zilog

Embedded in Life

An IXYS Company

*High-Performance 8-Bit Microcontrollers*

# Z8 Encore! XP® F1680 Series

## Product Specification

PS025015-1212

PRELIMINARY

Copyright ©2012 Zilog®, Inc. All rights reserved.

www.zilog.com

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

## LIFE SUPPORT POLICY

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

### As used herein

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

### Document Disclaimer

©2012 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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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 and Z8 Encore! MC are trademarks or registered trademarks of Zilog, Inc. All other product or service names are the property of their respective owners.

# Revision History

Each instance in the Revision History table below reflects a change to this document from its previous version. For more details, click the appropriate links in the table.

| Date | Revision Level | Description | Page |
|---|---|---|---|
| Dec 2012 | 15 | Added Timer Clock Source footnote to the TxCTS2 Register. | 117 |
| Nov 2012 | 14 | In the Multi-Channel Timer chapter, corrected/ clarified instances of the TInA–TInD and TOutA–TOutD to T4CHA–T4CHD GPIO pins to more accurately address their relationship to $T_{IN}$. | 120 |
| Oct 2011 | 13 | Revised Flash Sector Protect Register descriptions per CR 13212; revised Packaging chapter. | 274, 371 |
| May 2011 | 12 | Correction to Trim Bit Address 0001H Register per CR 13091. | 283 |
| Oct 2010 | 11 | Comparator 1 Control Register (CMP1) address formerly showed F90H; now corrected to F91H. | 258 |
| Sep 2010 | 10 | Removed references to LSBF bit in Master-In/Slave-Out, Master-Out/Slave-In and SPI Master Operation sections. | 199, 207 |
| Aug 2010 | 09 | Changed the frequency for the Internal Precision RC Oscillator from 1.3842 to 1.3824 in Table 168 per CR 12961. | 316 |
| Jun 2008 | 08 | Updated Trim Option Bits at 0005H (TVREF). | 287 |
| Mar 2008 | 07 | Updated Operation of the On-Chip Debugger Interface and Ordering Information sections. Added Target OCD Connector Interface. | 296 |

| Date | Revision Level | Description | Page |
|---|---|---|---|
| Oct 2007 | 06 | Updated Trim Bit Address Description, Trim Bit Address 0007H, Trim Bit Address 0008H, DC Characteristics, Supply Current Characteristics, VDD Versus Maximum System Clock Frequency, Watchdog Timer Electrical Characteristics and Timing, Analog-to-Digital Converter Electrical Characteristics and Timing, Comparator Electrical Characteristics, Low Power Operational Amplifier Characteristics, IPO Electrical Characteristics and Low Voltage Detect Electrical Characteristics. Added Figure 73. | 282, 288, 288, 350, 352, 356, 357, 359, 360, 361 |
| Sep 2007 | 05 | Updated Supply Current Characteristics. | 352 |
| Aug 2007 | 04 | Changed description of Z8F16800144ZCOG to Z8 Encore! XP Dual 44-pin F1680 Series Development Kit. Updated electrical characteristics in Table 189, Table 190, Table 192, Table 193, Table 195, Table 196. Removed VBO_Trim section and table. | 350, 352, 358, 359, 360, 374 |

