interrupt service is NOT disabled. Any interrupts that occur during the NVDS execution must not disturb the working register and existing stack contents; or else the array becomes corrupted. Disabling interrupts before executing NVDS operations is recommended.

Use of the NVDS requires 16 bytes of available stack space. The contents of the working register set are saved before calling NVDS read or write routines.

For correct NVDS operation, the Flash Frequency Registers must be programmed based on the system clock frequency. For more details, see [Flash Operation Timing Using Flash Frequency Registers](#) on page 258.

## Byte Write

To write a byte to the NVDS array, the user code must first push the address, then the data byte onto the stack. The user code issues a `CALL` instruction to the address of the Byte Write routine (`0x43FD`). At the return from the sub-routine, the write status byte resides in working register R0. The bit fields of this status byte are defined in [Table 161](#). Also, the user code should pop the address and data bytes off the stack.

The write routine uses 16 bytes of stack space in addition to the two bytes of address and data pushed by the user code. Sufficient memory must be available for this stack usage.

Because of the flash memory architecture, NVDS writes exhibit a non-uniform execution time. In general, a write takes 136  $\mu$ s (assuming a 20 MHz system clock). For every 200 writes, however, a maintenance operation is necessary. In this rare occurrence, the write takes up to 58 ms to complete. Slower system clock speeds result in proportionally higher execution times.

NVDS byte writes to invalid addresses (those exceeding the NVDS array size) have no effect. Illegal write operations have a 7  $\mu$ s execution time.

**Table 161. Write Status Byte**

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved					FE	IGADDR	WE
DEFAULT VALUE	0	0	0	0	0	0	0	0

**Reserved—Must be 0.**

**FE—Flash Error**

If Flash error is detected, this bit is set to 1.

**IGADDR—Illegal address**

When NVDS byte writes to invalid addresses (those exceeding the NVDS array size) occur, this bit is set to 1.

► **Note:** *When the NVDS array size is 256 bytes, there is no address exceeding the size, the IGADDR bit is of no use.*

**WE—Write Error**

A failure occurs during writing data into Flash. When writing data into a certain address, a read back operation is performed. If the read back value is not the same as the value written, this bit is set to 1.

## Byte Read

To read a byte from the NVDS array, user code must first push the address onto the stack. User code issues a `CALL` instruction to the address of the byte-read routine (`0x4000`). At the return from the sub-routine, the read byte resides in working register R0, and the read status byte resides in working register R1. The bit fields of this status byte are defined in [Table 162](#). Also, the user code should pop the address byte off the stack.

The read routine uses 16 bytes of stack space in addition to the 1 byte of address pushed by you. Sufficient memory must be available for this stack usage. Because of the Flash memory architecture, NVDS reads exhibit a non-uniform execution time. A read operation takes between 71  $\mu$ s and 258  $\mu$ s (assuming a 20 MHz system clock). Slower system clock speeds result in proportionally higher execution times.

NVDS byte reads from invalid addresses (those exceeding the NVDS array size) return 0xff. Illegal read operations have a 6  $\mu$ s execution time. The status byte returned by the NVDS read routine is zero for successful read. If the status byte is non-zero, there is a corrupted value in the NVDS array at the location being read. In this case, the value returned in R0 is the byte most recently written to the array that does not have an error.

**Table 162. Read Status Byte**

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved			DE	Reserved	FE	IGADDR	Reserved
DEFAULT VALUE	0	0	0	0	0	0	0	0

**Reserved—Must be 0.**

### **DE—Data Error**

When reading a NVDS address, if an error is found in the latest data corresponding to this NVDS address, this bit is set to 1. NVDS source code steps forward until finding a valid data at this address.

### **FE—Flash Error**

If Flash error is detected, this bit is set to 1.

### **IGADDR—Illegal address**

When NVDS byte reads from invalid addresses (those exceeding the NVDS array size) occur, this bit is set to 1.

► **Note:** *When the NVDS array size is 256 bytes, there is no address exceeding the size, the IGADDR bit is of no use.*

## Power Failure Protection

The NVDS routines employ error checking mechanisms to ensure a power failure endangers only the most recently written byte. Bytes previously written to the array are not perturbed. For this protection to function, the VBO must be enabled (see [Low-Power Modes](#) on page 45) and configured for a threshold voltage of 2.4 V or greater (see [Trim Bit Address Space](#) on page 272).

A System Reset (such as a pin reset or watchdog timer reset) that occurs during a write operation also perturbs the byte currently being written. All other bytes in the array are unperturbed.

## Optimizing NVDS Memory Usage for Execution Speed

As listed in [Table 163](#), the NVDS read time varies drastically, this discrepancy being a trade-off for minimizing the frequency of writes that require post-write page erases. The NVDS read time of address N is a function of the number of writes to addresses other than N since the most recent write to address N, as well as the number of writes since the most recent page erase. Neglecting effects caused by page erases and results caused by the initial condition in which the NVDS is blank, a rule of thumb is that every write since the most recent page erase causes read times of unwritten addresses to increase by 0.8  $\mu$ s, up to a maximum of 258  $\mu$ s.

**Table 163.NVDS Read Time**

Operation	Minimum Latency ( $\mu$ s)	Maximum Latency ( $\mu$ s)
Read	71	258
Write	126	136
Illegal Read	6	6
Illegal Write	7	7

If NVDS read performance is critical to your software architecture, you can optimize your code for speed by using either of the methods listed below.

1. Periodically refresh all addresses that are used. This is the most useful method. The optimal use of NVDS in terms of speed is to rotate the writes evenly among all addresses planned to use, bringing all reads closer to the minimum read time. Because the minimum read time is much less than the write time, however, actual speed benefits are not always realized.
2. Use as few unique addresses as possible. This helps to optimize the impact of refreshing.

# On-Chip Debugger

## Overview

The Z8 Encore! XP F1680 Series device contains an integrated On-Chip Debugger (OCD) that provides advanced debugging features including:

- Reading and writing of the Register File
- Reading and writing of Program and Data Memory
- Setting of Breakpoints
- Executing eZ8 CPU instructions

## Architecture

The OCD consists of four primary functional blocks:

- Transmitter
- Receiver
- Auto-baud detector/generator
- Debug controller

Figure 55 illustrates the architecture of the On-Chip Debugger.

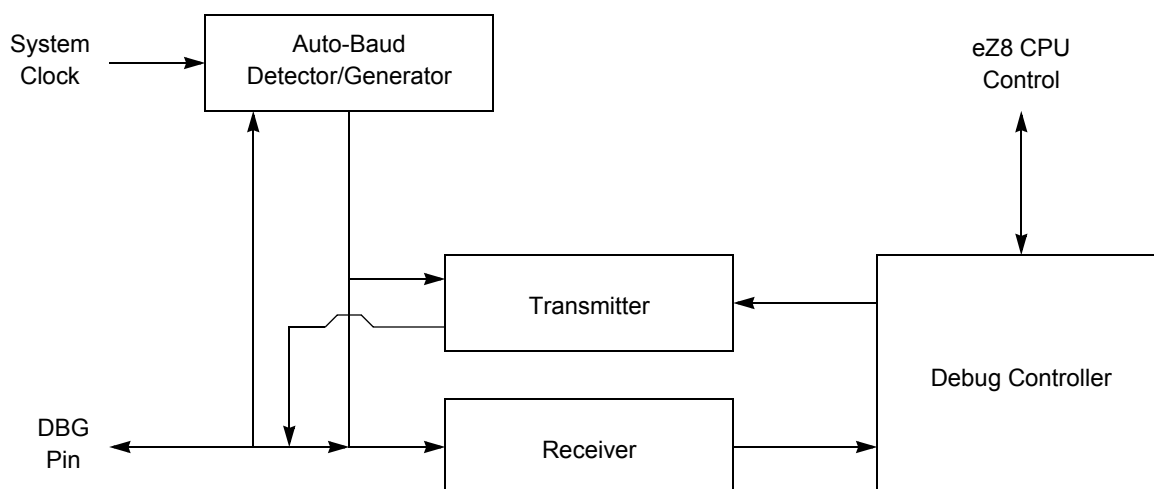


Figure 55. On-Chip Debugger Block Diagram

## Operation

### On-Chip Debugger Interface

The On-Chip Debugger (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 interfaces the Z8 Encore! XP F1680 Series device 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 depicted in [Figure 56](#) and [Figure 57](#).



**Caution:** For operation of the Z8 Encore! XP F1680 Series device, 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 should always be connected to  $V_{DD}$  through an external pull-up resistor.

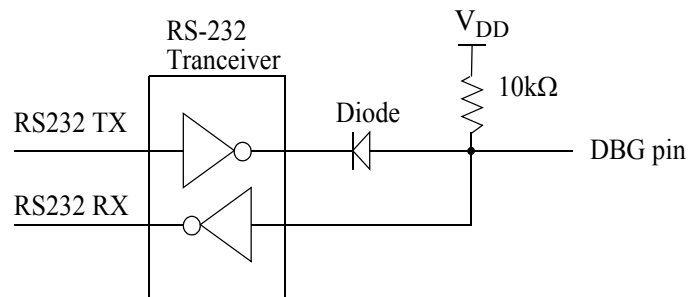


Figure 56. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (1)

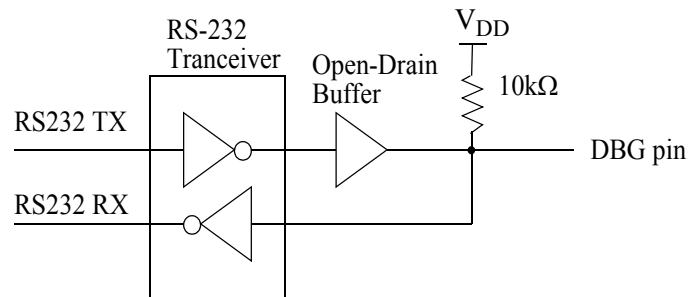


Figure 57. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (2)

## **DEBUG Mode**

The operating characteristics of the Z8 Encore! XP F1680 Series device 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 or otherwise defined by the on-chip peripheral to disable in DEBUG mode.
- Automatically exits HALT mode.
- Constantly refreshes the Watch-Dog Timer, if enabled.

### **Entering DEBUG Mode**

The device enters DEBUG mode following any of the these operations:

- Writing the DBGMODE bit in the OCD Control Register to 1 using the OCD interface.
- eZ8 CPU execution of a BRK (Breakpoint) instruction (when enabled).
- Match of PC to OCDCNTR register (when enabled).
- OCDCNTR register decrements to 0000H (when enabled).
- The DBG pin is Low when the device exits Reset.

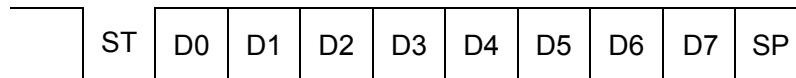
### **Exiting DEBUG Mode**

The device exits DEBUG mode following any of these operations:

- Clearing the DBGMODE bit in the OCD Control Register to 0.
- Power-on reset.
- Voltage Brownout reset.
- Asserting the  $\overline{\text{RESET}}$  pin Low to initiate a Reset.
- Driving the DBG pin Low when the device is in STOP mode initiates a System Reset.

## **OCD Data Format**

The On-Chip Debugger (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 58](#) on page 288).



ST = Start Bit  
SP = Stop Bit  
D0-D7 = Data Bits

Figure 58. 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 On-Chip Debugger has 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 contains 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. If the data can be synchronized with the system clock, the autobaud generator can run as high as the system clock frequency (1 clock / bit). The maximum recommended baud rate is the system clock frequency divided by 8. Table 164 lists minimum and recommended maximum baud rates for sample crystal frequencies.

Table 164. OCD Baud-Rate Limits

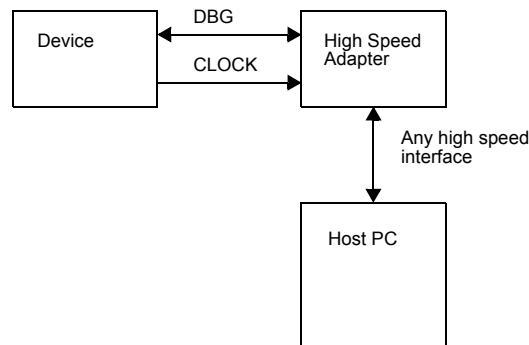
System Clock Frequency	Maximum Asynchronous Baud Rate (bits/s)	Minimum Baud Rate (bits/s)
20.0 MHz	2.5 M	39.1 k
1.0 MHz	125 k	1.96 k
32 kHz	4096	64

If the OCD receives a Serial Break (ten or more continuous bits Low) the Auto-Baud Detector/Generator resets. The Auto-Baud Detector/Generator can then be reconfigured by sending 80H. If the Auto-Baud Detector overflows while measuring the Auto-Baud character, the Auto-Baud Detector will remain reset.

## High Speed Synchronous

It is possible to operate the serial On-Chip Debugger at high speeds. To operate at high speeds, data must be synchronized with an external clock. High speed synchronous communication will only work when using an external clock source. To operate in high speed synchronous mode, simply Auto-Baud to the desired speed. The Auto-Baud generator will automatically run at the desired baud rate.

Slow bus rise times due to the pullup resistor become a limiting factor when operating at high speeds. To compensate for slow rise times, the output driver can be configured to drive the line high. If the TXD (Transmit Drive) bit is set, the line will be driven both high and low during transmission. The line starts being driven at the beginning of the start bit and stops being driven at the middle of the stop bit. If the TXDH (Transmit Drive High) bit is set, the line will be driven high until the input is high or the center of the bit occurs, whichever is first. If both TXD and TXDH are set, the pin will be driven high for one clock period at the beginning of each 0 to 1 transition. An example of a high speed synchronous interface is illustrated in [Figure 59](#).



**Figure 59. Synchronous Operation**

## OCD Serial Errors

The On-Chip Debugger can detect any of the following error conditions on the DBG pin:

- Serial Break (a minimum of ten 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 F1680 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.

Automatic Reset

The Z8 Encore! XP F1680 Series devices have the capability to switch clock sources during operation. If the Auto-Baud is set and the clock source is switched, the Auto-Baud value becomes invalid. A new Auto-Baud value must be configured with the new clock frequency.

The oscillator control logic has clock switch detection. If a clock switch is detected and the Auto-Baud is set, the device will automatically send a Serial Break for 4096 clocks. This will reset the Auto-Baud and indicate to the host that a new Auto-Baud character should be sent.

Transmit Flow Control

Transmit flow control is implemented by the use of a remote start bit. When transmit flow control is enabled, the transmitter will wait for the remote host to send the start bit. Transmit flow control is useful in applications where receive overruns can occur.

The remote host can transmit a remote start bit by sending the character FFH. The transmitter will append its data after the start bit. Due to the wire-and nature of the open drain bus, the start bit sent by the remote host and the data bits sent by the Z8 Encore! XP F1680 Series device appear as one character. This is illustrated in Figure 60.

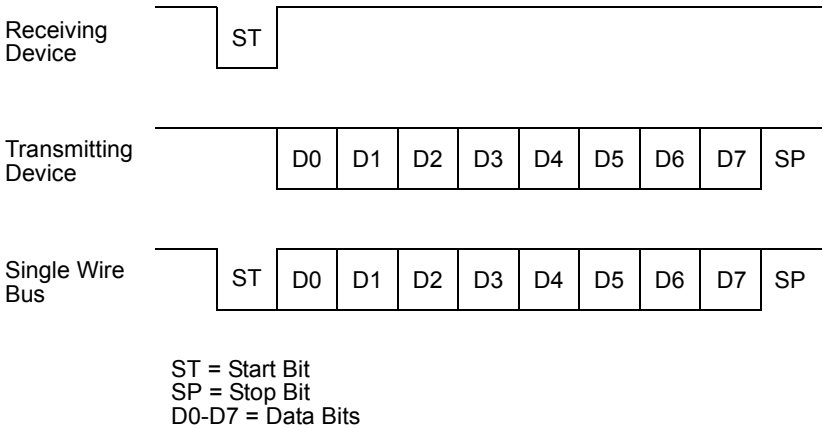


Figure 60. Start Bit Flow Control

## Breakpoints

Execution Breakpoints are generated using the BRK instruction (opcode 00H). When the eZ8 CPU decodes a BRK instruction, it signals the On-Chip Debugger. 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 remains able to service interrupt requests.

The loop on BRK instruction can service interrupts in the background. For interrupts to be serviced in the background, there cannot be any breakpoints in the interrupt service routine. 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 does not automatically enable interrupts when using this feature. Interrupts are typically disabled during critical sections of code where interrupts do not occur (such as adjusting the stack pointer or modifying shared data).

Host 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 on which it is looping, software must not set the DBGMODE bit of the OCDCTL register. The CPU may have vectored to an interrupt service routine. Instead, software clears the BRKLP bit. This allows the CPU to finish the interrupt service routine it may be in and return to the BRK instruction. When the CPU returns to the BRK instruction on which it was previously looping, it automatically sets the DBGMODE bit and enters DEBUG mode.

The majority of the OCD commands remain disabled when the eZ8 CPU is looping on a BRK instruction. The eZ8 CPU 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, erase the corresponding page of Flash memory and reprogram with the original data.

## OCD CNTR Register

The On-Chip Debugger 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 0.
- Generate a BRK when its value matches the Program Counter.

When configured as a counter, the OCDCNTR register starts counting when the On-Chip Debugger leaves DEBUG mode and stops counting when it enters DEBUG mode again or when it reaches the maximum count of `FFFFH`. The OCDCNTR register automatically resets itself to `0000H` when the OCD exits DEBUG mode if it is configured to count clock cycles between breakpoints.

If the OCDCNTR register is configured to generate a BRK when it counts down to zero, it will not be reset when the CPU starts running. The counter will start counting down toward zero once the On-Chip debugger leaves DEBUG mode. If the On-Chip Debugger enters DEBUG mode before the OCDCNTR register counts down to zero, the OCDCNTR will stop counting.

If the OCDCNTR register is configured to generate a BRK when the program counter matches the OCDCNTR register, the OCDCNTR register will not be reset when the CPU resumes executing and it will not be decremented when the CPU is running. A BRK will be generated when the program counter matches the value in the OCDCNTR register before executing the instruction at the location of the program counter.



**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 retains the residual value when generating the CRC. If the OCDCNTR is used to generate a BRK, its value must be written as a final step before leaving DEBUG mode.*

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

When the OCDCNTR register is read, it returns the inverse of the data in this register. The OCDCNTR register is only decremented when counting. The mode where it counts the number of clock cycles in between execution is achieved by counting down from its maximum count. When the OCDCNTR register is read, the counter appears to have counted up because its value is inverted. The value in this register is always inverted when it is read. If this register is used as a hardware breakpoint, the value read from this register will be the inverse of the data actually in the register.

## On-Chip Debugger Commands

The host communicates to the On-Chip Debugger by sending OCD commands using the DBG interface. During normal operation, only a subset of the OCD commands are available. In Debug mode, all OCD commands become available unless the user code is protected by programming the Flash Read Protect Option Bit (FRP). The Flash Read Protect Option Bit prevents the code in memory from being read out of the Z8 Encore! XP F1680 Series device. When this option is enabled, several of the OCD commands are disabled. When the Read Protect Option Bit is enabled and the Information

Area Write Protect bits are enabled, asserting the TESTMODE pad does NOT put the Z8 Encore! XP F1680 Series in Flash Test mode.

Table 165 contains a summary of the On-Chip Debugger commands. Each OCD command is described in further detail in the bulleted list following the table.

The table indicates those commands that operate when the device is not in DEBUG mode (normal operation) and those commands that are disabled by programming the Flash Read Protect Option Bit.

**Table 165. On-Chip Debugger Commands**

Debug Command	Command Byte	Enabled when NOT in DEBUG mode?	Disabled by Read Protect Option Bit
Read 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	–
Read OCD Control Register	05H	Yes	–
Write Program Counter	06H	–	Disabled
Read Program Counter	07H	–	Disabled
Write Register	08H	–	Writes to on-chip peripheral registers are enabled. Writes to the on-chip RAM are disabled.
Read Register	09H	–	Reads from on-chip peripheral registers are enabled. Reads from the on-chip RAM are disabled.
Write Program Memory	0AH	–	Disabled
Read Program Memory	0BH	–	Disabled
Write Data Memory	0CH	–	Disabled
Read Data Memory	0DH	–	Disabled
Read Program Memory CRC	0EH	–	–
Reserved	0FH	–	–
Step Instruction	10H	–	Disabled
Stuff Instruction	11H	–	Disabled
Execute Instruction	12H	–	Disabled
Write Line Control Register	18H	–	–
Read Line Control Register	19H	–	–
Read Baud Reload Register	1BH	–	–

Table 165. On-Chip Debugger Commands (Continued)

Debug Command	Command Byte	Enabled when NOT in DEBUG mode?	Disabled by Read Protect Option Bit
Write Test Mode Register	F0H	–	Flash Test mode is not be enabled if the Information Area Write Protect Option Bit is enabled.
Read Test Mode Register	F1H	–	–
Write Option Bit Registers	F2H	–	Cannot write the Read Protect or Information Area Write Protect Option Bits.
Read Option Bit Registers	F3H	–	–
Note: Unlisted command byte values are reserved.			

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

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

DBG ← 00H

DBG → REVID[15:8] (Major revision number)

DBG → REVID[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 0, or generate a BRK when its value matches the Program Counter. Because this register is really a down counter, the returned value is inverted when this register is read so the returned result appears to be an up counter. If the device is not in DEBUG mode, this command returns FFFFH.