# Table of Contents

Revision History . . . . . iii
List of Figures . . . . . xv
List of Tables . . . . . xviii
Chapter 1. Overview . . . . . 1
1.1. Features . . . . . 1
1.2. Part Selection Guide . . . . . 2
1.3. Block Diagram . . . . . 3
1.4. An Overview of the eZ8 CPU and its Peripherals . . . . . 4
1.4.1. General-Purpose Input/Output . . . . . 4
1.4.2. Flash Controller . . . . . 4
1.4.3. Non-Volatile Data Storage . . . . . 5
1.4.4. Internal Precision Oscillator . . . . . 5
1.4.5. Crystal Oscillator . . . . . 5
1.4.6. Secondary Oscillator . . . . . 5
1.4.7. 10-Bit Analog-to-Digital Converter . . . . . 5
1.4.8. Low-Power Operational Amplifier . . . . . 5
1.4.9. Analog Comparator . . . . . 5
1.4.10. Temperature Sensor . . . . . 6
1.4.11. Low-Voltage Detector . . . . . 6
1.4.12. Enhanced SPI . . . . . 6
1.4.13. UART with LIN . . . . . 6
1.4.14. Master/Slave I2C . . . . . 6
1.4.15. Timers . . . . . 6
1.4.16. Multi-Channel Timer . . . . . 7
1.4.17. Interrupt Controller . . . . . 7
1.4.18. Reset Controller . . . . . 7
1.4.19. On-Chip Debugger . . . . . 7
1.4.20. Direct LED Drive . . . . . 7
1.5. Acronyms and Expansions . . . . . 8
Chapter 2. Pin Description . . . . . 10
2.1. Available Packages . . . . . 10
2.2. Pin Configurations . . . . . 10
2.3. Signal Descriptions . . . . . 14
2.4. Pin Characteristics . . . . . 17

Chapter 3. Address Space . . . . . 19
3.1. Register File . . . . . 19
3.2. Program Memory . . . . . 20
3.3. Data Memory . . . . . 21
3.4. Flash Information Area . . . . . 21
Chapter 4. Register Map . . . . . 23
Chapter 5. Reset, Stop Mode Recovery and Low-Voltage Detection . . . . . 31
5.1. Reset Types . . . . . 31
5.2. Reset Sources . . . . . 33
5.2.1. Power-On Reset . . . . . 34
5.2.2. Voltage Brown-Out Reset . . . . . 35
5.2.3. Watchdog Timer Reset . . . . . 36
5.2.4. External Reset Input . . . . . 37
5.2.5. External Reset Indicator . . . . . 37
5.2.6. On-Chip Debugger Initiated Reset . . . . . 37
5.3. Stop Mode Recovery . . . . . 37
5.3.1. Stop Mode Recovery Using Watchdog Timer Time-Out . . . . . 38
5.3.2. Stop Mode Recovery Using Timer Interrupt . . . . . 38
5.3.3. Stop Mode Recovery Using Comparator Interrupt . . . . . 39
5.3.4. Stop Mode Recovery Using GPIO Port Pin Transition . . . . . 39
5.3.5. Stop Mode Recovery Using External RESET Pin . . . . . 39
5.4. Low-Voltage Detection . . . . . 39
5.5. Reset Register Definitions . . . . . 40
Chapter 6. Low-Power Modes . . . . . 42
6.1. STOP Mode . . . . . 42
6.2. HALT Mode . . . . . 43
6.3. Peripheral-Level Power Control . . . . . 43
6.4. Power Control Register Definitions . . . . . 44
Chapter 7. General-Purpose Input/Output . . . . . 46
7.1. GPIO Port Availability by Device . . . . . 46
7.2. Architecture . . . . . 47
7.3. GPIO Alternate Functions . . . . . 47
7.4. Direct LED Drive . . . . . 48
7.5. Shared Reset Pin . . . . . 48
7.6. Crystal Oscillator Override . . . . . 48
7.7. 32kHz Secondary Oscillator Override . . . . . 48

7.8. 5V Tolerance . . . . . 48
7.9. External Clock Setup . . . . . 49
7.10. GPIO Interrupts . . . . . 58
7.11. GPIO Control Register Definitions . . . . . 58
7.11.1. Port A–E Address Registers . . . . . 59
7.11.2. Port A–E Control Registers . . . . . 60
7.11.3. Port A–E Data Direction Subregisters . . . . . 60
7.11.4. Port A–E Alternate Function Subregisters . . . . . 61
7.11.5. Port A–E Output Control Subregisters . . . . . 62
7.11.6. Port A–E High Drive Enable Subregisters . . . . . 62
7.11.7. Port A–E Stop Mode Recovery Source Enable Subregisters . . . . . 63
7.11.8. Port A–E Pull-up Enable Subregisters . . . . . 63
7.11.9. Port A–E Alternate Function Set 1 Subregisters . . . . . 64
7.11.10.Port A–E Alternate Function Set 2 Subregisters . . . . . 64
7.11.11.Port A–E Input Data Registers . . . . . 65
7.11.12.Port A–E Output Data Register . . . . . 66
7.11.13.LED Drive Enable Register . . . . . 66
7.11.14.LED Drive Level Registers . . . . . 67

Chapter 8. Interrupt Controller . . . . . 68
8.1. Interrupt Vector Listing . . . . . 68
8.2. Architecture . . . . . 70
8.3. Operation . . . . . 70
8.3.1. Master Interrupt Enable . . . . . 70
8.3.2. Interrupt Vectors and Priority . . . . . 71
8.3.3. Interrupt Assertion . . . . . 71
8.3.4. Software Interrupt Assertion . . . . . 72
8.4. Interrupt Control Register Definitions . . . . . 72
8.4.1. Interrupt Request 0 Register . . . . . 73
8.4.2. Interrupt Request 1 Register . . . . . 74
8.4.3. Interrupt Request 2 Register . . . . . 75
8.4.4. IRQ0 Enable High and Low Bit Registers . . . . . 76
8.4.5. IRQ1 Enable High and Low Bit Registers . . . . . 77
8.4.6. IRQ2 Enable High and Low Bit Registers . . . . . 79
8.4.7. Interrupt Edge Select Register . . . . . 82
8.4.8. Shared Interrupt Select Register . . . . . 82
8.4.9. Interrupt Control Register . . . . . 83

Chapter 9. Timers . . . . . 84
9.1. Architecture . . . . . 85
9.2. Operation . . . . . 85

9.2.1. Timer Clock Source . . . . . 85
9.2.2. Low-Power Modes . . . . . 86
9.2.3. Timer Operating Modes . . . . . 87
9.2.4. Reading the Timer Count Values . . . . . 106
9.2.5. Timer Output Signal Operation . . . . . 106
9.2.6. Timer Noise Filter . . . . . 106
9.2.7. Architecture . . . . . 106
9.3. Timer Control Register Definitions . . . . . 108
9.3.1. Timer 0–2 High and Low Byte Registers . . . . . 109
9.3.2. Timer Reload High and Low Byte Registers . . . . . 109
9.3.3. Timer 0–2 PWM0 High and Low Byte Registers . . . . . 110
9.3.4. Timer 0–2 PWM1 High and Low Byte Registers . . . . . 111
9.3.5. Timer 0–2 Control Registers . . . . . 112
9.3.6. Timer 0–2 Status Registers . . . . . 118
9.3.7. Timer 0–2 Noise Filter Control Register . . . . . 119

Chapter 10. Multi-Channel Timer . . . . . 120
10.1. Architecture . . . . . 120
10.2. Timer Operation . . . . . 121
10.2.1. Multi-Channel Timer Counter . . . . . 121
10.2.2. Clock Source . . . . . 121
10.2.3. Multi-Channel Timer Clock Prescaler . . . . . 122
10.2.4. Multi-Channel Timer Start . . . . . 122
10.2.5. Multi-Channel Timer Mode Control . . . . . 122
10.2.6. Count Modulo Mode . . . . . 122
10.2.7. Count Up/Down Mode . . . . . 123
10.3. Capture/Compare Channel Operation . . . . . 124
10.3.1. One-Shot Compare Operation . . . . . 124
10.3.2. Continuous Compare Operation . . . . . 124
10.3.3. PWM Output Operation . . . . . 125
10.3.4. Capture Operation . . . . . 125
10.4. Multi-Channel Timer Interrupts . . . . . 125
10.4.1. Timer Interrupt . . . . . 125
10.4.2. Capture/Compare Channel Interrupt . . . . . 125
10.5. Low-Power Modes . . . . . 126
10.5.1. Operation in HALT Mode . . . . . 126
10.5.2. Operation in STOP Mode . . . . . 126
10.5.3. Power Reduction During Operation . . . . . 126
10.6. Multi-Channel Timer Applications Examples . . . . . 126
10.6.1. PWM Programmable Deadband Generation . . . . . 126
10.6.2. Multiple Timer Intervals Generation . . . . . 127