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

- **Write OCD Control Register (04H)**—The Write OCD Control Register command writes the data that follows to the OCDCTL register.

```
DBG ← 04H
DBG ← OCDCTL[7:0]
```

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

```
DBG ← 05H
DBG → OCDCTL[7:0]
```

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

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

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

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

- **Write Register (08H)**—The Write Register command writes data to the Register File. Data can be written 1–256 bytes at a time (256 bytes can be written by setting size to 0). If the device is not in DEBUG mode, the address and data values are discarded. If the Read Protect Option Bit is enabled, then only writes to the on-chip peripheral 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). If the device is not in DEBUG mode or if the Read Protect Option Bit is enabled and on-chip RAM is being read from, 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 0). The on-chip Flash Controller must be written 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 0). 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
```

- **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 is 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 to 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 ( $x^{16} + x^{12} + x^5 + 1$ ). The CRC is preset to all 1s. The least-significant bit of the data is shifted through the polynomial first. The CRC is inverted when it is transmitted. 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 (PC) 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
```

- **Write Line Control Register (18H)**—The Write Line Control Register command writes the data that follows to the Line Control register.

```
DBG <-- 18H
DBG <-- LCR[7:0]
```

- **Read Line Control Register (19H)**—The Read Line Control Register command returns the current value in the Line Control register.

```
DBG <-- 19H
DBG --> LCR[7:0]
```

- **Read Baud Reload Register (1BH)**—The Read Baud Reload Register command returns the current value in the Baud Reload register.

```
DBG ← 1BH
DBG → BAUD[15:8]
DBG → BAUD[7:0]
```

- **Write Test Mode Register (F0H)**—The Write Test Mode Register command writes the data that follows to the Test Mode register.

```
DBG <-- F0H
DBG <-- TESTMODE[7:0]
```

- **Read Test Mode Register (F1H)**—The Read Test Mode Register command returns the current value in the Test Mode register.

```
DBG <-- F1H
DBG --> TESTMODE[7:0]
```

- **Write Option Bit Registers (F2H)**—The Write Option Bit Registers command is used to write to the Option Bit configuration registers. The Option Bit configuration registers store the device configuration and are loaded from Flash every time the Z8 Encore! XP F1680 Series is reset. The registers may be temporarily written using this OCD command to test peripherals without having to program the Flash Information Area and resetting the Z8 Encore! XP F1680 Series. The ZilogUserSel bit selects between Zilog Option Bits (1) and user Option Bits (0).

```
DBG <-- F2H
DBG <-- {ZilogUserSel, 1'b0, OptAddr[4:0]}
DBG <-- OptData[7:0]
```

- **Read Option Bit Registers (F3H)**—The Read Option Bit Registers command is used to read the Option Bit registers that store the device configuration that is read out of flash when the Z8 Encore! XP F1680 Series is reset. The ZilogUserSel bit selects between reading Zilog Option Bits (1) or user Option Bits (0).

```
DBG <-- F1H
DBG <-- {ZilogUserSel, 1'b0, OptAddr[4:0]}
DBG --> OptData[7:0]
```

## On-Chip Debugger Control Register Definitions

### OCD Control Register

The OCD Control register (Table 166 on page 299) controls the state of the On-Chip Debugger. This register is used to enter or exit DEBUG mode and to enable the BRK instruction. It can also reset the Z8 Encore! XP F1680 Series device.

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

**Table 166. OCD Control Register (OCDCTL)**

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

#### **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 resume execution. This bit is automatically set when a BRK instruction is decoded and Breakpoints are enabled.

0 = The device is running (operating in NORMAL mode).

1 = The device is in DEBUG mode.

#### **BRKEN—Breakpoint Enable**

This bit controls the behavior of BRK instruction (opcode 00H). By default, Breakpoints are disabled and the BRK instruction behaves like a NOP. If this bit is set to 1 and a BRK instruction is decoded, the OCD takes action depending upon the BRKLOOP 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 a Debug Acknowledge character (FFH) to the host when a Breakpoint occurs. This bit automatically clears itself when an acknowledge character is sent.

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 and breakpoints are enabled (BRKEN is 1). If this bit is 0, the DBGMODE bit is automatically set to 1 and the OCD enters DEBUG mode. If BRKLOOP is set to 1, 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—Must be 0.**

**RST—Reset**

Setting this bit to 1 resets the device. The controller goes through a normal POR sequence with the exception that the On-Chip Debugger is not reset. This bit is automatically cleared to 0 when the reset finishes.

0 = No effect.

1 = Reset the device.

## OCD Status Register

The OCD Status register (Table 167) reports status information about the current state of the debugger and the system.

**Table 167. OCD Status Register (OCDSTAT)**

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

**IDLE—CPU idle**

This bit is set if the part is in Debug mode (DBGMODE is 1) or if a BRK instruction has occurred since the last time OCDCTL was written. This can be used to determine if the CPU is running or if it is idle.

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 (Flash option bit is 1).

1 = The Read Protect Option Bit is enabled (Flash option bit is 0), disabling many OCD commands.

**Reserved—Must be 0.****Line Control Register**

The Line Control register is used to configure the output driver characteristics during transmission. This register is only used in high-speed implementations.

**Table 168. OCD Line Control Register (OCDLCR)**

<b>BITS</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>FIELD</b>	Reset		NBTX	NBEN	TXFC	TXDH	TXD	TXHD
<b>RESET</b>	00		0	0	0	0	0	0
<b>R/W</b>	R		R/W	R/W	R/W	R/W	R/W	R/W

**NBTX—Nine Bit Transmit**

This control bit sets the polarity of the ninth bit when nine bit mode is enabled.

0 = Ninth bit is zero.

1 = Ninth bit is one.

**NBEN—Nine Bit Enable**

This control bit enables nine bit mode. This is useful when transmit flow control using remote start bit is enabled to detect valid characters.

0 = Nine Bit mode disabled.

1 = Nine Bit mode enabled.

**TXFC—Transmit Flow Control**

0 = Transmit Flow Control disabled.

1 = Transmit Flow Control using Remote Start bit.

**TXDH — Transmit Drive High**

0 = Pin is not driven high during 0 to 1 transitions.

1 = Pin is driven high during 0 to 1 transitions.

**TXD—Transmit Drive**

0 = Pin is only driven low during transmission (Open-Drain).

1 = Pin is always driven during transmission.



**TXHD—Transmit High Drive Strength**

0 = Pin output driver is low drive strength.  
1 = Pin output driver is high drive strength.

**Baud Reload Register**

The Baud Reload register contains the measured Auto-Baud value.

**Table 169. Baud Reload Register**

<b>BITS</b>	<b>15</b>	<b>14</b>	<b>13</b>	<b>12</b>	<b>11</b>	<b>10</b>	<b>9</b>	<b>8</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>FIELD</b>	Reserved				RELOAD											
<b>RESET</b>	0H				000H											
<b>R/W</b>	R				R											

**RELOAD—Baud Reload Value**

This value is the measured Auto-Baud value. Its value can be calculated using the following formula

$$\text{RELOAD} = \frac{\text{SYSCLK}}{\text{BAUDRATE}} \times 8$$

# Oscillator Control

## Overview

The Z8 Encore! XP F1680 Series devices use five possible clocking schemes, each user-selectable:

- On-chip precision trimmed RC oscillator
- On-chip oscillator using off-chip crystal or resonator
- On-chip oscillator using external RC network
- External clock drive
- On-chip low precision Watchdog Timer oscillator

In addition, Z8 Encore! XP F1680 Series devices contain:

- A clock failure detection and recovery circuitry allowing continued operation despite a failure of the primary oscillator
- A peripheral clock that allows timers to be clocked directly from an external 32 kHz watch crystal

## Operation

This chapter discusses the logic used to select the system clock and handle primary oscillator failures. A description of the specific operation of each oscillator is outlined elsewhere in this document. The detailed description of the [Watchdog Timer Oscillator](#) starts on page 137, the [Internal Precision Oscillator](#) description starts on page 315, and the chapter outlining the [Crystal Oscillator](#) begins on page 309.

## System Clock Selection

The oscillator control block is selected from the available clocks. [Table 170](#) on page 304 details each clock source and its usage.

**Table 170. Oscillator Configuration and Selection**

Clock Source	Characteristics	Required Setup
Internal Precision RC Oscillator	<ul style="list-style-type: none"> <li>Selectable frequency 0.0432 MHz, 0.0864 MHz, 0.3456 MHz, 0.6912 MHz, 1.3842 MHz, 2.7648 MHz, 5.5296 MHz, and 11.0592 MHz</li> <li>± 4% accuracy when trimmed</li> <li>No external components required</li> </ul>	<ul style="list-style-type: none"> <li>Unlock and write Oscillator Control Register (OSCCTL0) to enable and select oscillator frequency.</li> </ul>
External Crystal/Resonator	<ul style="list-style-type: none"> <li>32 kHz to 20 MHz</li> <li>Very high accuracy (dependent on crystal or resonator used)</li> <li>External components required</li> </ul>	<ul style="list-style-type: none"> <li>Configure Flash option bits for correct external oscillator mode</li> <li>Unlock and write OSCCTL0 to enable crystal oscillator, wait for it to stabilize and select as system clock (if the EXTL_AO option bit has been de-asserted, no waiting is required)</li> </ul>
External RC Oscillator	<ul style="list-style-type: none"> <li>32 kHz to 4 MHz</li> <li>Accuracy dependent on external components</li> </ul>	<ul style="list-style-type: none"> <li>Configure Flash option bits for correct external oscillator mode</li> <li>Unlock and write OSCCTL0 to enable crystal oscillator and select as system clock</li> </ul>
External Clock Drive	<ul style="list-style-type: none"> <li>0 to 20 MHz</li> <li>Accuracy dependent on external clock source</li> </ul>	<ul style="list-style-type: none"> <li>Write GPIO registers to configure PB3 pin for external clock function</li> <li>Unlock and write OSCCTL0 to select external system clock</li> <li>Apply external clock signal to GPIO</li> </ul>
Internal WDT Oscillator	<ul style="list-style-type: none"> <li>10 kHz nominal</li> <li>± 40% accuracy; no external components required</li> <li>Low power consumption</li> </ul>	<ul style="list-style-type: none"> <li>Enable WDT if not enabled and wait until WDT Oscillator is operating.</li> <li>Unlock and write Oscillator Control Register (OSCCTL0) to enable and select oscillator</li> </ul>



**Caution:** *Unintentional accesses to the oscillator control register can actually stop the chip by switching to a non-functioning oscillator. To prevent this condition, the oscillator control block employs a register unlocking/locking scheme.*

### OSC Control Register Unlocking/Locking

To write the oscillator control register (OSCCTL0 and OSCCTL1), unlock it by making two writes to the OSCCTLx register with the values `E7H` followed by `18H`. A third write to the OSCCTLx register changes the value of the actual register and returns the register to a locked state. Any other sequence of oscillator control register writes has no effect. The values written to unlock the register must be ordered correctly, but are not necessarily consecutive. It is possible to write to or read from other registers within the unlocking/locking operation.

When selecting a new clock source, the primary oscillator failure detection circuitry and the Watchdog Timer oscillator failure circuitry must be disabled. If POFEN and WOFEN are not disabled prior to a clock switch-over, it is possible to generate an interrupt for a failure of either oscillator. The Failure detection circuitry can be enabled anytime after a successful write of SCKSEL in the oscillator control register.

The internal precision oscillator is enabled by default. If the user code changes to a different oscillator, it may be appropriate to disable the IPO for power savings. Disabling the IPO does not occur automatically.

## Clock Failure Detection and Recovery

### Primary Oscillator Failure

The Z8 Encore! XP F1680 Series devices can generate non-maskable interrupt-like events when the primary oscillator fails. To maintain system function in this situation, the clock failure recovery circuitry automatically forces the Watchdog Timer oscillator to drive the system clock. The Watchdog Timer oscillator must be enabled to allow the recovery. Although this oscillator runs at a much slower speed than the original system clock, the CPU continues to operate allowing execution of a clock failure vector and software routines that either remedy the oscillator failure or issue a failure alert. This automatic switch-over is not available, if the Watchdog Timer is the primary oscillator. It is also unavailable if the Watchdog Timer oscillator is disabled, though it is not necessary to enable the Watchdog Timer reset function outlined in [Watchdog Timer](#) chapter of this document on page 137.

The primary oscillator failure detection circuitry asserts if the system clock frequency drops below 1 kHz  $\pm 50\%$ . If an external signal is selected as the system oscillator, it is possible that a very slow but non-failing clock can generate a failure condition. Under these conditions, do not enable the clock failure circuitry (POFEN must be removed from the OSCCTL0 register).

### Watchdog Timer Failure

In the event of a Watchdog Timer oscillator failure, a similar non-maskable interrupt-like event is issued. This event does not trigger an attendant clock switch-over, but alerts the CPU of the failure. After a Watchdog Timer failure, it is no longer possible to detect a primary oscillator failure. The failure detection circuitry does not function if the Watchdog Timer is used as the primary oscillator or if the Watchdog Timer oscillator has been disabled. For either of these cases, it is necessary to disable the detection circuitry by removing the WDFEN bit of the OSCCTL0 register.

The Watchdog Timer oscillator failure-detection circuit counts system clocks while looking for a Watchdog Timer clock. The logic counts 8004 system clock cycles before determining that a failure has occurred. The system clock rate determines the speed at which the Watchdog Timer failure can be detected. A very slow system clock results in very slow detection times.



**Caution:** *It is possible to disable the clock failure detection circuitry as well as all functioning clock sources. In this case, the Z8 Encore! XP F1680 Series device ceases functioning and can only be recovered by Power-on reset.*

## Peripheral Clock

The peripheral clock is based on a low-frequency/low-power 32 kHz secondary oscillator that can be used with an external watch crystal. The peripheral clock is only available for driving Timer and associated noise filter operation. It is not supported for other peripherals. The dedicated peripheral clock source allows Timer operation when the device is in STOP Mode.

Table 171 summarizes peripheral clock source features and usage.

**Table 171. Peripheral Clock Source and Usage**

Peripheral Clock Source	Characteristics	Required Set-Up
Secondary Oscillator	<ul style="list-style-type: none"> <li>- Optimized for use with a 32 kHz Watch Crystal</li> <li>- Very high Accuracy</li> <li>- Dedicated XTAL pins</li> <li>- No external components</li> </ul>	<ul style="list-style-type: none"> <li>- Unlock and write OSCCTL1 to enable secondary oscillator</li> <li>- Select peripheral clock at Timer clock source in TxCTL2 register</li> </ul>

## Oscillator Control Register Definitions

### Oscillator Control0 Register

The Oscillator Control Register (OSCCTL0) enables/disables the various oscillator circuits, enables/disables the failure detection/recovery circuitry, and selects the primary oscillator, which becomes the system clock.

The Oscillator Control 0 Register must be unlocked before writing. Writing the two step sequence `E7H` followed by `18H` to the Oscillator Control 0 Register unlocks it. The register is locked at successful completion of a register write to the OSCCTL0.

**Table 172. Oscillator Control0 Register (OSCCTL0)**

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

**INTEN—Internal Precision Oscillator Enable**

1 = Internal precision oscillator is enabled.  
0 = Internal precision oscillator is disabled.

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

1 = Crystal oscillator is enabled.  
0 = Crystal oscillator is disabled.

**WDTEN—Watchdog Timer Oscillator Enable**

1 = Watchdog Timer oscillator is enabled.  
0 = Watchdog Timer oscillator is disabled.

**POFEN—Primary Oscillator Failure Detection Enable**

1 = Failure detection and recovery of primary oscillator is enabled.  
0 = Failure detection and recovery of primary oscillator is disabled.

**WDFEN—Watchdog Timer Oscillator Failure Detection Enable**

1 = Failure detection of Watchdog Timer oscillator is enabled.  
0 = Failure detection of Watchdog Timer oscillator is disabled.

**SCKSEL—System Clock Oscillator Select**

000 = Internal precision oscillator functions as system clock.  
001 = Reserved.  
010 = Crystal oscillator or external RC oscillator functions as system clock.  
011 = Watchdog Timer oscillator functions as system.  
100 = External clock signal on PB3 functions as system clock.  
101 = Reserved.  
110 = Reserved.  
111 = Reserved.

► **Note:** *The "INTEN" should be disabled when you use other clocks as system clock.*

## Oscillator Control1 Register

The Oscillator Control1 Register (OSCCTL1) enables/disables the secondary oscillator circuits which becomes the peripheral clock. The Oscillator Control1 Register is also used to select the internal precision-oscillator frequency.

The Oscillator Control 1 Register must be unlocked before writing. Writing the two step sequence `E7H` followed by `18H` to the Oscillator Control 1 Register unlocks it. The register is locked at successful completion of a register write to the OSCCTL1.

**Table 173. Oscillator Control1 Register (OSCCTL1)**

<b>BITS</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>FIELD</b>	SECEN	SECRDY	Reserved			INTSEL		
<b>RESET</b>	0	0	1	0	0	0	0	0
<b>R/W</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>ADDR</b>	F87H							

**SECEN—Secondary Oscillator Enable**

1 = 32 kHz Secondary Oscillator is enabled.

0 = 32 kHz Secondary Oscillator is disabled.

**SECRDY—Secondary Oscillator Ready Flag**

1 = 32 kHz Secondary Oscillator is stable and running.

0 = 32 kHz Secondary Oscillator is not running.

**Reserved—must be 0**

**INTSEL—Internal Precision Oscillator Frequency Select**

000 = Internal Precision Oscillator Frequency is 11.0592 MHz.

001 = Internal Precision Oscillator Frequency is 5.5296 MHz.

010 = Internal Precision Oscillator Frequency is 2.7648 MHz.

011 = Internal Precision Oscillator Frequency is 1.3824 MHz.

100 = Internal Precision Oscillator Frequency is 0.6912 MHz.

101 = Internal Precision Oscillator Frequency is 0.3456 MHz.

110 = Internal Precision Oscillator Frequency is 0.0864 MHz.

111 = Internal Precision Oscillator Frequency is 0.0432 MHz.

# Crystal Oscillator

## Overview

The products in the Z8 Encore! XP F1680 Series contain a main on-chip crystal oscillator for use with external crystals with 1 MHz to 20 MHz frequencies, and a secondary 32 K crystal oscillator. In addition, the external oscillator supports external RC networks with oscillation frequencies up to 4 MHz. 32 K secondary crystal oscillator has no external RC oscillator mode. The on-chip crystal oscillator can be used to generate the primary system clock for the internal eZ8 CPU and the majority of the on-chip peripherals. And the secondary 32 K crystal oscillator can only be used to generate clock for three timers.

Alternatively, the  $X_{IN}$  and  $X2_{IN}$  input pin can also accept a CMOS-level clock input signal (for  $X_{IN}$ , 32 kHz – 20 MHz; for  $X2_{IN}$ , below 4 MHz). If an external clock generator is used, the  $X_{OUT}$  or  $X2_{OUT}$  pin must be left unconnected. The Z8 Encore! XP F1680 Series products do not contain an internal clock divider. The frequency of the signal on the  $X_{IN}$  input pin determines the frequency of the system clock, and the frequency of the signal on the  $X2_{IN}$  determines the frequency of timers.

► **Note:** *Although the  $X_{IN}$  pin can be used as an main system clock input for an external clock generator, the  $CLKIN$  pin is better suited for such use (see [System Clock Selection](#) on page 303).*

## Operating Modes

The main on-chip crystal oscillator supports three oscillator modes:

- Medium power for use with medium frequency crystals or ceramic resonators (1 MHz to 8 MHz).
- Maximum power for use with high frequency crystals (8 MHz to 20 MHz).
- On-chip oscillator configured for use with external RC networks or external clock input (<4 MHz).

The main on-chip crystal oscillator mode is selected using user-programmable Flash Option Bits. For information, see [Flash Option Bits](#) on page 267.

The secondary 32 K crystal oscillator supports two oscillator modes:

- NORMAL mode for use with 32 K crystals (32 kHz).
- On-chip oscillator configured for use with external clock input.

The secondary 32 K crystal oscillator mode is selected using user-programmable Flash Option Bits. For information, see [Flash Option Bits](#) on page 267.

## Main Crystal Oscillator Operation

The Flash Option bit XTLDIS controls whether the crystal oscillator is enabled during reset. The crystal may later be disabled after reset if a new oscillator has been selected as the system clock. If the crystal is manually enabled after reset through the OSCCTL register, the user code must wait at least 1000 crystal oscillator cycles for the crystal to stabilize. After this, the crystal oscillator may be selected as the system clock.

Figure 61 illustrates 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 174. Printed circuit board layout must add no more than 4 pF of stray capacitance to either the X<sub>IN</sub> or X<sub>OUT</sub> pins. If oscillation does not occur, reduce the values of capacitors C<sub>1</sub> and C<sub>2</sub> to decrease loading.