10.7. Multi-Channel Timer Control Register Definitions . . . . . 128
10.7.1. Multi-Channel Timer Address Map . . . . . 128
10.7.2. Multi-Channel Timer High and Low Byte Registers . . . . . 130
10.7.3. Multi-Channel Timer Reload High and Low Byte Registers . . . . . 130
10.7.4. Multi-Channel Timer Subaddress Register . . . . . 131
10.7.5. Multi-Channel Timer Subregister x (0, 1, or 2) . . . . . 132
10.7.6. Multi-Channel Timer Control 0, Control 1 Registers . . . . . 132
10.7.7. Multi-Channel Timer Channel Status 0 and Status 1 Registers . . . 135
10.7.8. Multi-Channel Timer Channel-y Control Registers . . . . . 137
10.7.9. Multi-Channel Timer Channel-y High and Low Byte Registers . . 139

Chapter 11. Watchdog Timer . . . . . 140
11.1. Operation . . . . . 140
11.1.1. Watchdog Timer Refresh . . . . . 141
11.1.2. Watchdog Timer Time-Out Response . . . . . 141
11.1.3. Watchdog Timer Reload Unlock Sequence . . . . . 142
11.2. Watchdog Timer Register Definitions . . . . . 142
11.2.1. Watchdog Timer Reload High and Low Byte Registers . . . . . 143

Chapter 12. LIN-UART . . . . . 144
12.1. LIN-UART Architecture . . . . . 144
12.1.1. Data Format for Standard UART Modes . . . . . 145
12.1.2. Transmitting Data using the Polled Method . . . . . 146
12.1.3. Transmitting Data Using Interrupt-Driven Method . . . . . 147
12.1.4. Receiving Data Using Polled Method . . . . . 148
12.1.5. Receiving Data Using the Interrupt-Driven Method . . . . . 149
12.1.6. Clear To Send Operation . . . . . 150
12.1.7. External Driver Enable . . . . . 150
12.1.8. LIN-UART Special Modes . . . . . 151
12.1.9. MULTIPROCESSOR Mode . . . . . 151
12.1.10.LIN Protocol Mode . . . . . 153
12.1.11.LIN-UART Interrupts . . . . . 157
12.1.12.LIN-UART Baud Rate Generator . . . . . 160
12.2. Noise Filter . . . . . 160
12.2.1. Architecture . . . . . 161
12.2.2. Operation . . . . . 161
12.3. LIN-UART Control Register Definitions . . . . . 163
12.3.1. LIN-UART Transmit Data Register . . . . . 163
12.3.2. LIN-UART Receive Data Register . . . . . 164
12.3.3. LIN-UART Status 0 Register . . . . . 165
12.3.4. LIN-UART Mode Select and Status Register . . . . . 168

12.3.5. LIN-UART Control 0 Register . . . . . 170
12.3.6. LIN-UART Control 1 Registers . . . . . 171
12.3.7. Noise Filter Control Register . . . . . 174
12.3.8. LIN Control Register . . . . . 175
12.3.9. LIN-UART Address Compare Register . . . . . 177
12.3.10. LIN-UART Baud Rate High and Low Byte Registers . . . . . 177