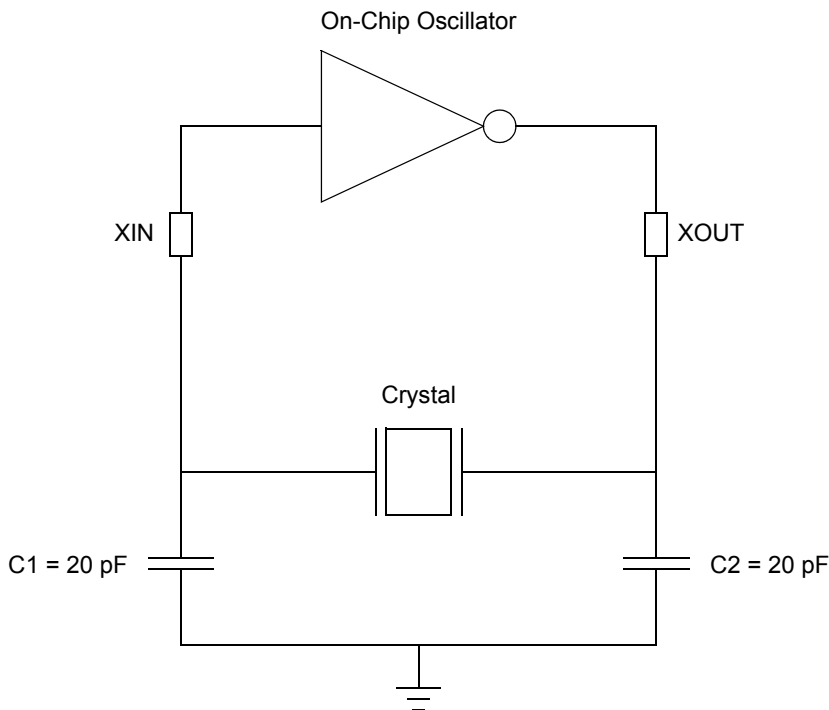


Figure 61. Recommended 20 MHz Crystal Oscillator Configuration

Table 174. Recommended Crystal Oscillator Specifications

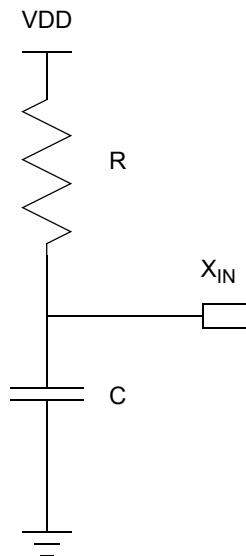
Parameter	Value	Units	Comments
Frequency	20	MHz	
Resonance	Parallel		

**Table 174. Recommended Crystal Oscillator Specifications (Continued)**

Parameter	Value	Units	Comments
Mode	Fundamental		
Series Resistance ( $R_S$ )	60	$\Omega$	Maximum
Load Capacitance ( $C_L$ )	30	pF	Maximum
Shunt Capacitance ( $C_0$ )	7	pF	Maximum
Drive Level	1	mW	Maximum

## Main Oscillator Operation with External RC Network

Figure 62 illustrates a recommended configuration for connection with an external resistor-capacitor (RC) network.



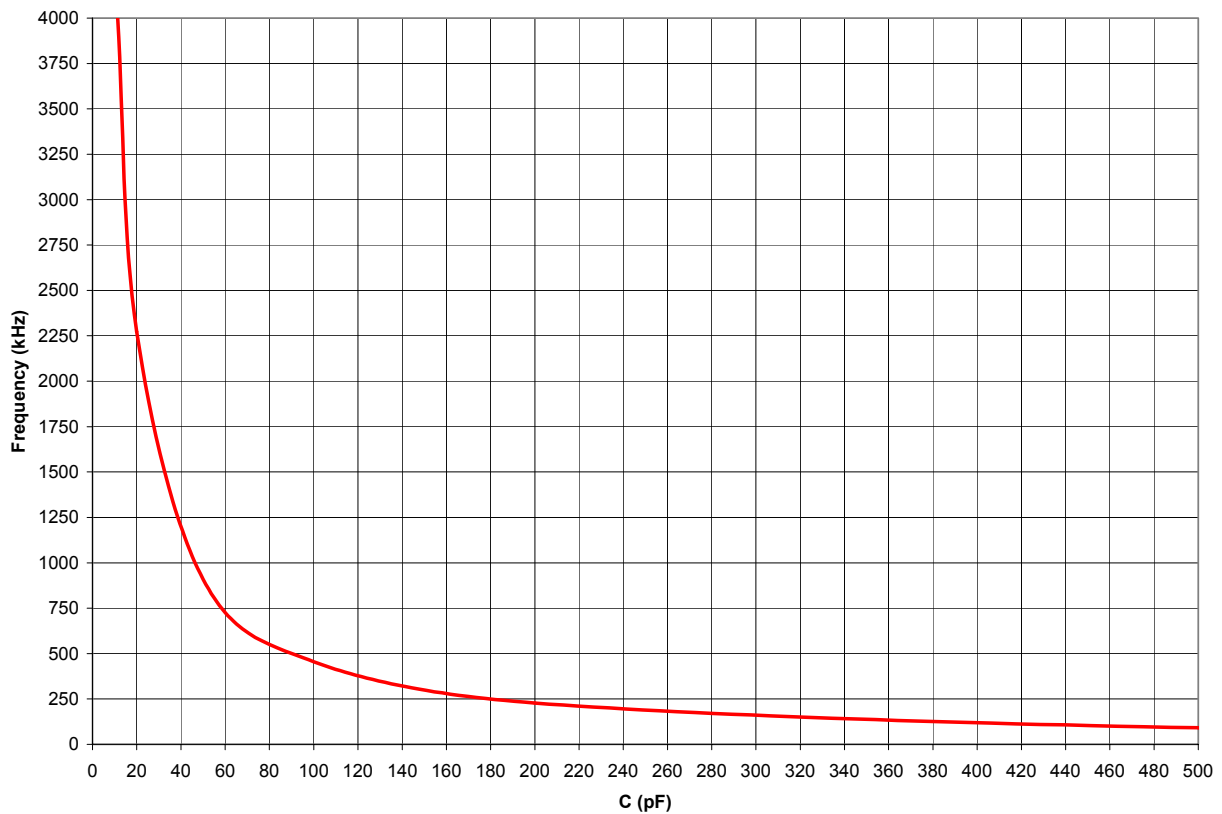
**Figure 62. Connecting the On-Chip Oscillator to an External RC Network**

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

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

Figure 63 illustrates the typical (3.3 V and 25 °C) oscillator frequency as a function of the capacitor ( $C$  in pF) employed in the RC network assuming a 45 K $\Omega$  external resistor. For very small values of  $C$ , the parasitic capacitance of the oscillator  $X_{IN}$  pin and the printed circuit board should be included in the estimation of the oscillator frequency.

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



**Figure 63. Typical RC Oscillator Frequency as a Function of the External Capacitance with a 45 K $\Omega$  Resistor**



**Caution:** *When using the external RC oscillator mode, the oscillator can stop oscillating if the power supply drops below 1.6 V, but before the power supply drops to the Voltage Brownout threshold. The oscillator resumes oscillation when the supply voltage exceeds 1.6 V.*

## Secondary Crystal Oscillator Operation

Figure 64 illustrates a recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 32 kHz. The recommended 32 kHz crystal specifications are provided in Table 175. Printed circuit board layout must add no more than 4 pF of stray capacitance to either the X<sub>IN</sub> or X<sub>OUT</sub> pins. If oscillation does not occur, reduce the values of capacitors C<sub>1</sub> and C<sub>2</sub> to decrease loading.

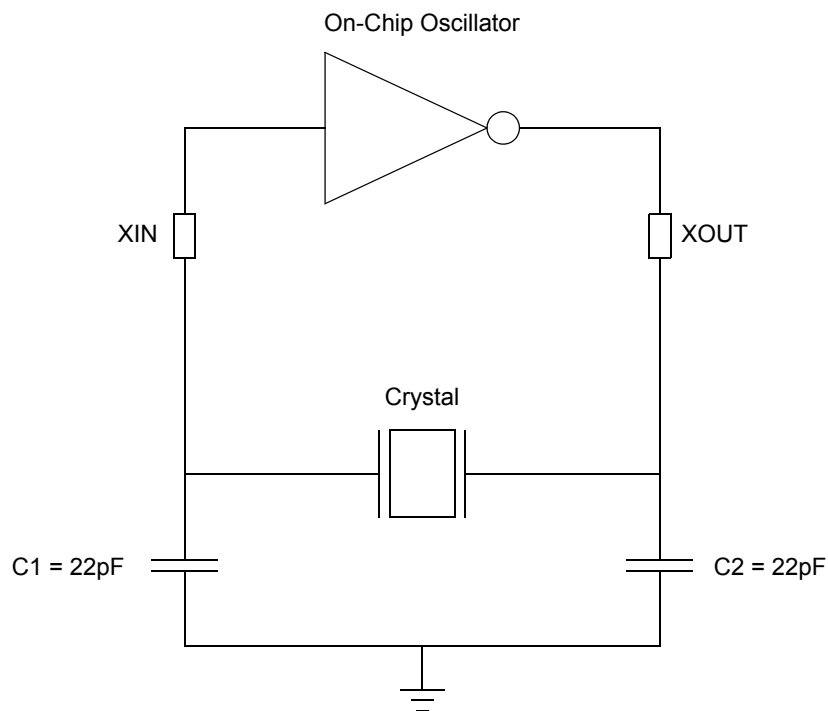


Figure 64. Recommended 32 kHz Crystal Oscillator Configuration

Table 175. Recommended Crystal Oscillator Specifications

Parameter	Value	Units	Comments
Frequency	32	kHz	
Resonance	Serial		
Mode	Fundamental		
Series Resistance (R <sub>S</sub> )	160K	W	Maximum
Load Capacitance (C <sub>L</sub> )	30	pF	Maximum
Shunt Capacitance (C <sub>0</sub> )	5	pF	Maximum
Drive Level	0.1	mW	Maximum



# Internal Precision Oscillator

## Overview

The Internal Precision Oscillator (IPO) is designed for use without external components. You can either manually trim the oscillator for a non-standard frequency or use the automatic factory-trimmed version to achieve a 0.0432–11.0592 MHz frequency. IPO features include:

- On-chip RC oscillator that does not require external components
- Selectable output frequency: 11.0592 MHz, 5.5296 MHz, 2.7648 MHz, 1.3824 MHz, 0.6912 MHz, 0.3456 MHz, 0.0864 MHz, and 0.0432 MHz.
- Trimming possible through Flash-option bits with user override
- Elimination of crystals or ceramic resonators in applications where high timing accuracy is not required
- Accuracy:  $\pm 4\%$  over temperature and voltage

## Operation

The internal oscillator is an RC relaxation oscillator that has its sensitivity to power supply variation minimized. By using ratio tracking thresholds, the effect of power supply voltage is cancelled out. The dominant source of oscillator error is the absolute variance of chip level fabricated components, like capacitors. An 8-bit trimming register incorporated into the design compensates for absolute variation of oscillator frequency. Once trimmed, the oscillator frequency is stable and does not require subsequent calibration. Trimming is performed during manufacturing and is not necessary for you to repeat unless a frequency other than one of the selectable frequencies is required. Power down this block for minimum system power.

By default, the oscillator is configured through the Flash Option bits. However, the user code can override these trim values as described in [Trim Bit Address Space](#) on page 272.

Select one of the frequencies for the oscillator using the INTSEL bits in the [Oscillator Control1 Register](#) on page 307.



# eZ8 CPU Instruction Set

## Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without concern for 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 contains labels, operations, operands, and comments.

Labels are 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, #01  ; Another Load (LD) instruction with two operands.
              ; The first operand, Extended Mode Register Address 234H,
              ; identifies 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 and 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. You must follow this binary format if you prefer manual program coding or intend to implement your 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 as listed in [Table 176](#).

**Table 176. Assembly Language Syntax Example 1**

<b>Assembly Language Code</b>	ADD	43H,	08H	(ADD dst, src)
<b>Object Code</b>	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 as listed in [Table 177](#).

**Table 177. Assembly Language Syntax Example 2**

<b>Assembly Language Code</b>	ADD	43H,	R8	(ADD dst, src)
<b>Object Code</b>	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 section [eZ8 CPU Instruction Summary](#) on page 325, the operands, condition codes, status flags, and address modes are represented by a notational shorthand provided in [Table 178](#).

Table 178. 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 <i>eZ8 CPU User Manual</i>
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 179 contains additional symbols that are used throughout the section [eZ8 CPU Instruction Summary](#) on page 325.

**Table 179. 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 that the source data is added to the destination data and the result is stored in the destination location.

## eZ8 CPU Instruction Classes

eZ8 CPU instructions is divided functionally into the following groups:

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Table 180 through Table 187 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 instructions are to 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 180. 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
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 181. 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 182. 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 183. CPU Control Instructions**

Mnemonic	Operands	Instruction
ATM	—	Atomic Execution
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 184. Load Instructions**

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
CLR	dst	Clear
LD	dst, src	Load
LDC	dst, src	Load Constant to/from Program Memory
LDCI	dst, src	Load Constant to/from Program Memory and Auto-Increment Addresses
LDE	dst, src	Load External Data to/from Data Memory
LDEI	dst, src	Load External Data to/from Data Memory and Auto-Increment Addresses
LDWX	dst, src	Load Word using Extended Addressing
LDX	dst, src	Load using Extended Addressing
LEA	dst, X(src)	Load Effective Address
POP	dst	Pop
POPX	dst	Pop using Extended Addressing
PUSH	src	Push
PUSHX	src	Push using Extended Addressing

**Table 185. Logical Instructions**

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
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 186. 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 187. 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 188 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 188. 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	dst ← dst + src + 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	dst ← dst + src + C	ER	ER	18	*	*	*	*	0	*	4	3
		ER	IM	19							4	3
ADD dst, src	dst ← dst + 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	dst ← dst + src	ER	ER	08	*	*	*	*	0	*	4	3
		ER	IM	09							4	3
AND dst, src	dst ← dst AND src	r	r	52	–	*	*	0	–	–	2	3
		r	lr	53							2	4
		R	R	54							3	3
		R	IR	55							3	4
		R	IM	56							3	3
		IR	IM	57							3	4
Flags Notation:		* = Value is a function of the result of the operation. – = Unaffected X = Undefined			0 = Reset to 0 1 = Set to 1							

Table 188. 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		
ANDX dst, src	dst ← dst AND src	ER	ER	58	–	*	*	0	–	–	4	3
		ER	IM	59							4	3
ATM	Block all interrupt and DMA requests during execution of the next 3 instructions			2F	–	–	–	–	–	–	1	2
BCLR bit, dst	dst[bit] ← 0	r		E2	–	*	*	0	–	–	2	2
BIT p, bit, dst	dst[bit] ← p	r		E2	–	*	*	0	–	–	2	2
BRK	Debugger Break			00	–	–	–	–	–	–	1	1
BSET bit, dst	dst[bit] ← 1	r		E2	–	*	*	0	–	–	2	2
BSWAP dst	dst[7:0] ← dst[0:7]	R		D5	X	*	*	0	–	–	2	2
BTJ p, bit, src, dst	if src[bit] = p PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			lr	F7							3	4
BTJNZ bit, src, dst	if src[bit] = 1 PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			lr	F7							3	4
BTJZ bit, src, dst	if src[bit] = 0 PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			lr	F7							3	4
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
Flags Notation:	* = Value is a function of the result of the operation. – = Unaffected X = Undefined				0 = Reset to 0 1 = Set to 1							

Table 188. 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		
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
DECW dst	dst ← dst - 1	RR		80	–	*	*	*	–	–	2	5
		IRR		81							2	6
DI	IRQCTL[7] ← 0			8F	–	–	–	–	–	–	1	2
DJNZ dst, RA	dst ← dst – 1 if dst ≠ 0 PC ← PC + X	r		0A-FA	–	–	–	–	–	–	2	3
EI	IRQCTL[7] ← 1			9F	–	–	–	–	–	–	1	2
HALT	Halt Mode			7F	–	–	–	–	–	–	1	2
Flags Notation:	* = Value is a function of the result of the operation. – = Unaffected X = Undefined				0 = Reset to 0 1 = Set to 1							

Table 188. 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		
INC dst	dst ← dst + 1	R		20	–	*	*	–	–	–	2	2
		IR		21							2	3
		r		0E-FE							1	2
INCW dst	dst ← dst + 1	RR		A0	–	*	*	*	–	–	2	5
		IRR		A1							2	6
IRET	FLAGS ← @SP SP ← SP + 1 PC ← @SP SP ← SP + 2 IRQCTL[7] ← 1			BF	*	*	*	*	*	*	1	5
JP dst	PC ← dst	DA		8D	–	–	–	–	–	–	3	2
		IRR		C4							2	3
JP cc, dst	if cc is true PC ← dst	DA		0D-FD	–	–	–	–	–	–	3	2
JR dst	PC ← PC + X	DA		8B	–	–	–	–	–	–	2	2
JR cc, dst	if cc is true PC ← PC + X	DA		0B-FB	–	–	–	–	–	–	2	2
LD dst, rc	dst ← 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	dst ← src	r	lrr	C2	–	–	–	–	–	–	2	5
		lr	lrr	C5							2	9
		lrr	r	D2							2	5
Flags Notation:	* = Value is a function of the result of the operation. – = Unaffected X = Undefined				0 = Reset to 0 1 = Set to 1							

Table 188. 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		
LDCI dst, src	dst ← src r ← r + 1 rr ← rr + 1	lr	lrr	C3	–	–	–	–	–	–	2	9
		lrr	lr	D3							2	9
LDE dst, src	dst ← src	r	lrr	82	–	–	–	–	–	–	2	5
		lrr	r	92							2	5
LDEI dst, src	dst ← src r ← r + 1 rr ← rr + 1	lr	lrr	83	–	–	–	–	–	–	2	9
		lrr	lr	93							2	9
LDWX dst, src	dst ← src	ER	ER	1FE8	–	–	–	–	–	–	5	4
LDX dst, src	dst ← 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)	dst ← src + X	r	X(r)	98	–	–	–	–	–	–	3	3
		rr	X(rr)	99							3	5
MULT dst	dst[15:0] ← dst[15:8] * dst[7:0]	RR		F4	–	–	–	–	–	–	2	8
NOP	No operation			0F	–	–	–	–	–	–	1	2
Flags Notation:	* = Value is a function of the result of the operation. – = Unaffected X = Undefined				0 = Reset to 0 1 = Set to 1							

Table 188. eZ8 CPU Instruction Summary (Continued)

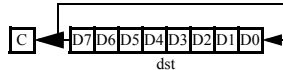
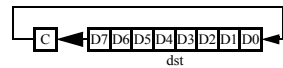
Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
OR dst, src	dst ← dst OR src	r	r	42	–	*	*	0	–	–	2	3
		r	lr	43							2	4
		R	R	44							3	3
		R	IR	45							3	4
		R	IM	46							3	3
		IR	IM	47							3	4
ORX dst, src	dst ← dst OR src	ER	ER	48	–	*	*	0	–	–	4	3
		ER	IM	49							4	3
POP dst	dst ← @SP SP ← SP + 1	R		50	–	–	–	–	–	–	2	2
		IR		51							2	3
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
		IM		IF70							3	2
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
RLC dst		R		10	*	*	*	*	–	–	2	2
		IR		11								2
Flags Notation:	* = Value is a function of the result of the operation.				0 = Reset to 0							
	– = Unaffected											
	X = Undefined				1 = Set to 1							

Table 188. eZ8 CPU Instruction Summary (Continued)

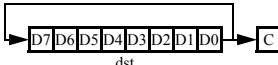
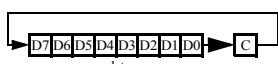
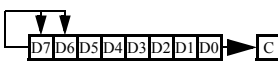

Assembly Mnemonic	Symbolic Operation	Address Mode		Opcode(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
RR dst		R		E0	*	*	*	*	–	–	2	2
		IR		E1							2	3
RRC dst		R		C0	*	*	*	*	–	–	2	2
		IR		C1							2	3
SBC dst, src	$dst \leftarrow dst - src - C$	r	r	32	*	*	*	*	1	*	2	3
		r	lr	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	$dst \leftarrow dst - src - C$	ER	ER	38	*	*	*	*	1	*	4	3
		ER	IM	39							4	3
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
Flags Notation:		* = Value is a function of the result of the operation. – = Unaffected X = Undefined										
		0 = Reset to 0 1 = Set to 1										

Table 188. 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		
SUBX dst, src	dst ← dst – src	ER	ER	28	*	*	*	*	1	*	4	3
		ER	IM	29							4	3
SWAP dst	dst[7:4] ↔ dst[3:0]	R		F0	X	*	*	X	–	–	2	2
		IR		F1							2	3
TCM dst, src	(NOT dst) AND src	r	r	62	–	*	*	0	–	–	2	3
		r	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
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
Flags Notation:	* = Value is a function of the result of the operation. – = Unaffected X = Undefined				0 = Reset to 0 1 = Set to 1							

Table 188. 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		
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							

# Opcode Maps

Figure 65 displays a description of the opcode map data and the abbreviations. Figure 66 and Figure 67 provide information about each of the eZ8 CPU instructions. Table 189 lists Opcode Map abbreviations.