Chapter 13. Infrared Encoder/Decoder . . . . . 182
13.1. Architecture . . . . . 182
13.2. Operation . . . . . 182
13.2.1. Transmitting IrDA Data . . . . . 183
13.2.2. Receiving IrDA Data . . . . . 184
13.3. Infrared Encoder/Decoder Control Register Definitions . . . . . 185

Chapter 14. Analog-to-Digital Converter . . . . . 186
14.1. Architecture . . . . . 186
14.2. Operation . . . . . 186
14.2.1. ADC Timing . . . . . 187
14.2.2. ADC Interrupt . . . . . 188
14.2.3. Reference Buffer . . . . . 188
14.2.4. Internal Voltage Reference Generator . . . . . 189
14.2.5. Calibration and Compensation . . . . . 189
14.3. ADC Control Register Definitions . . . . . 189
14.3.1. ADC Control Register 0 . . . . . 189
14.3.2. ADC Raw Data High Byte Register . . . . . 191
14.3.3. ADC Data High Byte Register . . . . . 191
14.3.4. ADC Data Low Bits Register . . . . . 192
14.3.5. Sample Settling Time Register . . . . . 193
14.3.6. Sample Time Register . . . . . 194
14.3.7. ADC Clock Prescale Register . . . . . 195

Chapter 15. Low-Power Operational Amplifier . . . . . 196

Chapter 16. Enhanced Serial Peripheral Interface . . . . . 197
16.1. Architecture . . . . . 197
16.2. ESPI Signals . . . . . 199
16.2.1. Master-In/Slave-Out . . . . . 199
16.2.2. Master-Out/Slave-In . . . . . 199
16.2.3. Serial Clock . . . . . 199
16.2.4. Slave Select . . . . . 200
16.3. Operation . . . . . 200
16.3.1. Throughput . . . . . 201

16.3.2. ESPI Clock Phase and Polarity Control . . . . . 201
16.3.3. Slave Select Modes of Operation . . . . . 203
16.3.4. SPI Protocol Configuration . . . . . 207
16.3.5. Error Detection . . . . . 210
16.3.6. ESPI Interrupts . . . . . 211
16.3.7. ESPI Baud Rate Generator . . . . . 212
16.4. ESPI Control Register Definitions . . . . . 213
16.4.1. ESPI Data Register . . . . . 213
16.4.2. ESPI Transmit Data Command and Receive Data Buffer Control Register . . . . . 214
16.4.3. ESPI Control Register . . . . . 215
16.4.4. ESPI Mode Register . . . . . 217
16.4.5. ESPI Status Register . . . . . 219
16.4.6. ESPI State Register . . . . . 220
16.4.7. ESPI Baud Rate High and Low Byte Registers . . . . . 221

Chapter 17. I2C Master/Slave Controller . . . . . 223
17.1. Architecture . . . . . 223
17.1.1. I2C Master/Slave Controller Registers . . . . . 224
17.2. Operation . . . . . 225
17.2.1. SDA and SCL Signals . . . . . 225
17.2.2. $I^2C$ Interrupts . . . . . 226
17.2.3. Start and Stop Conditions . . . . . 228
17.2.4. Software Control of I2C Transactions . . . . . 228
17.2.5. Master Transactions . . . . . 228
17.2.6. Slave Transactions . . . . . 236
17.3. $I^2C$ Control Register Definitions . . . . . 243
17.3.1. I2C Data Register . . . . . 243
17.3.2. I2C Interrupt Status Register . . . . . 245
17.3.3. I2C Control Register . . . . . 247
17.3.4. I2C Baud Rate High and Low Byte Registers . . . . . 248
17.3.5. I2C State Register . . . . . 250
17.3.6. I2C Mode Register . . . . . 253
17.3.7. I2C Slave Address Register . . . . . 255

Chapter 18. Comparator . . . . . 256
18.1. Operation . . . . . 256
18.2. Comparator Control Register Definitions . . . . . 257
18.2.1. Comparator 0 Control Register . . . . . 257
18.2.2. Comparator 1 Control Register . . . . . 258