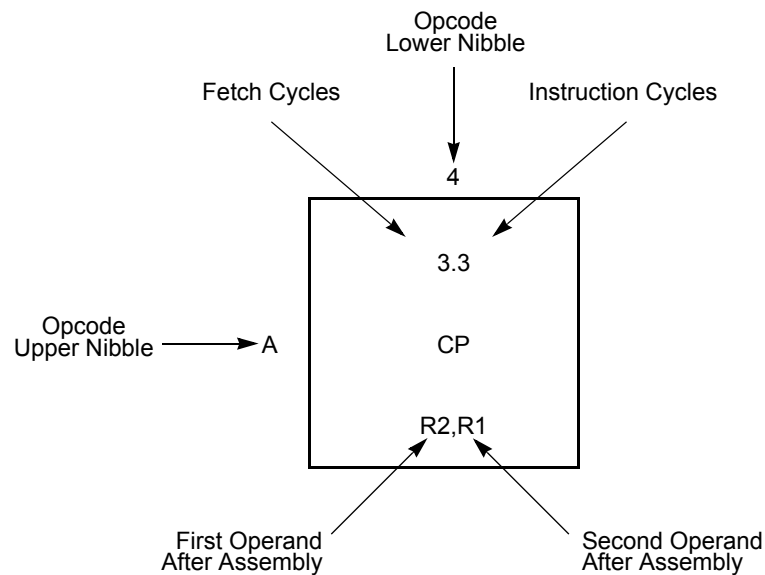


Figure 65. Opcode Map Cell Description

Table 189. 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	1.1 BRK	2.2 SRP IM	2.3 ADD r1,r2	2.4 ADD r1,r2	3.3 ADD R2,R1	3.4 ADD IR2,R1	3.3 ADD R1,IM	3.4 ADD IR1,IM	4.3 ADDX ER2,ER1	4.3 ADDX IM,ER1	2.3 DJNZ r1,X	2.2 JR cc,X	2.2 LD r1,IM	3.2 JP cc,DA	1.2 INC r1	1.2 NOP
	1	2.2 RLC R1	2.3 RLC IR1	2.3 ADC r1,r2	2.4 ADC r1,r2	3.3 ADC R2,R1	3.4 ADC IR2,R1	3.3 ADC R1,IM	3.4 ADC IR1,IM	4.3 ADCX ER2,ER1	4.3 ADCX IM,ER1						See 2nd Opcode Map
	2	2.2 INC R1	2.3 INC IR1	2.3 SUB r1,r2	2.4 SUB r1,r2	3.3 SUB R2,R1	3.4 SUB IR2,R1	3.3 SUB R1,IM	3.4 SUB IR1,IM	4.3 SUBX ER2,ER1	4.3 SUBX IM,ER1						1, 2 ATM
	3	2.2 DEC R1	2.3 DEC IR1	2.3 SBC r1,r2	2.4 SBC r1,r2	3.3 SBC R2,R1	3.4 SBC IR2,R1	3.3 SBC R1,IM	3.4 SBC IR1,IM	4.3 SBCX ER2,ER1	4.3 SBCX IM,ER1						
	4	2.2 DA R1	2.3 DA IR1	2.3 OR r1,r2	2.4 OR r1,r2	3.3 OR R2,R1	3.4 OR IR2,R1	3.3 OR R1,IM	3.4 OR IR1,IM	4.3 ORX ER2,ER1	4.3 ORX IM,ER1						
	5	2.2 POP R1	2.3 POP IR1	2.3 AND r1,r2	2.4 AND r1,r2	3.3 AND R2,R1	3.4 AND IR2,R1	3.3 AND R1,IM	3.4 AND IR1,IM	4.3 ANDX ER2,ER1	4.3 ANDX IM,ER1						1.2 WDT
	6	2.2 COM R1	2.3 COM IR1	2.3 TCM r1,r2	2.4 TCM r1,r2	3.3 TCM R2,R1	3.4 TCM IR2,R1	3.3 TCM R1,IM	3.4 TCM IR1,IM	4.3 TCMX ER2,ER1	4.3 TCMX IM,ER1						1.2 STOP
	7	2.2 PUSH R2	2.3 PUSH IR2	2.3 TM r1,r2	2.4 TM r1,r2	3.3 TM R2,R1	3.4 TM IR2,R1	3.3 TM R1,IM	3.4 TM IR1,IM	4.3 TMX ER2,ER1	4.3 TMX IM,ER1						1.2 HALT
	8	2.5 DECW RR1	2.6 DECW IRR1	2.5 LDE r1,lr2	2.9 LDEI lr1,lr2	3.2 LDX r1,ER2	3.3 LDX lr1,ER2	3.4 LDX IRR2,R1	3.5 LDX IRR2,IR1	3.4 LDX r1,rr2,X	3.4 LDX rr1,r2,X						1.2 DI
	9	2.2 RL R1	2.3 RL IR1	2.5 LDE r2,lr1	2.9 LDEI lr2,lr1	3.2 LDX r2,ER1	3.3 LDX lr2,ER1	3.4 LDX R2,IRR1	3.5 LDX IR2,IRR1	3.3 LEA r1,r2,X	3.5 LEA rr1,rr2,X						1.2 EI
	A	2.5 INCW RR1	2.6 INCW IRR1	2.3 CP r1,r2	2.4 CP r1,r2	3.3 CP R2,R1	3.4 CP IR2,R1	3.3 CP R1,IM	3.4 CP IR1,IM	4.3 CPX ER2,ER1	4.3 CPX IM,ER1						1.4 RET
	B	2.2 CLR R1	2.3 CLR IR1	2.3 XOR r1,r2	2.4 XOR r1,r2	3.3 XOR R2,R1	3.4 XOR IR2,R1	3.3 XOR R1,IM	3.4 XOR IR1,IM	4.3 XORX ER2,ER1	4.3 XORX IM,ER1						1.5 IRET
	C	2.2 RRC R1	2.3 RRC IR1	2.5 LDC r1,lr2	2.9 LDCI lr1,lr2	2.3 JP IRR1	2.9 LDC lr1,lr2		3.4 LD r1,r2,X	3.2 PUSHX ER2							1.2 RCF
	D	2.2 SRA R1	2.3 SRA IR1	2.5 LDC r2,lr1	2.9 LDCI lr2,lr1	2.6 CALL IRR1	2.2 BSWAP R1	3.3 CALL DA	3.4 LD r2,r1,X	3.2 POPX ER1							1.2 SCF
	E	2.2 RR R1	2.3 RR IR1	2.2 BIT p,b,r1	2.3 LD r1,r2	3.2 LD R2,R1	3.3 LD IR2,R1	3.2 LD R1,IM	3.3 LD IR1,IM	4.2 LDX ER2,ER1	4.2 LDX IM,ER1						1.2 CCF
	F	2.2 SWAP R1	2.3 SWAP IR1	2.6 TRAP Vector	2.3 LD lr1,r2	2.8 MULT RR1	3.3 LD R2,IR1	3.3 BTJ p,b,r1,X	3.4 BTJ p,b,lr1,X								

Figure 66. 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	3, 2 PUSH IM															
	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									5, 4 LDWX ER2,ER1							
	F																

Figure 67. Second Opcode Map after 1FH

# Electrical Characteristics

The data in this chapter is pre-qualification and pre-characterization and is subject to change. Additional electrical characteristics may be found in the individual chapters.

## Absolute Maximum Ratings

Stresses greater than those listed in [Table 190](#) may 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 may affect device reliability. For improved reliability, tie unused inputs to one of the supply voltages ( $V_{DD}$  or  $V_{SS}$ ).

**Table 190. Absolute Maximum Ratings**

Parameter	Min	Max	Units	Notes
Ambient temperature under bias	0	+105	°C	
Storage temperature	-65	+150	°C	
Voltage on any pin with respect to $V_{SS}$	-0.3	+5.5	V	1
Voltage on $V_{DD}$ pin with respect to $V_{SS}$	-0.3	+3.6	V	
Maximum current on input and/or inactive output pin	-5	+5	μA	
Maximum output current from active output pin	-25	+25	mA	
<b>20-pin Packages Maximum Ratings at 0 °C to 70 °C</b>				
Total power dissipation		430	mW	
Maximum current into $V_{DD}$ or out of $V_{SS}$		120	mA	
<b>28-pin Packages Maximum Ratings at 0 °C to 70 °C</b>				
Total power dissipation		450	mW	
Maximum current into $V_{DD}$ or out of $V_{SS}$		125	mA	
<b>40-pin PDIP Maximum Ratings at -40 °C to 70 °C</b>				
Total power dissipation		1000	mW	
Maximum current into $V_{DD}$ or out of $V_{SS}$		275	mA	
<b>40-pin PDIP Maximum Ratings at 70 °C to 105 °C</b>				
Total power dissipation		540	mW	
Maximum current into $V_{DD}$ or out of $V_{SS}$		150	mA	

**Table 190. Absolute Maximum Ratings (Continued)**

Parameter	Min	Max	Units	Notes
<b>44-Pin QFN Maximum Ratings at -40 °C to 70 °C</b>				
Total power dissipation		750	mW	
Maximum current into $V_{DD}$ or out of $V_{SS}$		200	mA	
<b>44-Pin QFN Maximum Ratings at 70 °C to 105 °C</b>				
Total power dissipation		295	mW	
Maximum current into $V_{DD}$ or out of $V_{SS}$		83	mA	
<b>44-pin LQFP Maximum Ratings at -40 °C to 70 °C</b>				
Total power dissipation		750	mW	
Maximum current into $V_{DD}$ or out of $V_{SS}$		200	mA	
<b>44-pin LQFP Maximum Ratings at 70 °C to 105 °C</b>				
Total power dissipation		410	mW	
Maximum current into $V_{DD}$ or out of $V_{SS}$		114	mA	
Operating temperature is specified in DC Characteristics				
Note: <sup>1</sup> This voltage applies to all pins except the following: $V_{DD}$ , $AV_{DD}$ .				

## DC Characteristics

Table 191 lists the DC characteristics of the Z8 Encore! XP F1680 Series products. All voltages are referenced to  $V_{SS}$  which is the primary system ground.

**Table 191. DC Characteristics**

Symbol	Parameter	$T_A = 0\text{ °C to }+70\text{ °C}$ $T_A = -40\text{ °C to }+105\text{ °C}$			Units	Conditions
		Min	Typ	Max		
$V_{DD}$	Supply Voltage	1.8	–	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, XIN
$V_{IL2}$	Low Level Input Voltage	-0.3	–	$0.2 \cdot V_{DD}$	V	For $\overline{\text{RESET}}$ , DBG, XIN
$V_{IH1}$	High Level Input Voltage	$0.7 \cdot V_{DD}$	–	5.5	V	Port A, B, C, D, and E pins (Digital inputs)

Table 191. DC Characteristics (Continued)

Symbol	Parameter	$T_A = 0\text{ }^{\circ}\text{C to }+70\text{ }^{\circ}\text{C}$ $T_A = -40\text{ }^{\circ}\text{C to }+105\text{ }^{\circ}\text{C}$			Units	Conditions
		Min	Typ	Max		
$V_{IH2}$	High Level Input Voltage	$0.7 \cdot V_{DD}$	–	$V_{DD}+0.3$	V	Ports B and C (Analog)
$V_{OL1}$	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	$V_{DD}-0.5$	–	–	V	$I_{OH} = -2\text{ mA}$ ; $V_{DD} = 3.0\text{ V}$ High Output Drive disabled.
$V_{OL2}$	Low Level Output Voltage	–	–	0.6	V	$I_{OL} = 20\text{ mA}$ ; $V_{DD} = 3.3\text{ V}$ High Output Drive enabled.
$V_{OH2}$	High Level Output Voltage	$V_{DD}-0.5$	–	–	V	$I_{OH} = -20\text{ mA}$ ; $V_{DD} = 3.3\text{ V}$ High Output Drive enabled.
$I_{IL}$	Input Leakage Current	-5	–	+5	$\mu\text{A}$	$V_{DD} = 3.6\text{ V}$ ; $V_{IN} = V_{DD}$ or $V_{SS}$ <sup>1</sup>
$I_{TL}$	Tristate Leakage Current	-5	–	+5	$\mu\text{A}$	$V_{DD} = 3.6\text{ V}$
$I_{LED}$	Controlled LED Current Drive	1.5	3	4.5	mA	{AFS2,AFS1} = {0,0}, $V_{DD} = 3.3\text{ V}$
		2.8	7	10.5	mA	{AFS2,AFS1} = {0,1}, $V_{DD} = 3.3\text{ V}$
		7.8	13	19.5	mA	{AFS2,AFS1} = {1,0}, $V_{DD} = 3.3\text{ V}$
		12	20	30	mA	{AFS2,AFS1} = {1,1}, $V_{DD} = 3.3\text{ V}$
$C_{PAD}$	GPIO Port Pad Capacitance	–	$8.0^2$	–	pF	TBD
$C_{XIN}$	XIN Pad Capacitance	–	$8.0^2$	–	pF	TBD
$C_{XOUT}$	XOUT Pad Capacitance	–	$9.5^2$	–	pF	TBD
$I_{PU}$	Weak Pull-up Current	30	100	350	$\mu\text{A}$	$V_{DD} = 3.0 - 3.6\text{ V}$
Notes:						
<sup>1</sup> This condition excludes all pins that have on-chip pull-ups, when driven Low.						
<sup>2</sup> These values are provided for design guidance only and are not tested in production.						

The currents in Table 192 represent the power consumption without any peripherals active (unless otherwise noted). For design guidance, total power consumption will be the sum of all active peripheral currents plus the appropriate current characteristics shown below.

**Table 192. Supply Current Characteristics**

Symbol	Parameter	T <sub>A</sub> = 0 °C to +70 °C T <sub>A</sub> = -40 °C to +105 °C			Units	Conditions <sup>2</sup>
		Min	Typical <sup>1</sup>	Max		
I <sub>DDA1</sub>	Active Mode Device Current Executing from Flash		8.5		mA	Typical: 20 MHz <sup>3, 4, 5, 6</sup> , V <sub>DD</sub> = 3 V, Flash, 25 °C
I <sub>DDA2</sub>	Active Mode Device Current Executing from PRAM		6		mA	Typical: 20 MHz <sup>3, 4, 5, 6</sup> , V <sub>DD</sub> = 3 V, PRAM, 25 °C
I <sub>DDH</sub>	Halt Mode Device Current		TBD		mA	Typical: 20 MHz <sup>3, 4, 5</sup> , V <sub>DD</sub> typical, 25 °C
I <sub>DDS1</sub>	Stop Mode Device Current		2.5		μA	Typical: WDT, V <sub>DD</sub> typical, 25 °C, all peripherals including VBO disabled <sup>3, 4, 6</sup>
I <sub>DDS2</sub>	Stop Mode Device Current		<1		μA	Typical: V <sub>DD</sub> typical, 25 °C, all peripherals disabled including VBO and WDT <sup>3, 4, 6</sup>

**Notes:**

1. These values are provided for design guidance only and are not tested in production.
2. Typical conditions are defined as 3.3 V at 25 °C, unless otherwise noted.
3. All internal pull ups are disabled, and all push-pull outputs are unloaded.
4. All open-drain outputs are pulled up to V<sub>DD</sub>/AV<sub>DD</sub> and are at a high state.
5. System clock source is an external square wave clock signal driven through the CLK-IN pin.
6. All inputs are at V<sub>DD</sub>/AV<sub>DD</sub> or V<sub>SS</sub>/AV<sub>SS</sub> as appropriate.

Figure 68 through Figure 71 illustrate the typical current consumption at voltages of 1.8 V, 2.0 V, 2.7 V, 3.0 V, 3.3 V, and 3.6 V versus different system clock frequencies while operating at a temperature of 25 °C.

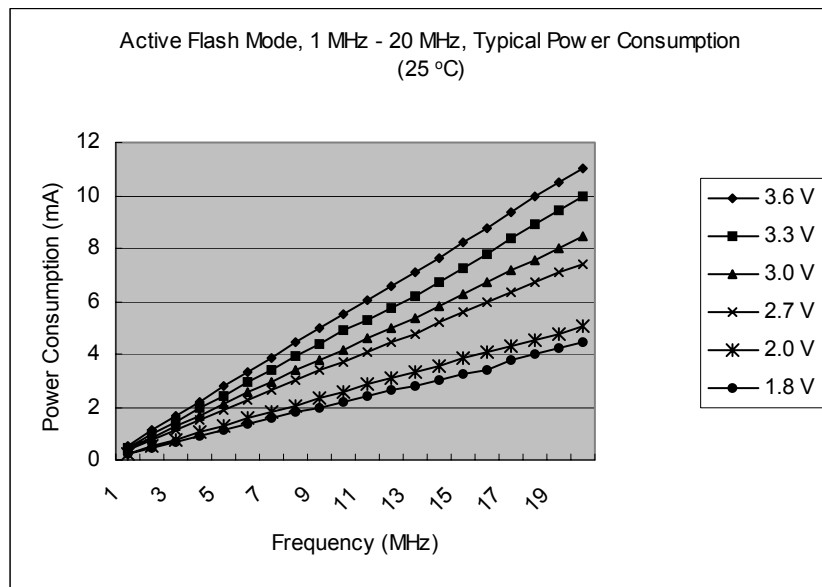


Figure 68. Typical Active Flash Mode Supply Current (1 – 20 MHz)

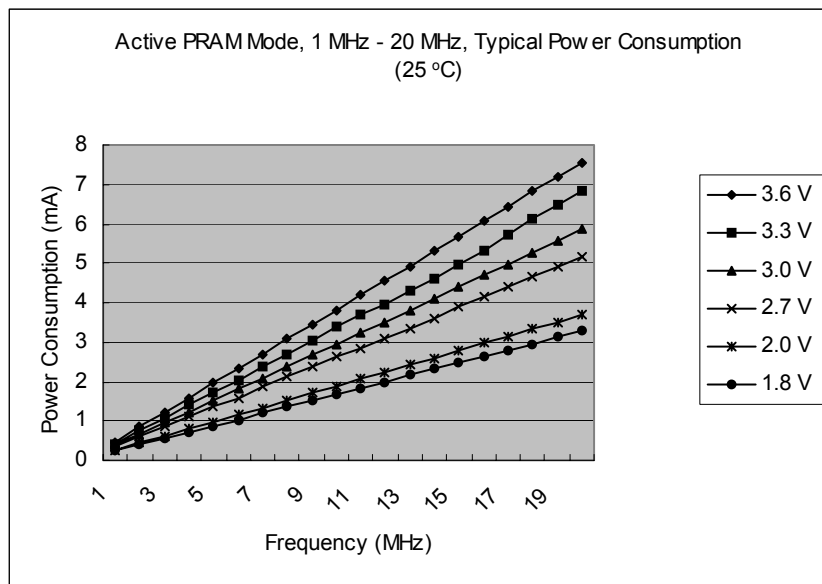


Figure 69. Typical Active PRAM Mode Supply Current (1 – 20 MHz)

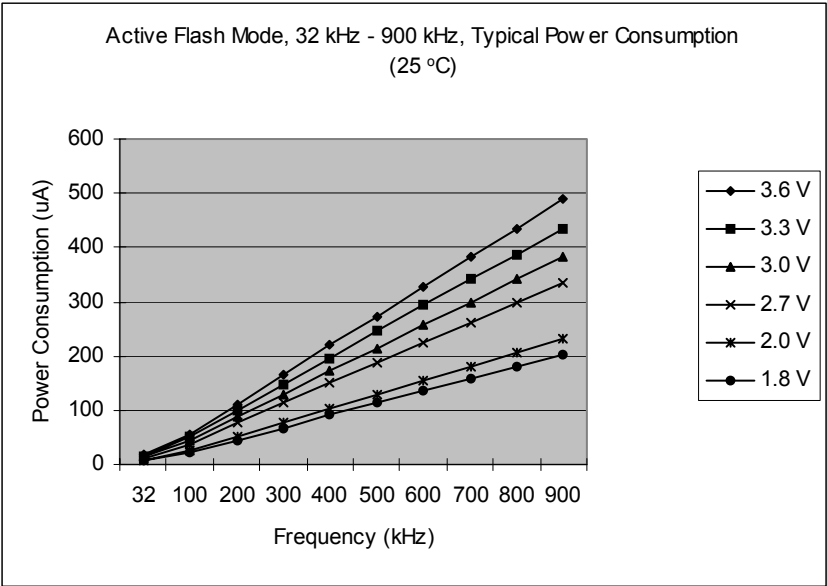


Figure 70. Typical Active Flash Mode Supply Current (32 – 900 kHz)

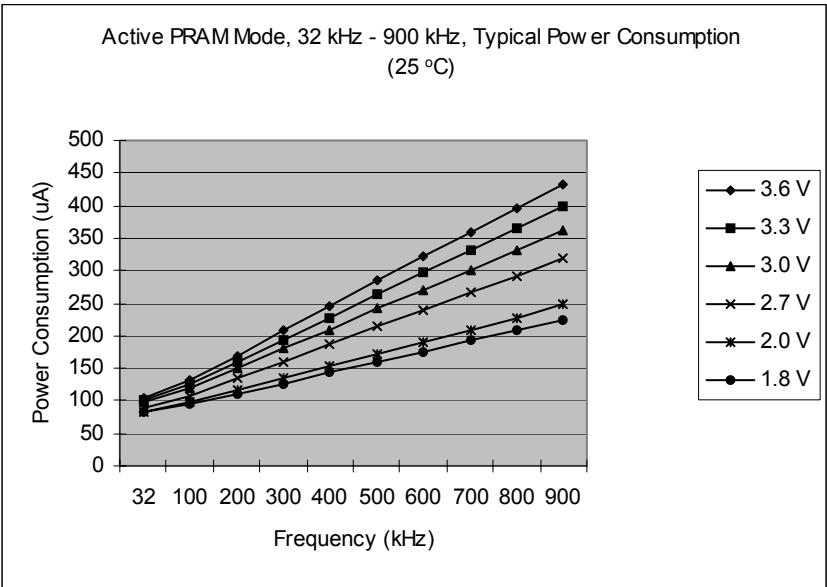


Figure 71. Typical Active PRAM Mode Supply Current (32–900 kHz)

Figure 72 on on page 343 illustrates the STOP mode supply current versus ambient temperature and Vdd level with all the peripherals disabled.

Idd Stop Current vs. Vdd  
with Temperature

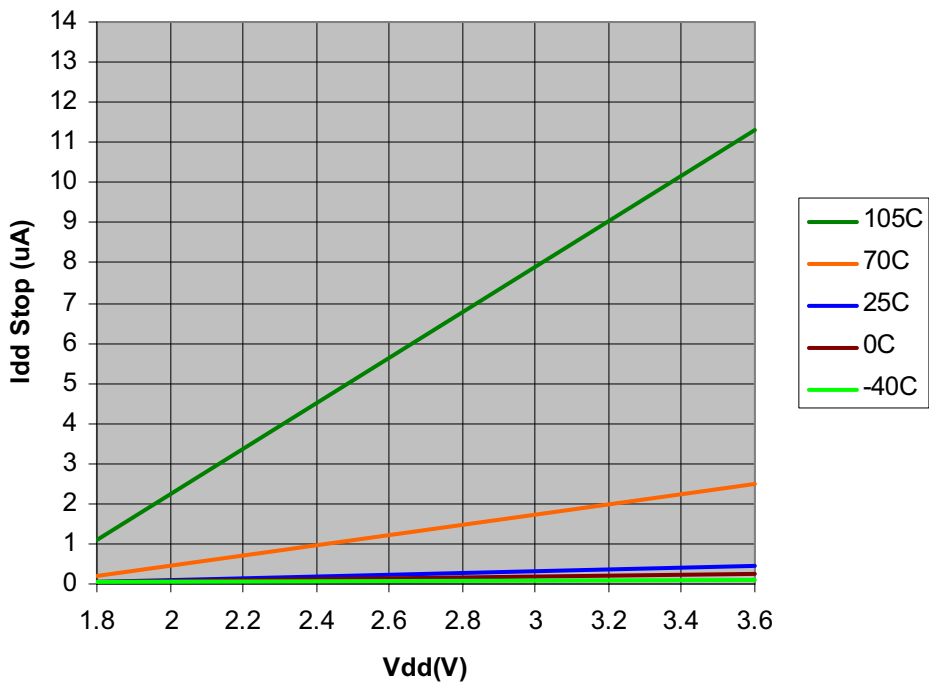


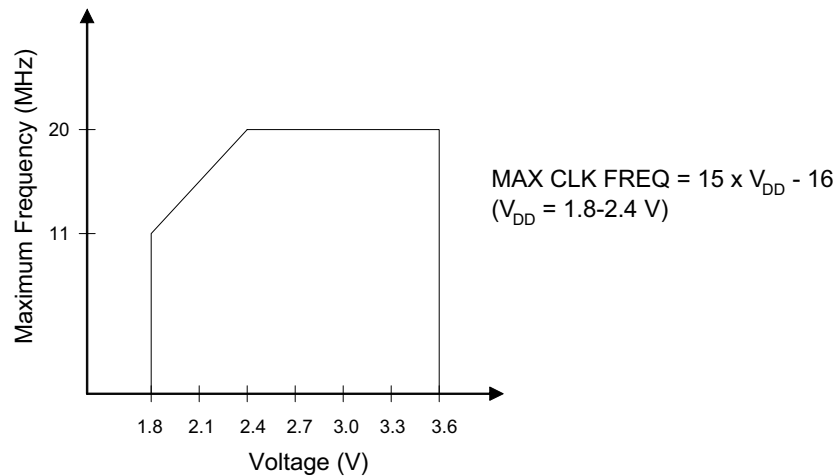
Figure 72. STOP Mode Current Consumption as a Function Vdd with Temperature as a Parameter—  
All Peripheral Disabled

## AC Characteristics

Table 193 on page 344 lists the AC characteristics and timing of the Z8 Encore! XP F1680 Series products. All AC timing information assumes a standard load of 50 pF on all outputs.

**Table 193. AC Characteristics**

Symbol	Parameter	$V_{DD} = 1.8 \text{ to } 3.6 \text{ V}$ $T_A = 0^\circ\text{C to } +70^\circ\text{C}$ $T_A = -40^\circ\text{C to } +105^\circ\text{C}$			Units	Conditions
		Min	Typ	Max		
F <sub>SYSCLK</sub>	System Clock Frequency	–		20.0	MHz	Read-only from Flash memory. See Figure 73.
		0.032768		20.0	MHz	Program or erasure of the Flash memory
T <sub>XIN</sub>	System Clock Period	50		–	ns	T <sub>CLK</sub> = 1/F <sub>sysclk</sub>
T <sub>XINH</sub>	System Clock High Time	20		30	ns	T <sub>CLK</sub> = 50 ns
T <sub>XINL</sub>	System Clock Low Time	20		30	ns	T <sub>CLK</sub> = 50 ns
T <sub>XINR</sub>	System Clock Rise Time	–		3	ns	T <sub>CLK</sub> = 50 ns
T <sub>XINF</sub>	System Clock Fall Time	–		3	ns	T <sub>CLK</sub> = 50 ns



**Figure 73.  $V_{DD}$  Versus Maximum System Clock Frequency**

## On-Chip Peripheral AC and DC Electrical Characteristics

**Table 194. Power-On Reset and Voltage Brownout Electrical Characteristics and Timing**

Symbol	Parameter	T <sub>A</sub> = 0°C to +70°C			T <sub>A</sub> = -40°C to +105°C			Units	Conditions
		Min	Typ	Max	Min	Typ <sup>1</sup>	Max		
V <sub>POR</sub>	Power-on reset Voltage Threshold				1.36	1.6	1.84	V	V <sub>DD</sub> = V <sub>POR</sub>
V <sub>TH</sub>	POR Start Voltage				—	0.6	—	V	
I <sub>DDPOR</sub>	POR Active Current				—	—	3	μA	
I <sub>DDQPOR</sub>	POR Quiescent Current				—	5	—	nA	
T <sub>ANA</sub>	Power-on reset Analog Delay				—	50	—	μs	V <sub>DD</sub> > V <sub>POR</sub>
T <sub>SMR</sub>	Stop Mode Recovery Delay				—	—	6	μs	
V <sub>VBO</sub>	Voltage Brown-Out Reset Voltage Threshold				1.6	1.8	2.0	V	V <sub>DD</sub> = V <sub>VBO</sub>
V <sub>HYS</sub>	Hysteresis of V <sub>VBO</sub>				—	80	—	mV	
T <sub>VBO</sub>	Voltage Brown-Out Pulse Rejection Period				—	10	—	μs	
I <sub>DDVBO</sub>	VBO Active Current				—	—	5	μA	
I <sub>DDQVBO</sub>	VBO Quiescent Current				—	5	—	nA	
T <sub>RAMP</sub>	V <sub>DD</sub> rampup time				0.10	—	100	ms	
<sup>1</sup> Data in the typical column is from characterization at 3.3V and 0°C. These values are provided for design guidance only and are not tested in production.									

**Table 195. Flash Memory Electrical Characteristics and Timing**

Parameter	$V_{DD} = 2.7 \text{ to } 3.6 \text{ V}$ $T_A = 0 \text{ }^{\circ}\text{C to } +70 \text{ }^{\circ}\text{C}$ $T_A = -40 \text{ }^{\circ}\text{C to } +105 \text{ }^{\circ}\text{C}$			Units	Conditions
	Min	Typ	Max		
Flash Byte Read Time	100	—	—	ns	$V_{DD} = 1.8 \text{ to } 3.6 \text{ V}$
Flash Byte Program Time	20	—	40	$\mu\text{s}$	
Flash Page Erase Time	50	—	—	ms	
Flash Mass Erase Time	50	—	—	ms	
Writes to Single Address Before Next Erase	—	—	2		
Data Retention	20	—	—	years	25 $^{\circ}\text{C}$
Endurance	5,000	—	—	cycles	Program/erase cycles

**Table 196. Watchdog Timer Electrical Characteristics and Timing**

Symbol	Parameter	$V_{DD} = 1.8 \text{ to } 3.6 \text{ V}$ $T_A = 0 \text{ }^{\circ}\text{C to } +70 \text{ }^{\circ}\text{C}$ $T_A = -40 \text{ }^{\circ}\text{C to } +105 \text{ }^{\circ}\text{C}$			Unit	Conditions
		Min	Typ	Max		
$T_{STARTUP}$		—	—	10	ms	After pd disable only
$I_{DDWDT}$	WDT Active Current	—	—	5	$\mu\text{A}$	
$I_{DDQWDT}$	WDT Quiescent Current	—	5	—	nA	
$F_{WDT}$	WDT Oscillator Frequency	2.5	5	20	kHz	

**Table 197. Non-Volatile Data Storage**

Parameter	$V_{DD} = 2.7 \text{ to } 3.6 \text{ V}$ $T_A = 0 \text{ }^{\circ}\text{C to } +70 \text{ }^{\circ}\text{C}$ $T_A = -40 \text{ }^{\circ}\text{C to } +105 \text{ }^{\circ}\text{C}$			Units	Conditions
	Min	Typ	Max		
NVDS Byte Read Time	34	—	519	$\mu\text{s}$	With system clock at 20 MHz
NVDS Byte Program Time	0.171	—	39.7	ms	With system clock at 20 MHz
Data Retention	20	—	—	years	25 $^{\circ}\text{C}$
Endurance	50,000	—	—	cycles	Cumulative write cycles for entire memory

**Table 198. Analog-to-Digital Converter Electrical Characteristics and Timing**

Symbol	Parameter	T <sub>A</sub> = 0 °C to +70 °C T <sub>A</sub> = -40 °C to +105 °C			Units	Conditions
		V <sub>DD</sub> = 1.8 to 3.6 V				
		Min	Typ	Max		
N	Resolution	–	10	–	Bit	
INL	Integral Nonlinearity	-5		5	LSB	
DNL	Differential Nonlinearity	-1		4	LSB	
	Gain Error		15		LSB	
	Offset Error	-15		15	LSB	PDIP package
		-9		9	LSB	Other packages
I <sub>DD</sub> ADC	ADC Active Current	–	–	2.5	mA	
I <sub>DDQ</sub> ADC	ADC Quiescent Current	–	5	–	nA	
V <sub>INT_REF</sub>	Internal Reference Voltage	–	1.6	–	V	REFEN=1, INTREF_SEL=0. See <a href="#">Table 101</a> .
		–	AVDD	–	V	REFEN=1, INTREF_SEL=1. See <a href="#">Table 101</a> .
V <sub>EXT_REF</sub>	External Reference Voltage	1.6	–	90% AVDD	V	REFEN=0. See <a href="#">Table 101</a> .
V <sub>INANA</sub>	Analog Input Range	0	–	1.6	V	Internal reference = 1.6 V
		0	–	90% AVDD	V	External reference or use AVDD as internal reference
C <sub>IN</sub>	Analog Input Load	–	–	5	pF	
T <sub>S</sub>	Sample Time	1.8	–	–	μs	
T <sub>H</sub>	Hold Time	0.5	–	–	μs	
T <sub>CONV</sub>	Conversion Time	–	13	–	clock cycles	
GBW <sub>IN</sub>	Input Bandwidth	–	200	–	kHz	
T <sub>WAKE</sub>	Wake-up Time	–	–	10	μs	External reference
		–	–	10	μs	Internal reference
f <sub>ADC_CLK</sub>	Maximum Frequency of adc_clk	–	–	5	MHz	V <sub>DD</sub> = 2.7 to 3.6 V
		–	–	2.5	MHz	V <sub>DD</sub> = 1.8 to 2.7 V

Table 199. Comparator Electrical Characteristics

Symbol	Parameter	T <sub>A</sub> = 0 °C to +70 °C T <sub>A</sub> = −40 °C to +105 °C			Units	Conditions
		V <sub>DD</sub> = 1.8 to 3.6 V				
		Min	Typ	Max		
V <sub>OS</sub>	Input DC Offset	−	5	−	mV	
V <sub>CREF_P</sub>	Programmable Internal Reference Voltage Range	0	−	1.8	V	
V <sub>CREF_D</sub>	Default Internal Reference Voltage	0.90	1.0	1.10	V	
I <sub>DDCMP</sub>	Comparator Active Current	−	−	400	μA	
I <sub>DDQCMP</sub>	Comparator Quiescent Current	−	5	−	nA	
V <sub>HYS</sub>	Input Hysteresis	−	8	−	mV	
T <sub>PROP</sub>	Propagation Delay	−	100	−	ns	

Table 200. Temperature Sensor Electrical Characteristics

Symbol	Parameter	T <sub>A</sub> = 0 °C to +70 °C T <sub>A</sub> = -40 °C to +105 °C						Units	Conditions
		V <sub>DD</sub> = 2.7 to 3.6 V			V <sub>DD</sub> = 1.8 to 2.7 V				
		Min	Typ	Max	Min	Typ	Max		
T <sub>AERR</sub>	Temperature Sensor Output Error	-7	–	+7	-10	–	+10	°C	-40 °C to +105 °C (as measured by ADC)
		-1.5	–	+1.5	-3	–	+3	°C	+20 °C to +30 °C (as measured by ADC)
		-10	–	10	-15	–	15	°C	-40 °C to +105 °C (as measured by comparator)
I <sub>DDTEMP</sub>	Temperature Sensor Active Current	–	–	100	–	–	100	μA	
I <sub>DDQTEMP</sub>	Temperature Sensor Quiescent Current	–	5	–	–	5	–	nA	
T <sub>WAKE</sub>	Time for Wake up	–	80	100	–	80	100	μs	

Table 201. Low Power Operational Amplifier Characteristics

Symbol	Parameter	T <sub>A</sub> = 0 °C to +70 °C T <sub>A</sub> = -40 °C to +105 °C						Units	Conditions
		V <sub>DD</sub> = 2.7 to 3.6 V			V <sub>DD</sub> = 1.8 to 2.7 V				
		Min	Typ	Max	Min	Typ	Max		
AV	DC Gain	–	80	–	–	60	–	dB	
PM	Phase Margin	–	53	–	–	45	–	deg	13 pF loading
GBW	Gain Bandwidth Product	–	0.3	–	–	0.3	–	MHz	
V <sub>OS</sub>	Input Offset Voltage	-4	–	4	-4	–	4	mV	
V <sub>OSTA</sub>	Input Offset Temperature Drift	–	1	10	–	1	10	μV/°C	
I <sub>outTA</sub>	Output Current (Drive ability of LPO)	50	–	–	40	–	–	μA	
I <sub>DD</sub> LPO	LPO Active Current	–	10	–	–	10	–	μA	
I <sub>DDQ</sub> LPO	LPO Quiescent Current	–	5	–	–	5	–	nA	
V <sub>COM</sub>	Maximum Common Input Voltage	–	–	1.4	–	–	0.7	V	

Table 202. IPO Electrical Characteristics

Symbol	Parameter	V <sub>DD</sub> = 1.8 to 3.6 V T <sub>A</sub> = -40 °C to +105 °C			V <sub>DD</sub> = 2.7 to 3.6 V T <sub>A</sub> = 0 °C to +70 °C			Units	Conditions
		Min	Typ	Max	Min	Typ	Max		
T <sub>SETUP</sub>	Setup Time for Output Frequency			15			15	μs	
I <sub>DD</sub> IPO	IPO Active Supply Current		500			500		μA	
I <sub>DDQ</sub> IPO	IPO Quiescent Current		5			5		nA	
F <sub>IPO</sub>	Output Frequency	10.6168	11.0592	11.5016	10.78272	11.0592	11.33568	MHz	±2.5% 2.7 to 3.6 V, 0–70 °C;  ±4% 1.8 to 2.7 V, 0–70 °C  ±4% 1.8 to 3.6 V, –40–105 °C
	Divided-by-2 Output Frequency	5.3084	5.5296	5.7508	5.39136	5.5296	5.66784		
	Divided-by-4 Output Frequency	2.6542	2.7648	2.8754	2.69568	2.7648	2.83392		
	Divided-by-8 Output Frequency	1.3271	1.3824	1.4377	1.34784	1.3824	1.41696		
	Divided-by-16 Output Frequency	0.6636	0.6912	0.7188	0.67392	0.6912	0.70848		
	Divided-by-32 Output Frequency	0.3318	0.3456	0.3594	0.33696	0.3456	0.35424		
	Divided-by-128 Output Frequency	0.0829	0.0864	0.0899	0.08424	0.0864	0.08856		
	Divided-by-256 Output Frequency	0.0415	0.0432	0.0449	0.04212	0.0432	0.04428		
	Duty Cycle of Output	45		55	45		55	%	

**Table 203. Low Voltage Detect Electrical Characteristics**

Symbol	Parameter	T <sub>A</sub> = 0 °C to +70 °C T <sub>A</sub> = -40 °C to +105 °C			Units	Conditions
		V <sub>DD</sub> = 1.8 to 3.6 V				
		Min	Typ	Max		
I <sub>DDLVD</sub>	LVD Active Current	—	—	50	μA	
I <sub>DDQLVD</sub>	LVD Quiescent Current	—	5	—	nA	
V <sub>TH</sub>	Detected Source Voltage	V <sub>TP</sub> - 10%	V <sub>TP</sub> <sup>1</sup>	V <sub>TP</sub> + 10%	V	
V <sub>TH_PRO</sub>	Detected Source Voltage for Flash Protection	2.4	2.5	2.6	V	
T <sub>DELAY</sub>	Delay from source voltage falling lower than V <sub>TP</sub> to I <sub>VD_OUT</sub> output logic high	50	1000	—	ns	

Note: <sup>1</sup> V<sub>TP</sub> is a user-set threshold voltage to be detected.

**Table 204. Crystal Oscillator Characteristics**

Symbol	Parameter	T <sub>A</sub> = 0 °C to +70 °C T <sub>A</sub> = -40 °C to +105 °C						Units	Conditions
		V <sub>DD</sub> = 2.7 to 3.6 V			V <sub>DD</sub> = 1.8 to 2.7 V				
		Min	Typ	Max	Min	Typ	Max		
I <sub>DDXTAL</sub>	Crystal Oscillator Active Supply Current	–	–	500	–	–	300	μA	
I <sub>DDQXTAL</sub>	Crystal Oscillator Quiescent Current	–	5	–	–	5	–	nA	
S <sub>CLK</sub>	Clk_out State in Crystal Disable	1	1	1	1	1	1		
F <sub>XTAL</sub>	External Crystal Oscillator Frequency	1	–	20	1	–	20	MHz	See <a href="#">Figure 73</a>
T <sub>SET</sub>	Startup Time After Enable	–	10,000	30,000	–	10,000	30,000	Cycle	
	Clk_out Duty Cycle	40	50	60	40	50	60	%	
	Clk_out Jitter	–	1	–	–	1	–	%	

**Table 205. Low Power 32 kHz Secondary Oscillator Characteristics**

Symbol	Parameter	T <sub>A</sub> = 0 °C to +70 °C T <sub>A</sub> = -40 °C to +105 °C						Units	Conditions
		V <sub>DD</sub> = 2.7 to 3.6 V			V <sub>DD</sub> = 1.8 to 2.7 V				
		Min	Typ	Max	Min	Typ	Max		
I <sub>DDXTAL2</sub>	32 kHz Secondary Oscillator Active Current	–	–	20	–	–	10	μA	
I <sub>DDQXTAL2</sub>	32 kHz Secondary Oscillator Quiescent Current	–	5	–	–	5	–	nA	
S <sub>CLK</sub>	Clk_out State in Crystal Disable	1	1	1	1	1	1		
F <sub>XTAL2</sub>	External Crystal Oscillator Frequency	–	32.768	–	–	32.768	–	kHz	
T <sub>SET</sub>	Startup Time After Enable	–	400	1000	–	400	1000	mS	
	Clk_out Duty Cycle	40	50	60	40	50	60	%	
	Clk_out Jitter	–	1	–	–	1	–	%	

## General Purpose I/O Port Input Data Sample Timing

Figure 74 illustrates timing of the GPIO Port input sampling. The input value on a GPIO Port pin is sampled on the rising edge of the system clock. The Port value is available to the eZ8 CPU on the second rising clock edge following the change of the Port value.

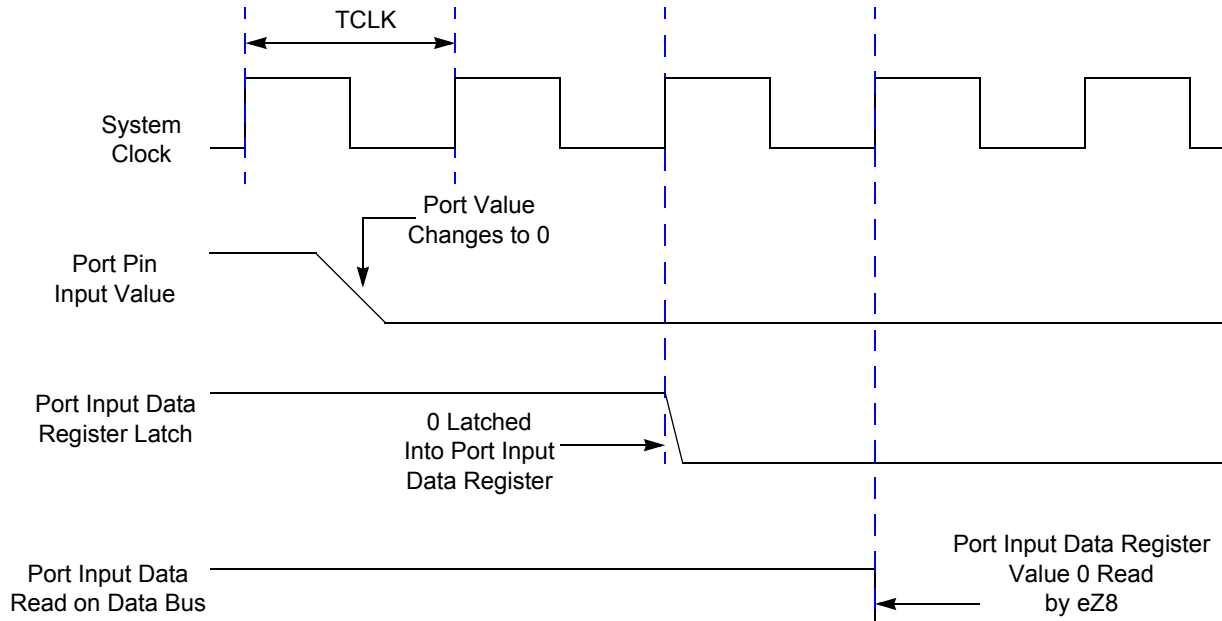


Figure 74. Port Input Sample Timing

Table 206. GPIO Port Input Timing

Parameter	Abbreviation	Delay (ns)	
		Min	Max
$T_{S\_PORT}$	Port Input Transition to XIN Rise Setup Time (Not pictured)	5	–
$T_{H\_PORT}$	XIN Rise to Port Input Transition Hold Time (Not pictured)	0	–
$T_{SMR}$	GPIO Port Pin Pulse Width to ensure Stop Mode Recovery (for GPIO Port Pins enabled as SMR sources)	1 $\mu$ s	

General Purpose I/O Port Output Timing

Figure 75 and Table 207 provides timing information for GPIO Port pins.