Chapter 19. Temperature Sensor . . . . . 260

19.1. Operation . . . . . 260
19.1.1. Calibration . . . . . 261

Chapter 20. Flash Memory . . . . . 262
20.1. Flash Information Area . . . . . 262
20.2. Operation . . . . . 265
20.2.1. Flash Operation Timing Using Flash Frequency Registers . . . . . 267
20.2.2. Flash Code Protection Against External Access . . . . . 267
20.2.3. Flash Code Protection Against Accidental Program and Erasure . 267
20.2.4. Byte Programming . . . . . 269
20.2.5. Page Erase . . . . . 270
20.2.6. Mass Erase . . . . . 270
20.2.7. Flash Controller Bypass . . . . . 270
20.2.8. Flash Controller Behavior in Debug Mode . . . . . 271
20.3. Flash Control Register Definitions . . . . . 271
20.3.1. Flash Control Register . . . . . 271
20.3.2. Flash Status Register . . . . . 272
20.3.3. Flash Page Select Register . . . . . 273
20.3.4. Flash Sector Protect Register . . . . . 274
20.3.5. Flash Frequency High and Low Byte Registers . . . . . 274

Chapter 21. Flash Option Bits . . . . . 276
21.1. Operation . . . . . 276
21.1.1. Option Bit Configuration by Reset . . . . . 276
21.1.2. Option Bit Types . . . . . 277
21.2. Flash Option Bit Control Register Definitions . . . . . 278
21.2.1. User Option Bits . . . . . 278
21.2.2. Trim Bit Data Option Bits . . . . . 281
21.2.3. Trim Bit Address Option Bits . . . . . 281
21.2.4. Trim Bit Address Space . . . . . 282
21.2.5. Zilog Calibration Option Bits . . . . . 289

Chapter 22. Nonvolatile Data Storage . . . . . 290
22.1. Operation . . . . . 290
22.2. NVDS Code Interface . . . . . 290
22.2.1. Byte Write . . . . . 291
22.2.2. Byte Read . . . . . 291
22.2.3. Power Failure Protection . . . . . 292
22.2.4. Optimizing NVDS Memory Usage for Execution Speed . . . . . 293

Chapter 23. On-Chip Debugger . . . . . 294
23.1. Architecture . . . . . 294

23.2. Operation of the On-Chip Debugger Interface . . . . . 295
23.2.1. DEBUG Mode . . . . . 297
23.2.2. OCD Data Format . . . . . 298
23.2.3. OCD Autobaud Detector/Generator . . . . . 298
23.2.4. High Speed Synchronous . . . . . 299
23.2.5. OCD Serial Errors . . . . . 300
23.2.6. Automatic Reset . . . . . 301
23.2.7. Transmit Flow Control . . . . . 301
23.2.8. Breakpoints . . . . . 301
23.2.9. OCDCNTR Register . . . . . 302
23.3. On-Chip Debugger Commands . . . . . 303
23.4. On-Chip Debugger Control Register Definitions . . . . . 309
23.4.1. OCD Control Register . . . . . 310
23.4.2. OCD Status Register . . . . . 312
23.4.3. Line Control Register . . . . . 313
23.4.4. Baud Reload Register . . . . . 314

Chapter 24. Oscillator Control . . . . . 315
24.1. Operation . . . . . 315
24.1.1. System Clock Selection . . . . . 315
24.1.2. Clock Failure Detection and Recovery . . . . . 317
24.2. Peripheral Clock . . . . . 318
24.3. Oscillator Control Register Definitions . . . . . 318
24.3.1. Oscillator Control 0 Register . . . . . 318
24.3.2. Oscillator Control1 Register . . . . . 320

Chapter 25. Crystal Oscillator . . . . . 321
25.1. Operating Modes . . . . . 321
25.2. Main Crystal Oscillator Operation . . . . . 322
25.3. Main Oscillator Operation with External RC Network . . . . . 323
25.4. Secondary Crystal Oscillator Operation . . . . . 325

Chapter 26. Internal Precision Oscillator . . . . . 327
26.1. Operation . . . . . 327

Chapter 27. eZ8 CPU Instruction Set . . . . . 328
27.1. Assembly Language Programming Introduction . . . . . 328
27.2. Assembly Language Syntax . . . . . 329
27.3. eZ8 CPU Instruction Notation . . . . . 330
27.4. eZ8 CPU Instruction Classes . . . . . 331
27.5. eZ8 CPU Instruction Summary . . . . . 336

Chapter 28. Op Code Maps . . . . . 345

Chapter 29. Electrical Characteristics . . . . . 349

29.1. Absolute Maximum Ratings . . . . . 349

29.2. DC Characteristics . . . . . 350

29.3. AC Characteristics . . . . . 357

29.4. On-Chip Peripheral AC and DC Electrical Characteristics . . . . . 358

29.4.1. General Purpose I/O Port Input Data Sample Timing . . . . . 366

29.4.2. General Purpose I/O Port Output Timing . . . . . 367

29.4.3. On-Chip Debugger Timing . . . . . 368

29.4.4. UART Timing . . . . . 369

Chapter 30. Packaging . . . . . 371

Chapter 31. Ordering Information . . . . . 372

31.1. Part Number Suffix Designations . . . . . 375

Index . . . . . 377

Customer Support . . . . . 387

# List of Figures

Figure 1. F1680 Series MCU Block Diagram . . . . . 3
Figure 2. Z8F2480, Z8F1680 and Z8F0880 in 20-Pin SOIC, SSOP or PDIP Packages . . . . . 11
Figure 3. Z8F2480, Z8F1680 and Z8F0880 in 28-Pin SOIC, SSOP or PDIP Packages . . . . . 11
Figure 4. Z8F2480, Z8F1680 and Z8F0880 in 40-Pin Dual Inline Package (PDIP) . 12
Figure 5. Z8F2480, Z8F1680 and Z8F0880 in 44-Pin Low-Profile Quad Flat Package (LQFP) or Quad Flat No Lead (QFN) . . . . . 13
Figure 6. Power-On Reset Operation . . . . . 34
Figure 7. Power-On Reset Timing . . . . . 35
Figure 8. Voltage Brown-Out Reset Operation . . . . . 36
Figure 9. GPIO Port Pin Block Diagram . . . . . 47
Figure 10. Interrupt Controller Block Diagram . . . . . 70
Figure 11. Timer Block Diagram . . . . . 85
Figure 12. Noise Filter System Block Diagram . . . . . 107
Figure 13. Noise Filter Operation . . . . . 108
Figure 14. Multi-Channel Timer Block Diagram . . . . . 121
Figure 15. Count Modulo Mode . . . . . 123
Figure 16. Count Up/Down Mode . . . . . 123
Figure 17. Count Up/Down Mode with PWM Channel Outputs and Deadband . . . . 127
Figure 18. Count Max Mode with Channel Compare . . . . . 128
Figure 19. LIN-UART Block Diagram . . . . . 145
Figure 20. LIN-UART Asynchronous Data Format without Parity . . . . . 146
Figure 21. LIN-UART Asynchronous Data Format with Parity . . . . . 146
Figure 22. LIN-UART Driver Enable Signal Timing with One Stop Bit and Parity . 151
Figure 23. LIN-UART Asynchronous MULTIPROCESSOR Mode Data Format . . 152
Figure 24. LIN-UART Receiver Interrupt Service Routine Flow . . . . . 159
Figure 25. Noise Filter System Block Diagram . . . . . 161
Figure 26. Noise Filter Operation . . . . . 162