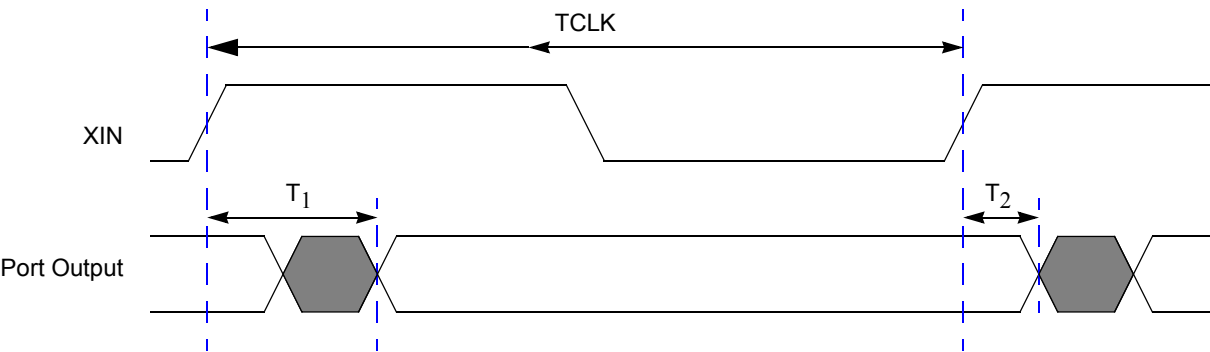


Figure 75. GPIO Port Output Timing

Table 207. GPIO Port Output Timing

		Delay (ns)	
Parameter	Abbreviation	Min	Max
GPIO Port pins			
T <sub>1</sub>	XIN Rise to Port Output Valid Delay	–	15
T <sub>2</sub>	XIN Rise to Port Output Hold Time	2	–

## On-Chip Debugger Timing

Figure 76 and Table 208 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4 ns maximum rise and fall time.

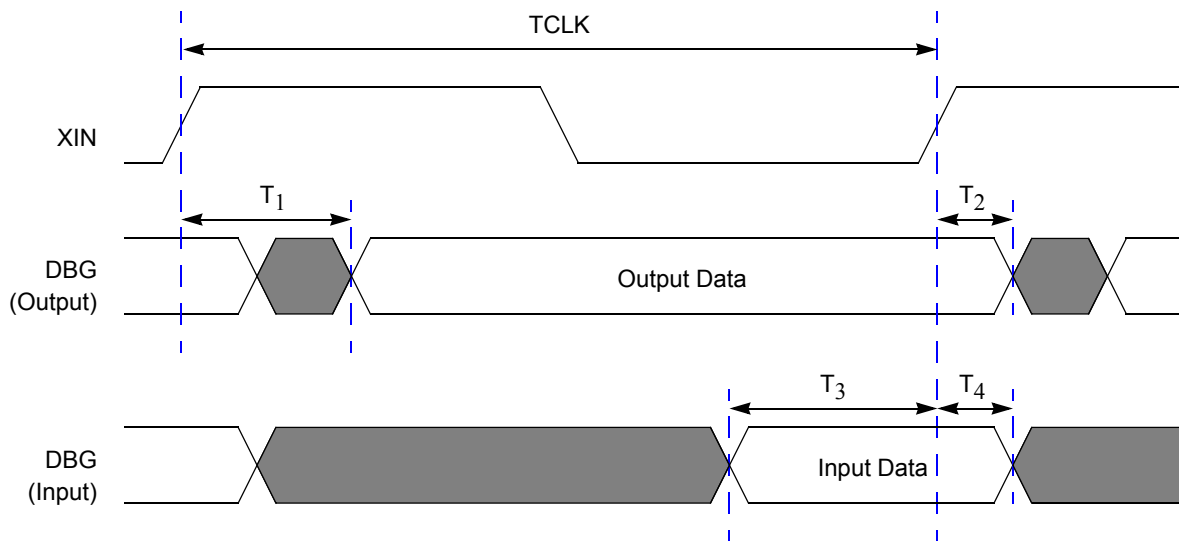


Figure 76. On-Chip Debugger Timing

Table 208. On-Chip Debugger Timing

Parameter	Abbreviation	Delay (ns)	
		Min	Max
DBG			
T <sub>1</sub>	XIN Rise to DBG Valid Delay	–	15
T <sub>2</sub>	XIN Rise to DBG Output Hold Time	2	–
T <sub>3</sub>	DBG to XIN Rise Input Setup Time	5	–
T <sub>4</sub>	DBG to XIN Rise Input Hold Time	5	–

## UART Timing

Figure 77 and Table 209 provide timing information for UART pins for the case where CTS is used for flow control. The CTS to DE assertion delay ( $T_1$ ) assumes that the transmit data register has been loaded with data prior to CTS assertion.

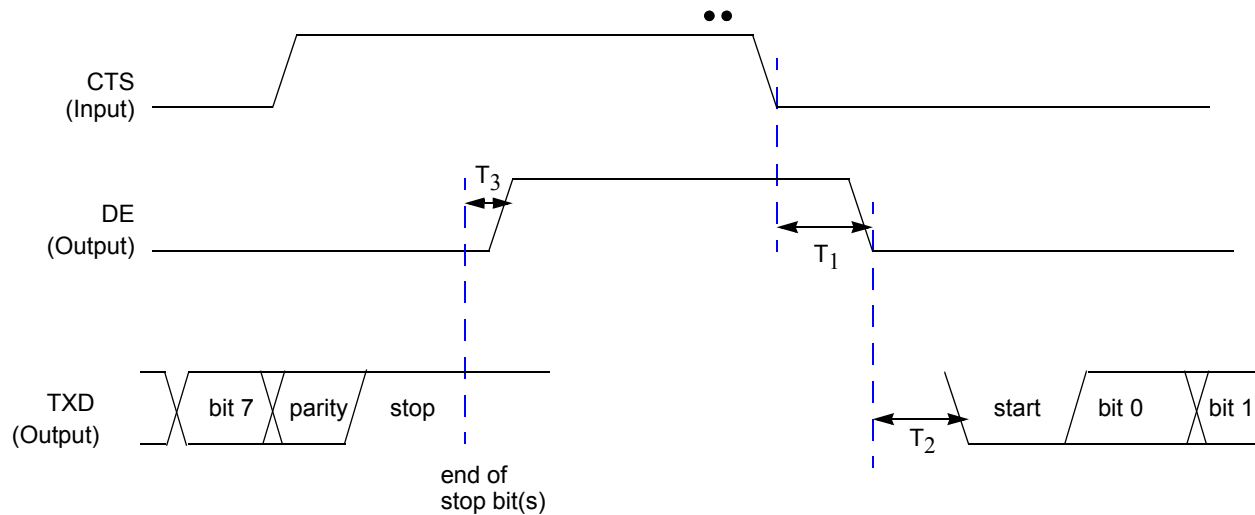


Figure 77. UART Timing With CTS

Table 209. UART Timing with CTS

Parameter	Abbreviation	Delay (ns)	
		Min	Max
UART			
T <sub>1</sub>	CTS Fall to DE output delay	2 * XIN period	2 * XIN period + 1 bit time
T <sub>2</sub>	DE assertion to TXD falling edge (start bit) delay	± 5	
T <sub>3</sub>	End of Stop Bit(s) to DE deassertion delay	± 5	

Figure 78 and Table 210 on page 357 provide timing information for UART pins for the case where CTS is not used for flow control. DE asserts after the transmit data register has been written. 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.

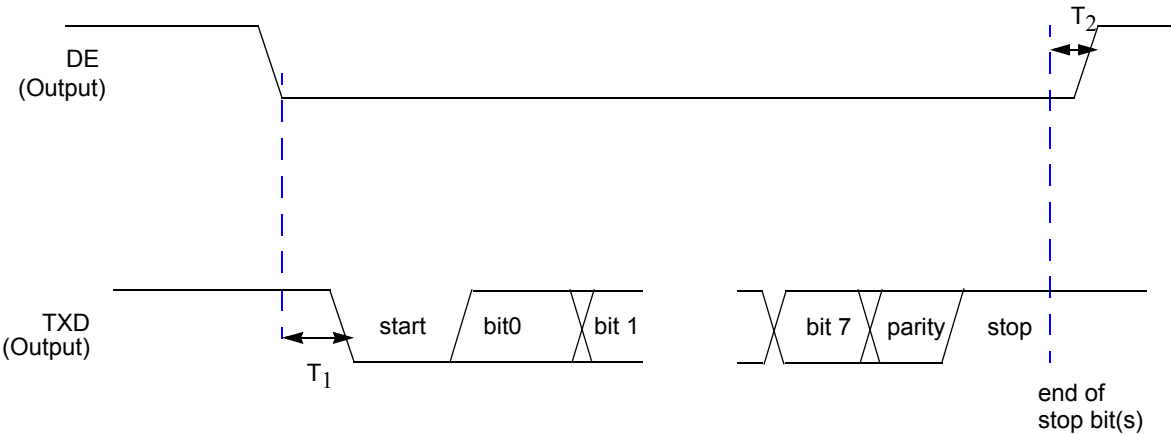


Figure 78. UART Timing Without CTS

Table 210. UART Timing Without CTS

Parameter	Abbreviation	Delay (ns)	
		Min	Max
UART			
T <sub>1</sub>	DE assertion to TXD falling edge (start bit) delay	1 * XIN period	1 bit time
T <sub>2</sub>	End of Stop Bit(s) to DE deassertion delay (Tx data register is empty)	± 5	



# Packaging

Figure 78 illustrates the 20-pin Plastic Dual Inline Package (PDIP) available for the Z8 Encore! XP F1680 Series devices.

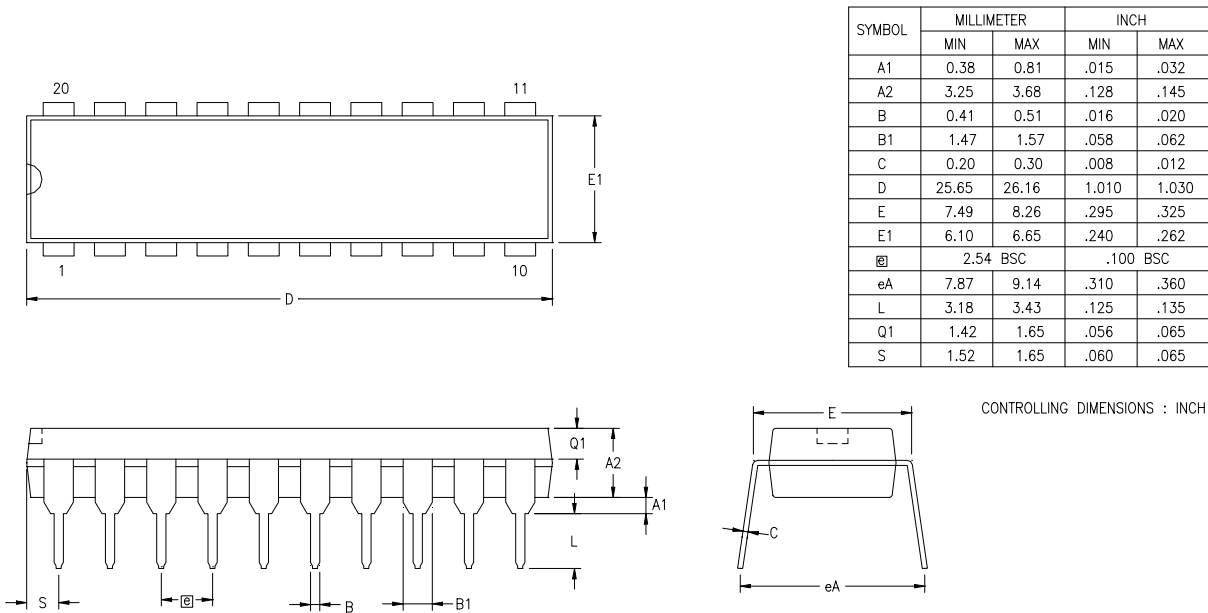
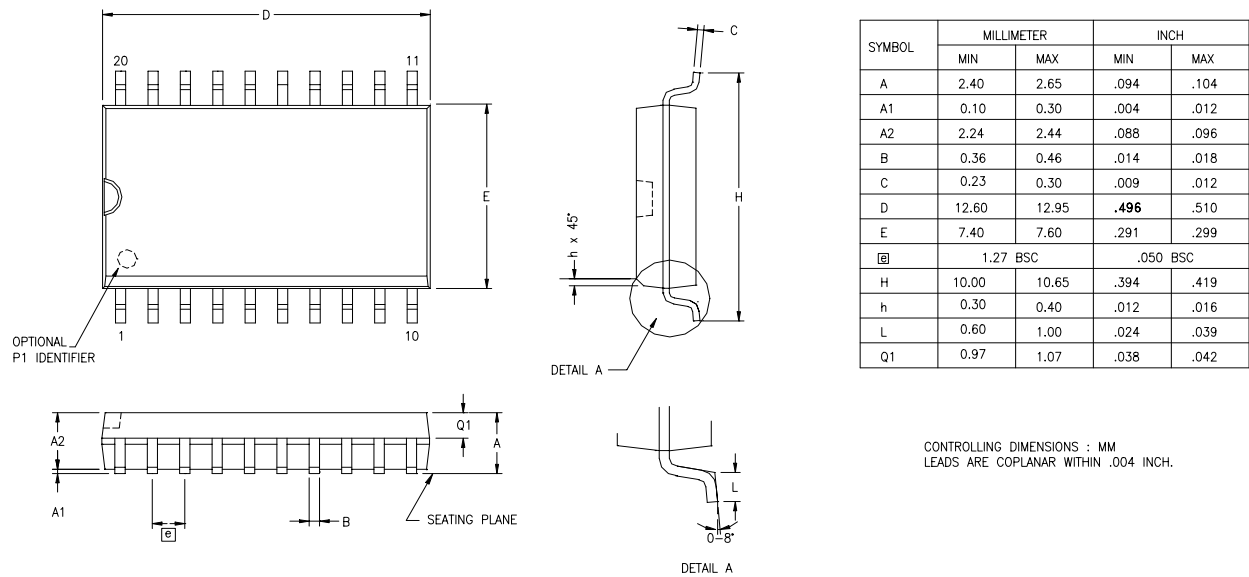


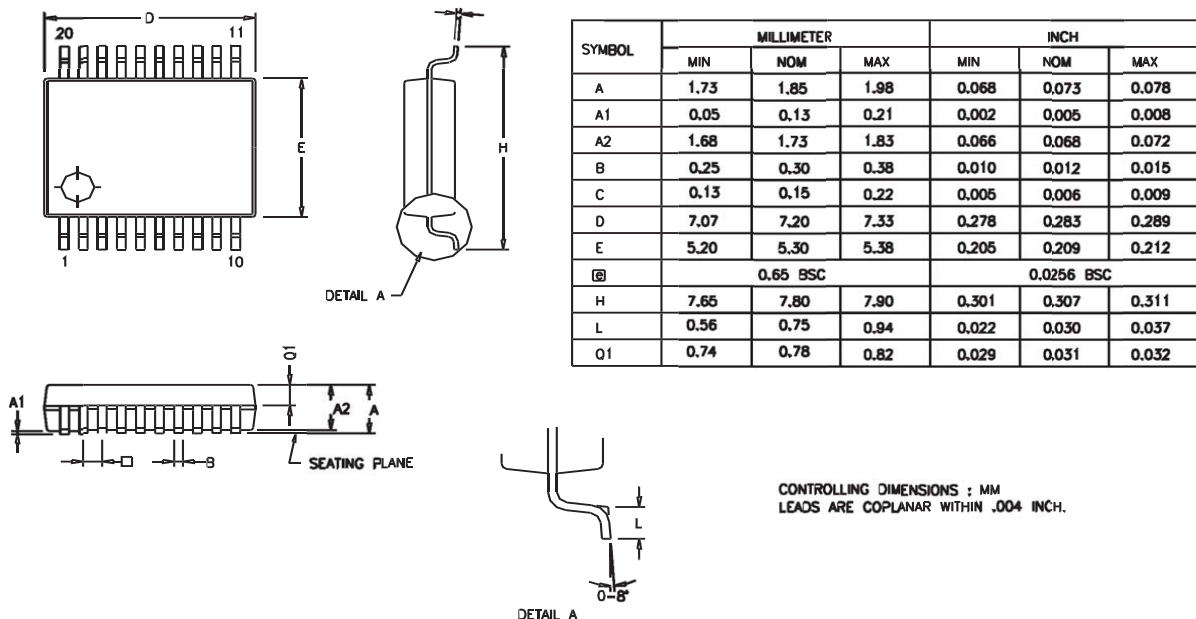
Figure 78. 20-Pin Plastic Dual Inline Package (PDIP)

Figure 79 illustrates the 20-pin Small Outline Integrated Circuit Package (SOIC) available for the Z8 Encore! XP F1680 Series devices.



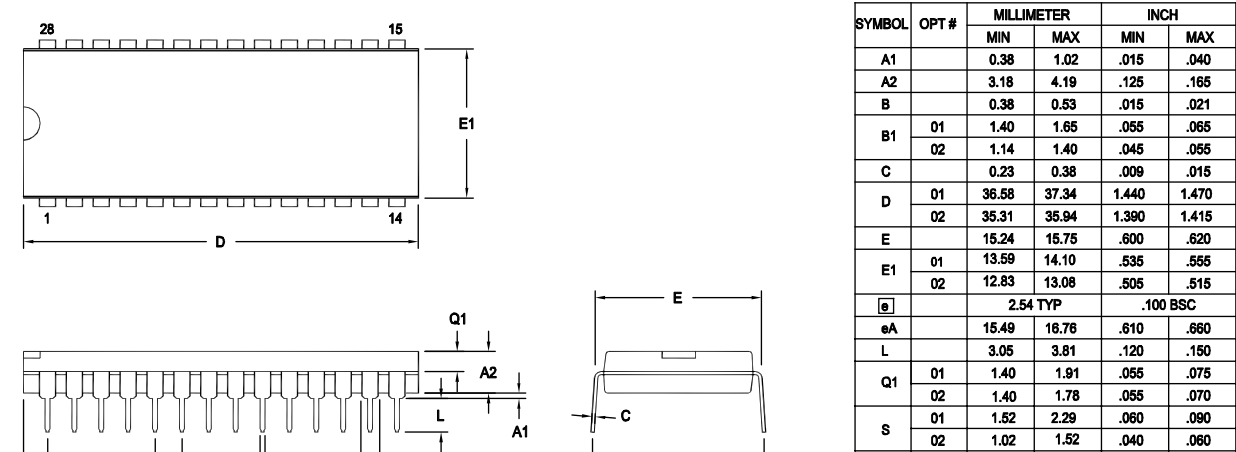
**Figure 79. 20-Pin Small Outline Integrated Circuit Package (SOIC)**

Figure 80 illustrates the 20-pin Small Shrink Outline Package (SSOP) available for the Z8 Encore! XP F1680 Series devices.



**Figure 80. 20-Pin Small Shrink Outline Package (SSOP)**

Figure 81 illustrates the 28-pin Plastic Dual Inline Package (PDIP) available for the Z8 Encore! XP F1680 Series devices.



CONTROLLING DIMENSIONS : INCH

Note: ZILOG supplies both options for production. Component layout PCB design should cover bigger option 01.

Figure 81. 28-Pin Plastic Dual In-line Package (PDIP)

Figure 82 illustrates the 28-pin Small Outline Integrated Circuit package (SOIC) available in the Z8 Encore! XP F1680 Series devices.

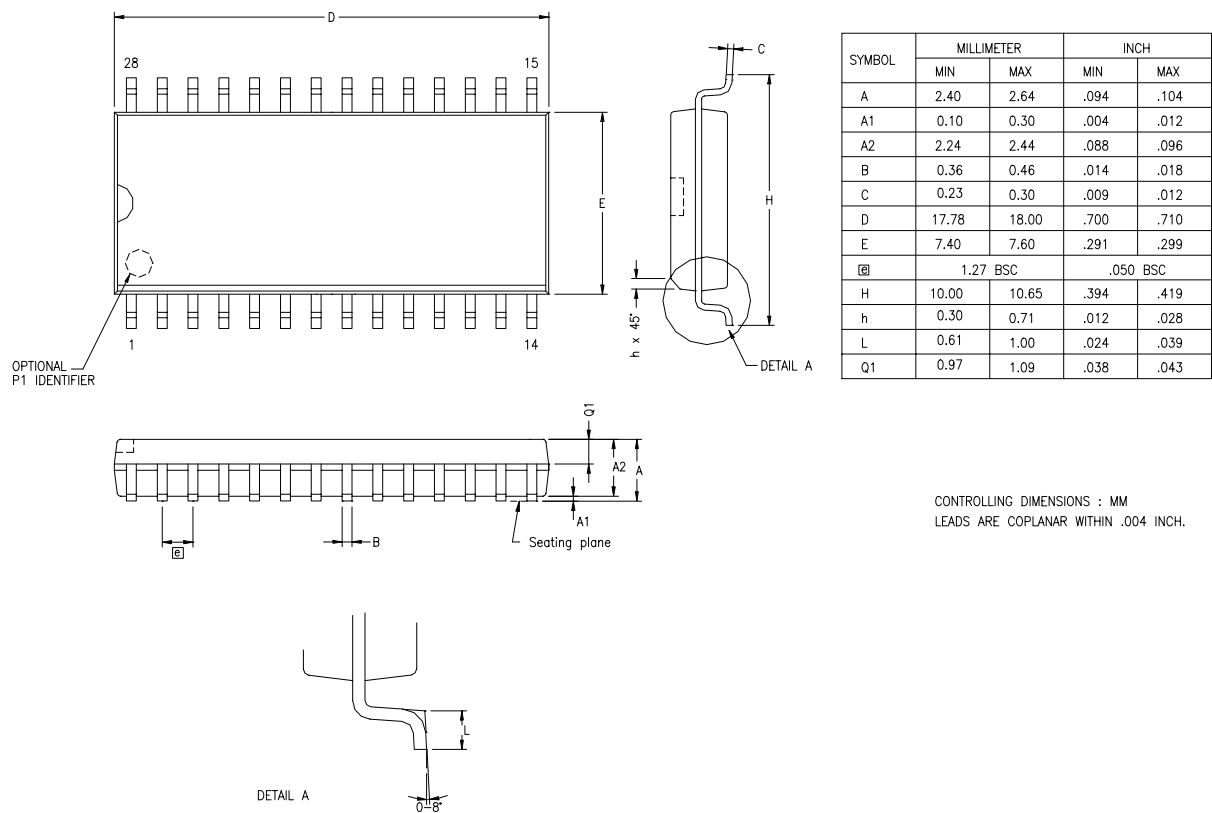


Figure 82. 28-Pin Small Outline Integrated Circuit Package (SOIC)

Figure 83 illustrates the 28-pin Small Shrink Outline Package (SSOP) available for the Z8 Encore! XP F1680 Series devices.

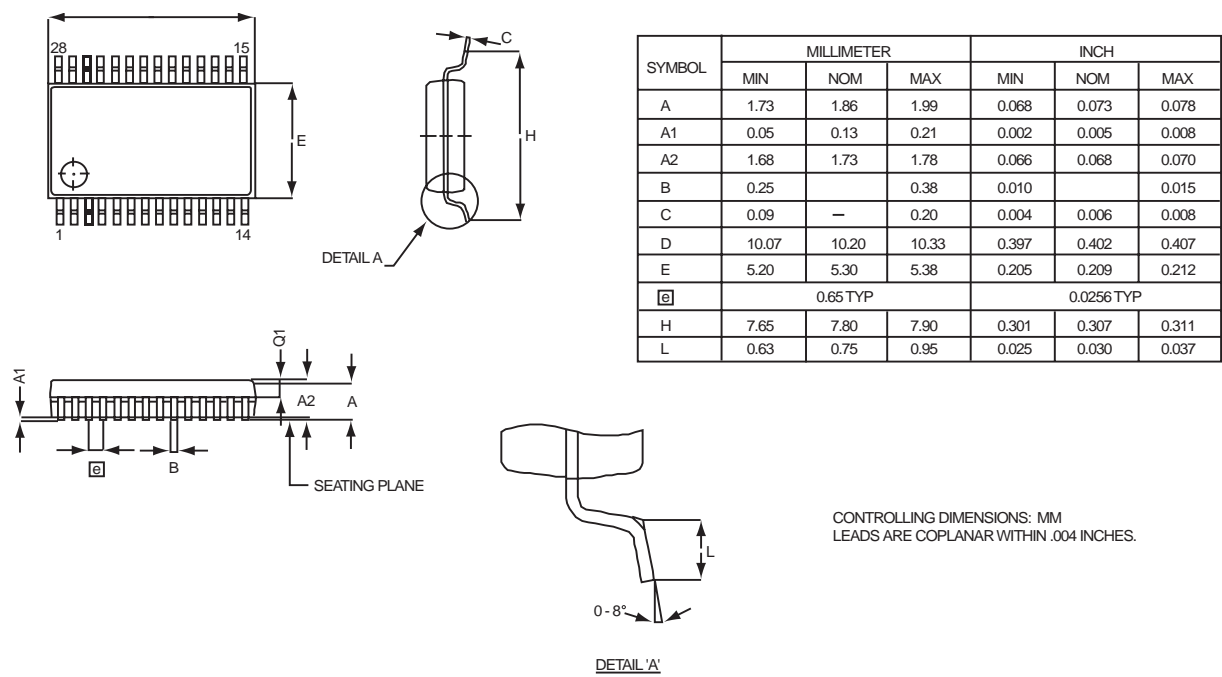
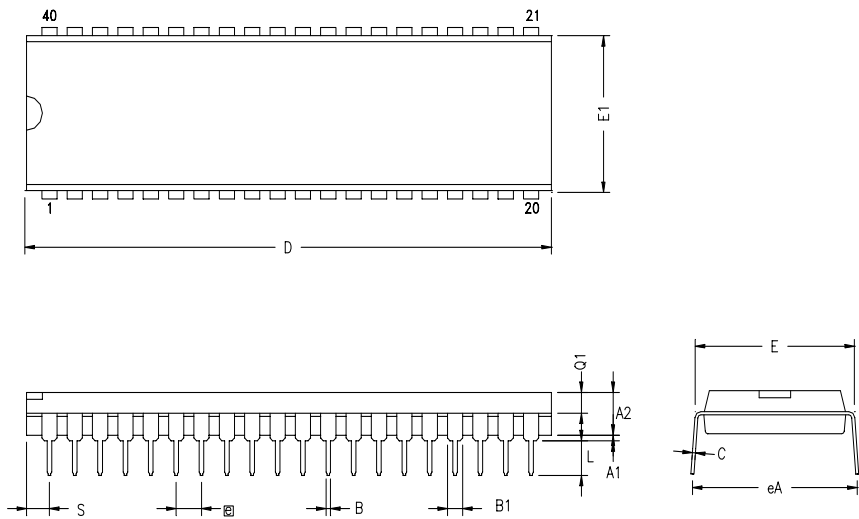


Figure 83. 28-Pin Small Shrink Outline Package (SSOP)

Figure 84 illustrates the 40-pin Plastic Dual-Inline Package (PDIP) available for the Z8 Encore! XP F1680 Series devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A1	0.51	1.02	.020	.040
A2	3.18	3.94	.125	.155
B	0.38	0.53	.015	.021
B1	1.02	1.52	.040	.060
C	0.23	0.38	.009	.015
D	52.07	52.58	2.050	2.070
E	15.24	15.75	.600	.620
E1	13.59	14.22	.535	.560
⌀	2.54 TYP		.100 TYP	
eA	15.49	16.76	.610	.660
L	3.05	3.81	.120	.150
Q1	1.40	1.91	.055	.075
S	1.52	2.29	.060	.090

CONTROLLING DIMENSIONS : INCH

Figure 84. 40-Lead Plastic Dual-Inline Package (PDIP)

Figure 85 illustrates the 44-pin Low-Profile Quad Flat Package (LQFP) available for the Z8 Encore! XP F1680 Series devices.

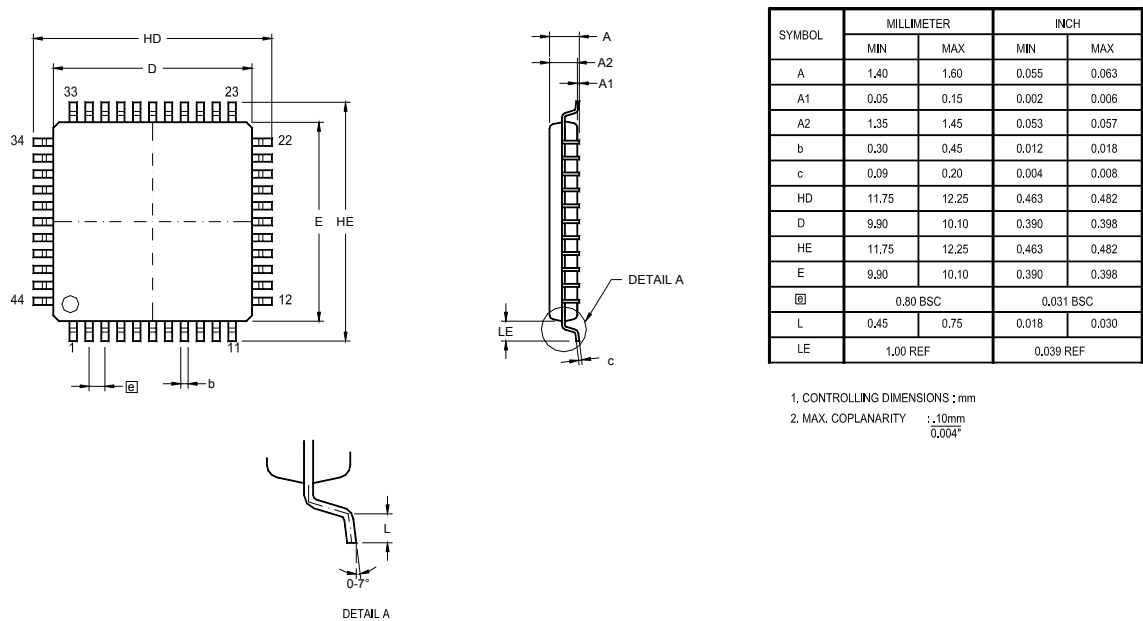


Figure 85. 44-Lead Low-Profile Quad Flat Package (LQFP)

Figure 86 illustrates the 44-pin Quad Flat No Lead (QFN) package available for the Z8 Encore! XP F1680 Series devices

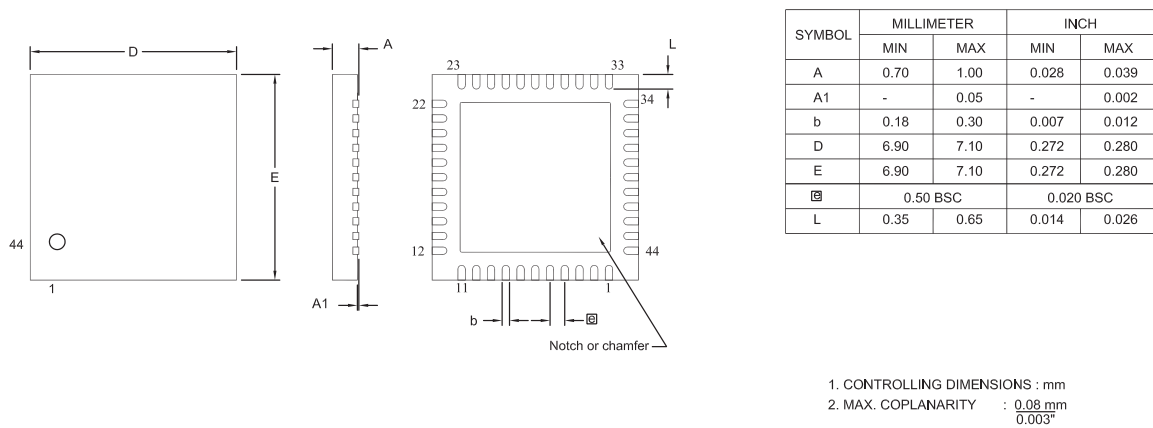


Figure 86. 44-Lead Quad Flat No Lead Package (QFN)



# Ordering Information

Order the Z8 Encore! XP<sup>®</sup> F1680 Series from Zilog<sup>®</sup>, using the following part numbers. For more information on ordering, please consult your local Zilog sales office. The Zilog website ([www.zilog.com](http://www.zilog.com)) lists all regional offices and provides additional Z8 Encore! XP product information.

Part Number	Flash	Register RAM	Program RAM	NVDS	I <sup>2</sup> C	SPI	I/O Lines	Interrupt Vectors	16-Bit Timers w/ PWM	10-Bit A/D Channels	UART with IrDA	Comparator	Temperature Sensor	Multichannel Timer	Description
<b>Z8 Encore! XP F1680 Series with 24 KB Flash, 10-Bit Analog-to-Digital Converter</b>															
<b>Standard Temperature: 0 °C to 70 °C</b>															
Z8F2480SH020SG	24 KB	2 KB	1 KB	0	1	0	17	20	3	7	1	1	1	0	SOIC 20-pin package
Z8F2480HH020SG	24 KB	2 KB	1 KB	0	1	0	17	20	3	7	1	1	1	0	SSOP 20-pin package
Z8F2480PH020SG	24 KB	2 KB	1 KB	0	1	0	17	20	3	7	1	1	1	0	PDIP 20-pin package
Z8F2480SJ020SG	24 KB	2 KB	1 KB	0	1	1	23	21	3	8	1	1	1	0	SOIC 28-pin package
Z8F2480HJ020SG	24 KB	2 KB	1 KB	0	1	1	23	21	3	8	1	1	1	0	SSOP 28-pin package
Z8F2480PJ020SG	24 KB	2 KB	1 KB	0	1	1	23	21	3	8	1	1	1	0	PDIP 28-pin package
Z8F2480PM020SG	24 KB	2 KB	1 KB	0	1	1	33	23	3	8	2	2	1	0	PDIP 40-pin package
Z8F2480AN020SG	24 KB	2 KB	1 KB	0	1	1	37	24	3	8	2	2	1	1	LQFP 44-pin package
Z8F2480QN020SG	24 KB	2 KB	1 KB	0	1	1	37	24	3	8	2	2	1	1	QFN 44-pin package
<b>Extended Temperature: -40 °C to 105 °C</b>															
Z8F2480SH020EG	24 KB	2 KB	1 KB	0	1	0	17	20	3	7	1	1	1	0	SOIC 20-pin package
Z8F2480HH020EG	24 KB	2 KB	1 KB	0	1	0	17	20	3	7	1	1	1	0	SSOP 20-pin package
Z8F2480PH020EG	24 KB	2 KB	1 KB	0	1	0	17	20	3	7	1	1	1	0	PDIP 20-pin package
Z8F2480SJ020EG	24 KB	2 KB	1 KB	0	1	1	23	21	3	8	1	1	1	0	SOIC 28-pin package
Z8F2480HJ020EG	24 KB	2 KB	1 KB	0	1	1	23	21	3	8	1	1	1	0	SSOP 28-pin package
Z8F2480PJ020EG	24 KB	2 KB	1 KB	0	1	1	23	21	3	8	1	1	1	0	PDIP 28-pin package
Z8F2480PM020EG	24 KB	2 KB	1 KB	0	1	1	33	23	3	8	2	2	1	0	PDIP 40-pin package
Z8F2480AN020EG	24 KB	2 KB	1 KB	0	1	1	37	24	3	8	2	2	1	1	LQFP 44-pin package
Z8F2480QN020EG	24 KB	2 KB	1 KB	0	1	1	37	24	3	8	2	2	1	1	QFN 44-pin package

Part Number	Flash	Register RAM	Program RAM	NVDS	I <sup>2</sup> C	SPI	I/O Lines	Interrupt Vectors	16-Bit Timers w/PWM	10-Bit A/D Channels	UART with IrDA	Comparator	Temperature Sensor	Multichannel Timer	Description
<b>Z8 Encore! XP F1680 Series with 16 KB Flash, 10-Bit Analog-to-Digital Converter</b>															

**Standard Temperature: 0 °C to 70 °C**

Z8F1680SH020SG	16 KB	2 KB	1 KB	256 B	1	0	17	20	3	7	1	1	1	0	SOIC 20-pin package
Z8F1680HH020SG	16 KB	2 KB	1 KB	256 B	1	0	17	20	3	7	1	1	1	0	SSOP 20-pin package
Z8F1680PH020SG	16 KB	2 KB	1 KB	256 B	1	0	17	20	3	7	1	1	1	0	PDIP 20-pin package
Z8F1680SJ020SG	16 KB	2 KB	1 KB	256 B	1	1	23	21	3	8	1	1	1	0	SOIC 28-pin package
Z8F1680HJ020SG	16 KB	2 KB	1 KB	256 B	1	1	23	21	3	8	1	1	1	0	SSOP 28-pin package
Z8F1680PJ020SG	16 KB	2 KB	1 KB	256 B	1	1	23	21	3	8	1	1	1	0	PDIP 28-pin package
Z8F1680PM020SG	16 KB	2 KB	1 KB	256 B	1	1	33	23	3	8	2	2	1	0	PDIP 40-pin package
Z8F1680AN020SG	16 KB	2 KB	1 KB	256 B	1	1	37	24	3	8	2	2	1	1	LQFP 44-pin package
Z8F1680QN020SG	16 KB	2 KB	1 KB	256 B	1	1	37	24	3	8	2	2	1	1	QFN 44-pin package

**Extended Temperature: –40°C to 105 °C**

Z8F1680SH020EG	16 KB	2 KB	1 KB	256 B	1	0	17	20	3	7	1	1	1	0	SOIC 20-pin package
Z8F1680HH020EG	16 KB	2 KB	1 KB	256 B	1	0	17	20	3	7	1	1	1	0	SSOP 20-pin package
Z8F1680PH020EG	16 KB	2 KB	1 KB	256 B	1	0	17	20	3	7	1	1	1	0	PDIP 20-pin package
Z8F1680SJ020EG	16 KB	2 KB	1 KB	256 B	1	1	23	21	3	8	1	1	1	0	SOIC 28-pin package
Z8F1680HJ020EG	16 KB	2 KB	1 KB	256 B	1	1	23	21	3	8	1	1	1	0	SSOP 28-pin package
Z8F1680PJ020EG	16 KB	2 KB	1 KB	256 B	1	1	23	21	3	8	1	1	1	0	PDIP 28-pin package
Z8F1680PM020EG	16 KB	2 KB	1 KB	256 B	1	1	33	23	3	8	2	2	1	0	PDIP 40-pin package
Z8F1680AN020EG	16 KB	2 KB	1 KB	256 B	1	1	37	24	3	8	2	2	1	1	LQFP 44-pin package
Z8F1680QN020EG	16 KB	2 KB	1 KB	256 B	1	1	37	24	3	8	2	2	1	1	QFN 44-pin package

Part Number	Flash	Register RAM	Program RAM	NVDS	I <sup>2</sup> C	SPI	I/O Lines	Interrupt Vectors	16-Bit Timers w/PWM	10-Bit A/D Channels	UART with IrDA	Comparator	Temperature Sensor	Multichannel Timer	Description
<b>Z8 Encore! XP F1680 Series with 8 KB Flash, 10-Bit Analog-to-Digital Converter</b>															
<b>Standard Temperature: 0 °C to 70 °C</b>															
Z8F0880SH020SG	8 KB	1 KB	1 KB	128 B	1	0	17	20	3	7	1	1	1	0	SOIC 20-pin package
Z8F0880HH020SG	8 KB	1 KB	1 KB	128 B	1	0	17	20	3	7	1	1	1	0	SSOP 20-pin package
Z8F0880PH020SG	8 KB	1 KB	1 KB	128 B	1	0	17	20	3	7	1	1	1	0	PDIP 20-pin package
Z8F0880SJ020SG	8 KB	1 KB	1 KB	128 B	1	1	23	21	3	8	1	1	1	0	SOIC 28-pin package
Z8F0880HJ020SG	8 KB	1 KB	1 KB	128 B	1	1	23	21	3	8	1	1	1	0	SSOP 28-pin package
Z8F0880PJ020SG	8 KB	1 KB	1 KB	128 B	1	1	23	21	3	8	1	1	1	0	PDIP 28-pin package
Z8F0880PM020SG	8 KB	1 KB	1 KB	128 B	1	1	33	23	3	8	2	2	1	0	PDIP 40-pin package
Z8F0880AN020SG	8 KB	1K B	1 KB	128 B	1	1	37	24	3	8	2	2	1	1	LQFP 44-pin package
Z8F0880QN020SG	8 KB	1 KB	1 KB	128 B	1	1	37	24	3	8	2	2	1	1	QFN 44-pin package
<b>Extended Temperature: –40 °C to 105 °C</b>															
Z8F0880SH020EG	8 KB	1 KB	1 KB	128 B	1	0	17	20	3	7	1	1	1	0	SOIC 20-pin package
Z8F0880HH020EG	8 KB	1 KB	1 KB	128 B	1	0	17	20	3	7	1	1	1	0	SSOP 20-pin package
Z8F0880PH020EG	8 KB	1 KB	1 KB	128 B	1	0	17	20	3	7	1	1	1	0	PDIP 20-pin package
Z8F0880SJ020EG	8 KB	1 KB	1 KB	128 B	1	1	23	21	3	8	1	1	1	0	SOIC 28-pin package
Z8F0880HJ020EG	8 KB	1 KB	1 KB	128 B	1	1	23	21	3	8	1	1	1	0	SSOP 28-pin package
Z8F0880PJ020EG	8 KB	1 KB	1 KB	128 B	1	1	23	21	3	8	1	1	1	0	PDIP 28-pin package
Z8F0880PM020EG	8 KB	1 KB	1 KB	128 B	1	1	33	23	3	8	2	2	1	0	PDIP 40-pin package
Z8F0880AN020EG	8 KB	1 KB	1 KB	128 B	1	1	37	24	3	8	2	2	1	1	LQFP 44-pin package
Z8F0880QN020EG	8 KB	1 KB	1 KB	128 B	1	1	37	24	3	8	2	2	1	1	QFN 44-pin package
Z8F16800128ZCOG														Z8 Encore! XP F1680 28-pin Series Development Kit	
Z8F16800144ZCOG														Z8 Encore! XP Dual 44-pin F1680 Series Development Kit	

## Part Number Suffix Designations

Z8 F 16 80 S H 020 S G

### Environmental Flow

G = Green Plastic Packaging Compound

### Temperature Range

S = Standard, 0 °C to 70 °C

E = Extended, -40 °C to +105 °C

### Speed

020 = 20 MHz

### Pin Count \*

H = 20

J = 28

M = 40

N = 44

### Package \*

S = SOIC

H = SSOP

P = PDIP

A = LQFP

Q = QFN

### Device Type

### Memory Size

24 = 24 KB Flash, 2 KB RAM, 0 B NVDS

16 = 16 KB Flash, 2 KB RAM, 256 B NVDS

08 = 8 KB Flash, 1 KB RAM, 128 B NVDS

### Memory Type

F = Flash

### Device Family

Z8 = Zilog's 8-Bit Microcontroller

\* See [Table 211](#) for the combination of package and pin count.

**Table 211. Package and Pin Count Description**

		Pin Count			
		20	28	40	44
Package	SOIC	√	√		
	SSOP	√	√		
	PDIP	√	√	√	
	LQFP				√
	QFN				√

## **Precharacterization Product**

The product represented by this document is newly introduced and Zilog® has not completed the full characterization of the product. The document states what Zilog knows about this product at this time, but additional features or nonconformance with some aspects of the document might be found, either by Zilog or its customers in the course of further application and characterization work. In addition, Zilog cautions that delivery might be uncertain at times, because of start-up yield issues.



# Index

## Symbols

# 320  
% 320  
@ 320

## Numerics

10-bit ADC 4  
40-lead plastic dual-inline package 362, 363,  
364  
44-lead low-profile quad flat package 365

## A

absolute maximum ratings 337  
AC characteristics 344  
ADC 321  
    block diagram 182  
    electrical characteristics and timing 347  
    overview 181  
ADC Channel Register 1 (ADCCTL) 184  
ADC Data High Byte Register (ADCDH) 185,  
186  
ADC Data Low Bit Register (ADC DL) 186, 187,  
188  
ADCX 321  
ADD 321  
add - extended addressing 321  
add with carry 321  
add with carry - extended addressing 321  
additional symbols 320  
address space 21  
ADDX 321  
analog block/PWM signal synchronization 183  
analog signals 16  
analog-to-digital converter  
    overview 181  
AND 323  
ANDX 323

architecture

    voltage measurements 181  
arithmetic instructions 321  
assembly language syntax 318

## B

B 320  
b 319  
baud rate generator, UART 156  
BCLR 322  
binary number suffix 320  
BIT 322  
bit 319  
    clear 322  
    manipulation instructions 322  
    set 322  
    set or clear 322  
    swap 322  
    test and jump 324  
    test and jump if non-zero 324  
    test and jump if zero 324  
bit jump and test if non-zero 324  
bit swap 324  
block diagram 3  
block transfer instructions 322  
BRK 324  
BSET 322  
BSWAP 322, 324  
BTJ 324  
BTJNZ 324  
BTJZ 324

## C

calibration and compensation, motor control  
measurements 184  
CALL procedure 324  
capture mode 112, 113  
capture/compare mode 112

- cc 319
- CCF 322
- characteristics, electrical 337
- clear 323
- clock phase (SPI) 195
- CLR 323
- COM 323
- compare - extended addressing 321
- compare with carry 321
- compare with carry - extended addressing 321
- complement 323
- complement carry flag 322
- condition code 319
- control register definition, UART 159
- control register, I2C 237
- Control Registers 21
- CP 321
- CPC 321
- CPCX 321
- CPU and peripheral overview 4
- CPU control instructions 322
- CPX 321
- current measurement
  - architecture 181
  - operation 181
- Customer Feedback Form 383

## **D**

- DA 319, 321
- data memory 23
- data register, I2C 235
- DC characteristics 338
- debugger, on-chip 285
- DEC 321
- decimal adjust 321
- decrement 321
- decrement and jump non-zero 324
- decrement word 321
- DECW 321
- destination operand 320
- device, port availability 49
- DI 322
- direct address 319

- disable interrupts 322
- DJNZ 324
- dst 320

## **E**

- EI 322
- electrical characteristics 337
  - ADC 347
  - flash memory and timing 346
  - GPIO input data sample timing 352
  - watch-dog timer 346, 348
- electrical noise 181
- enable interrupt 322
- ER 319
- extended addressing register 319
- external pin reset 38
- eZ8 CPU features 4
- eZ8 CPU instruction classes 320
- eZ8 CPU instruction notation 318
- eZ8 CPU instruction set 317
- eZ8 CPU instruction summary 325

## **F**

- FCTL register 262, 269
- features, Z8 Encore! 1
- first opcode map 335
- FLAGS 320
- flags register 320
- flash
  - controller 4
    - option bit address space 270
    - option bit configuration - reset 267
    - program memory address 0000H 270
    - program memory address 0001H 271
- flash memory 253
  - arrangement 254, 255, 256
  - byte programming 259
  - code protection 258
  - configurations 253
  - control register definitions 261, 269
  - controller bypass 260
  - electrical characteristics and timing 346

- flash control register 262, 269
- flash option bits 258
- flash status register 262
- flow chart 257
- frequency high and low byte registers 264
- mass erase 260
- operation 256
- operation timing 258
- page erase 260
- page select register 263
- FPS register 263
- FSTAT register 262

## G

- gated mode 112
- general-purpose I/O 49
- GPIO 4, 49
  - alternate functions 50
  - architecture 50
  - control register definitions 61
  - input data sample timing 352
  - interrupts 61
  - port A-C pull-up enable sub-registers 66, 67
  - port A-H address registers 62
  - port A-H alternate function sub-registers 64
  - port A-H control registers 63
  - port A-H data direction sub-registers 63
  - port A-H high drive enable sub-registers 65
  - port A-H input data registers 68
  - port A-H output control sub-registers 65
  - port A-H output data registers 69
  - port A-H stop mode recovery sub-registers 66
  - port availability by device 49
  - port input timing 353
  - port output timing 354

## H

- H 320
- HALT 322
- halt mode 46, 322

- hexadecimal number prefix/suffix 320

## I

- I2C 4
  - 10-bit address read transaction 226
  - 10-bit address transaction 223
  - 10-bit addressed slave data transfer format 224, 231
  - 7-bit address transaction 221, 228
  - 7-bit address, reading a transaction 225
  - 7-bit addressed slave data transfer format 222, 230
  - 7-bit receive data transfer format 226, 232, 234
  - baud high and low byte registers 239, 240, 244
  - C status register 241
  - controller 215
  - controller signals 15
  - interrupts 218
  - operation 217
  - SDA and SCL signals 217
  - stop and start conditions 220
- I2CBRH register 240, 245
- I2CCTL register 238
- I2CSTAT register 241
- IM 319
- immediate data 319
- immediate operand prefix 320
- INC 321
- increment 321
- increment word 321
- INCW 321
- indexed 319
- indirect address prefix 320
- indirect register 319
- indirect register pair 319
- indirect working register 319
- indirect working register pair 319
- infrared encoder/decoder (IrDA) 177
- Instruction Set 317
- instruction set, ez8 CPU 317

instructions

ADC 321  
ADCX 321  
ADD 321  
ADDX 321  
AND 323  
ANDX 323  
arithmetic 321  
BCLR 322  
BIT 322  
bit manipulation 322  
block transfer 322  
BRK 324  
BSET 322  
BSWAP 322, 324  
BTJ 324  
BTJNZ 324  
BTJZ 324  
CALL 324  
CCF 322  
CLR 323  
COM 323  
CP 321  
CPC 321  
CPCX 321  
CPU control 322  
CPX 321  
DA 321  
DEC 321  
DECW 321  
DI 322  
DJNZ 324  
EI 322  
HALT 322  
INC 321  
INCW 321  
IRET 324  
JP 324  
LD 323  
LDC 323  
LDCI 322, 323  
LDE 323  
LDEI 322  
LDX 323

LEA 323  
load 323  
logical 323  
MULT 321  
NOP 322  
OR 323  
ORX 323  
POP 323  
POPX 323  
program control 324  
PUSH 323  
PUSHX 323  
RCF 322  
RET 324  
RL 324  
RLC 324  
rotate and shift 324  
RR 324  
RRC 324  
SBC 321  
SCF 322  
SRA 324  
SRL 324  
SRP 322  
STOP 322  
SUB 321  
SUBX 321  
SWAP 324  
TCM 322  
TCMX 322  
TM 322  
TMX 322  
TRAP 324  
watch-dog timer refresh 322  
XOR 323  
XORX 323  
instructions, eZ8 classes of 320  
interrupt control register 83  
interrupt controller 71  
    architecture 71  
    interrupt assertion types 74  
    interrupt vectors and priority 74  
    operation 73  
    register definitions 75

- software interrupt assertion 75
- interrupt edge select register 82
- interrupt request 0 register 75
- interrupt request 1 register 76
- interrupt request 2 register 77
- interrupt return 324
- interrupt vector listing 71
- interrupts
  - SPI 204
  - UART 153
- IR 319
- Ir 319
- IrDA
  - architecture 105, 157, 177
  - block diagram 105, 157, 177
  - control register definitions 180
  - operation 105, 157, 177
  - receiving data 179
  - transmitting data 178
- IRET 324
- IRQ0 enable high and low bit registers 78
- IRQ1 enable high and low bit registers 79
- IRQ2 enable high and low bit registers 80
- IRR 319
- Irr 319

**J**

- JP 324
- jump, conditional, relative, and relative conditional 324

**L**

- LD 323
- LDC 323
- LDCI 322, 323
- LDE 323
- LDEI 322, 323
- LDX 323
- LEA 323
- load 323
- load constant 322
- load constant to/from program memory 323

- load constant with auto-increment addresses 323
- load effective address 323
- load external data 323
- load external data to/from data memory and auto-increment addresses 322
- load external to/from data memory and auto-increment addresses 323
- load instructions 323
- load using extended addressing 323
- logical AND 323
- logical AND/extended addressing 323
- logical exclusive OR 323
- logical exclusive OR/extended addressing 323
- logical instructions 323
- logical OR 323
- logical OR/extended addressing 323
- low power modes 45
- LQFP
  - 44 lead 365

## M

- master interrupt enable 73
- master-in, slave-out and-in 193
- memory
  - data 23
  - program 22
- MISO 193
- mode
  - capture 112, 113
  - capture/compare 112
  - gated 112
  - PWM 112, 113
- MOSI 193
- motor control measurements
  - ADC Control register definitions 184
  - calibration and compensation 184
  - interrupts 182, 184
  - overview 181
- MULT 321
- multiply 321
- multiprocessor mode, UART 148

## **N**

noise, electrical 181  
 NOP (no operation) 322  
 notation  
   b 319  
   cc 319  
   DA 319  
   ER 319  
   IM 319  
   IR 319  
   lr 319  
   IRR 319  
   lrr 319  
   p 319  
   R 319  
   r 319  
   RA 319  
   RR 319  
   rr 319  
   vector 319  
   X 319  
 notational shorthand 319

## **O**

OCD  
   architecture 285  
   auto-baud detector/generator 288  
   baud rate limits 288  
   block diagram 285  
   breakpoints 291  
   commands 292  
   control register 299  
   data format 287  
   DBG pin to RS-232 Interface 286  
   debug mode 287  
   debugger break 324  
   interface 286  
   serial errors 289  
   status register 300  
   timing 355  
 OCD commands  
   execute instruction (12H) 297  
   read data memory (0DH) 296

  read OCD control register (05H) 295  
   read OCD revision (00H) 294  
   read OCD status register (02H) 294  
   read program counter (07H) 295  
   read program memory (0BH) 296  
   read program memory CRC (0EH) 297  
   read register (09H) 295  
   read runtime counter (03H) 294  
   step instruction (10H) 297  
   stuff instruction (11H) 297  
   write data memory (0CH) 296  
   write OCD control register (04H) 295  
   write program counter (06H) 295  
   write program memory (0AH) 296  
   write register (08H) 295  
 on-chip debugger (OCD) 285  
 on-chip debugger signals 17  
 on-chip oscillator 309  
 opcode map  
   abbreviations 334  
   cell description 334  
   first 335  
   second after 1FH 336  
 operation 183  
   current measurement 181  
   voltage measurement timing diagram 183  
 Operational Description 33, 45, 49, 71, 85,  
 119, 137, 141, 177, 181, 189, 247, 251, 253,  
 267, 281, 285, 303, 309, 315  
 OR 323  
 ordering information 367  
 ORX 323  
 oscillator signals 16

## **P**

p 319  
 packaging  
   20-pin PDIP 359, 360  
   20-pin SSOP 360, 363  
   28-pin PDIP 361  
   28-pin SOIC 362  
 LQFP  
   44 lead 365

PDIP 362, 363, 364  
 part selection guide 2  
 PC 320  
 PDIP 362, 363, 364  
 peripheral AC and DC electrical characteristics 345  
 PHASE=0 timing (SPI) 196  
 PHASE=1 timing (SPI) 197  
 pin characteristics 17  
 Pin Descriptions 11  
 polarity 319  
 POP 323  
 pop using extended addressing 323  
 POPX 323  
 port availability, device 49  
 port input timing (GPIO) 353  
 port output timing, GPIO 354  
 power supply signals 17  
 power-on reset (POR) 35  
 program control instructions 324  
 program counter 320  
 program memory 22  
 PUSH 323  
 push using extended addressing 323  
 PUSHX 323  
 PWM mode 112, 113  
 PxADDR register 62  
 PxCTL register 63

## **R**

R 319  
 r 319  
 RA  
     register address 319  
 RCF 322  
 receive  
     7-bit data transfer format (I2C) 226, 232, 234  
     IrDA data 179  
 receiving UART data-interrupt-driven method 146  
 receiving UART data-pollled method 145  
 register 209, 319

baud low and high byte (I2C) 239, 240, 244  
 baud rate high and low byte (SPI) 213  
 control, I2C 237  
 data, SPI 205, 206  
 flash control (FCTL) 262, 269  
 flash high and low byte (FFREQH and FREEQL) 264  
 flash page select (FPS) 263  
 flash status (FSTAT) 262  
 GPIO port A-H address (PxADDR) 62  
 GPIO port A-H alternate function sub-registers 64  
 GPIO port A-H control address (PxCTL) 63  
 GPIO port A-H data direction sub-registers 64  
 I2C baud rate high (I2CBRH) 240, 245  
 I2C control (I2CCTL) 238  
 I2C status 241  
 I2C status (I2CSTAT) 241  
 mode, SPI 209  
 OCD control 299  
 OCD status 300  
 SPI baud rate high byte (SPIBRH) 213  
 SPI baud rate low byte (SPIBRL) 213  
 SPI control (SPICCTL) 207  
 SPI data (SPIDATA) 206  
 SPI status (SPISTAT) 210  
 status, SPI 210  
 UARTx baud rate high byte (UxBRH) 172  
 UARTx baud rate low byte (UxBRL) 172  
 UARTx Control 0 (UxCTL0) 165, 171  
 UARTx control 1 (UxCTL1) 116, 166, 169, 170  
 UARTx receive data (UxRXD) 159  
 UARTx status 0 (UxSTAT0) 160, 161  
 UARTx status 1 (UxSTAT1) 163  
 UARTx transmit data (UxTXD) 159  
 watch-dog timer control (WDTCTL) 248, 249, 306, 308  
 watchdog timer control (WDTCTL) 42  
 watch-dog timer reload high byte (WDTH) 140  
 register pair 319  
 register pointer 320

- registers
  - ADC channel 1 184
  - ADC data high byte 185, 186
  - ADC data low bit 186, 187, 188
- reset
  - and stop mode characteristics 34
  - carry flag 322
  - sources 35
- RET 324
- return 324
- RL 324
- RLC 324
- rotate and shift instructions 324
- rotate left 324
- rotate left through carry 324
- rotate right 324
- rotate right through carry 324
- RP 320
- RR 319, 324
- rr 319
- RRC 324

## S

- SBC 321
- SCF 322
- SCK 193
- SDA and SCL (I<sup>2</sup>C) signals 217
- second opcode map after 1FH 336
- serial clock 193
- serial peripheral interface (SPI) 191
- set carry flag 322
- set register pointer 322
- shift right arithmetic 324
- shift right logical 324
- signal descriptions 14
- slave data transfer formats (I<sup>2</sup>C) 224, 231
- slave select 194
- software trap 324
- source operand 320
- SP 320
- SPI
  - architecture 191
  - baud rate generator 205

- baud rate high and low byte register 213
- clock phase 195
- configured as slave 202
- control register definitions 205
- data register 205, 206
- error detection 203
- interrupts 204
- mode fault error 203
- mode register 209
- multi-master operation 200
- operation 193
- overrun error 203
- signals 193
- single master, multiple slave system 201
- single master, single slave system 201
- status register 210
- timing, PHASE = 0 196
- timing, PHASE=1 197
- SPI controller signals 15
- SPI mode (SPIMODE) 209
- SPIBRH register 213
- SPIBRL register 213
- SPICTL register 207
- SPIDATA register 206
- SPIMODE register 209
- SPISTAT register 210
- SRA 324
- src 320
- SRL 324
- SRP 322
- SS, SPI signal 193
- stack pointer 320
- STOP 322
- stop mode 45, 322
- stop mode recovery
  - sources 39, 41
  - using a GPIO port pin transition 41
  - using watch-dog timer time-out 40
- SUB 321
- subtract 321
- subtract - extended addressing 321
- subtract with carry 321
- subtract with carry - extended addressing 321
- SUBX 321

SWAP 324  
 swap nibbles 324  
 symbols, additional 320

## **T**

TCM 322  
 TCMX 322  
 Technical Support 383  
 test complement under mask 322  
 test complement under mask - extended addressing 322  
 test under mask 322  
 test under mask - extended addressing 322  
 timer signals 16  
 timers 85  
   architecture 85, 119  
   block diagram 86, 119  
   capture mode 96, 112, 113  
   capture/compare mode 100, 112  
   compare mode 99  
   continuous mode 90  
   counter mode 91, 92  
   gated mode 99, 112  
   operating mode 88  
   PWM mode 93, 95, 112, 113  
   reading the timer count values 104  
   reload high and low byte registers 108, 128, 129  
   timer control register definitions 107  
   timer output signal operation 104  
   triggered one-shot mode 89  
 timers 0-3  
   control registers 110, 111, 114, 115  
   high and low byte registers 107, 108, 109, 127  
 timing diagram, voltage measurement 183  
 TM 322  
 TMX 322  
 transmit  
   IrDA data 178  
 transmitting UART data-interrupt-driven method 144  
 transmitting UART data-polled method 143

TRAP 324

## **U**

UART 4  
   architecture 141  
   baud rate generator 156  
   baud rates table 174, 175  
   control register definitions 159  
   controller signals 15  
   data format 142  
   interrupts 153  
   multiprocessor mode 148  
   receiving data using interrupt-driven method 146  
   receiving data using the polled method 145  
   transmitting data using the interrupt-driven method 144  
   transmitting data using the polled method 143  
   x baud rate high and low registers 172  
   x control 0 and control 1 registers 165, 166  
   x status 0 and status 1 registers 160, 163  
 UxBRH register 172  
 UxBRL register 172  
 UxCTL0 register 165, 171  
 UxCTL1 register 116, 166, 169, 170  
 UxRXD register 159  
 UxSTAT0 register 160, 161  
 UxSTAT1 register 163  
 UxTXD register 159

## **V**

vector 319  
 voltage brownout reset (VBR) 37  
 voltage measurement timing diagram 183

## **W**

watch-dog timer  
   approximate time-out delay 137  
   approximate time-out delays 137, 247, 251, 281, 303, 315

- CNTL 37
  - control register 248, 249, 306, 307
  - electrical characteristics and timing 348
  - operation 137, 247, 251, 281, 303, 315
  - refresh 138
  - reload unlock sequence 139
  - reload upper, high and low registers 139
  - reset 38
  - reset in normal operation 138
  - reset in STOP mode 139
  - time-out response 138
- watchdog timer
  - electrical characteristics and timing 346
  - refresh 322
- WDTCTL register 42, 248, 249, 306, 308
- WDTH register 140
- working register 319
- working register pair 319

## **X**

- X 319
- XOR 323
- XORX 323

## **Z**

- Z8 Encore!
  - block diagram 3
  - features 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>.

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

ZiLOG:

[Z8F2480SJ020SG](#) [Z8F1680SJ020SG](#) [Z8F2480AN020SG](#) [Z8F1680AN020SG](#) [Z8F1680QN020SG](#)