Figure 27. Infrared Data Communication System Block Diagram . . . . . 182
Figure 28. Infrared Data Transmission . . . . . 183
Figure 29. IrDA Data Reception . . . . . 184
Figure 30. Analog-to-Digital Converter Block Diagram . . . . . 187
Figure 31. ADC Timing Diagram . . . . . 188
Figure 32. ADC Convert Timing . . . . . 188
Figure 33. ESPI Block Diagram . . . . . 198
Figure 34. ESPI Timing when PHASE=0 . . . . . 202
Figure 35. ESPI Timing when PHASE = 1 . . . . . 203
Figure 36. SPI Mode (SSMD = 00) . . . . . 205
Figure 37. Synchronous Frame Sync Pulse mode (SSMD = 10) . . . . . 206
Figure 38. Synchronous Message Framing Mode (SSMD = 11), Multiple Frames . . 207
Figure 39. ESPI Configured as an SPI Master in a Single Master, Single Slave System . . . . . 208
Figure 40. ESPI Configured as an SPI Master in a Single Master, Multiple Slave System . . . . . 208
Figure 41. ESPI Configured as an SPI Slave . . . . . 210
Figure 42. I2C Controller Block Diagram . . . . . 224
Figure 43. Data Transfer Format—Master Write Transaction with a 7-Bit Address . 230
Figure 44. Data Transfer Format—Master Write Transaction with a 10-Bit Address 231
Figure 45. Data Transfer Format—Master Read Transaction with a 7-Bit Address . 233
Figure 46. Data Transfer Format—Master Read Transaction with a 10-Bit Address 234
Figure 47. Data Transfer Format—Slave Receive Transaction with 7-Bit Address . . 238
Figure 48. Data Transfer Format—Slave Receive Transaction with 10-Bit Address . 239
Figure 49. Data Transfer Format—Slave Transmit Transaction with 7-bit Address . 240
Figure 50. Data Transfer Format—Slave Transmit Transaction with 10-Bit Address 242
Figure 51. 8KB Flash Memory Arrangement . . . . . 263
Figure 52. 16KB Flash Memory Arrangement . . . . . 264
Figure 53. 24KB Flash Memory Arrangement . . . . . 265
Figure 54. Flowchart: Flash Controller Operation . . . . . 266
Figure 55. On-Chip Debugger Block Diagram . . . . . 294

Figure 56. Target OCD Connector Interface . . . . . 296
Figure 57. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #1 of 2 . . . . . 296
Figure 58. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #2 of 2 . . . . . 297
Figure 59. OCD Data Format . . . . . 298
Figure 60. Synchronous Operation . . . . . 300
Figure 61. Start Bit Flow Control . . . . . 301
Figure 62. Recommended 20 MHz Crystal Oscillator Configuration . . . . . 322
Figure 63. Connecting the On-Chip Oscillator to an External RC Network . . . . . 323
Figure 64. Typical RC Oscillator Frequency as a Function of External Capacitance 324
Figure 65. Recommended 32 kHz Crystal Oscillator Configuration . . . . . 325
Figure 66. Op Code Map Cell Description . . . . . 345
Figure 67. First Op Code Map . . . . . 347
Figure 68. Second Op Code Map after 1FH . . . . . 348
Figure 69. Typical Active Flash Mode Supply Current (1–20 MHz) . . . . . 353
Figure 70. Typical Active PRAM Mode Supply Current (1–20 MHz) . . . . . 354
Figure 71. Typical Active Flash Mode Supply Current (32–900 kHz) . . . . . 354
Figure 72. Typical Active PRAM Mode Supply Current (32–900 kHz) . . . . . 355
Figure 73. STOP Mode Current Consumption as a Function of $V_{DD}$ with Temperature as a Parameter; all Peripherals Disabled . . . . . 356
Figure 74. VDD Versus Maximum System Clock Frequency . . . . . 357
Figure 75. Port Input Sample Timing . . . . . 366
Figure 76. GPIO Port Output Timing . . . . . 367
Figure 77. On-Chip Debugger Timing . . . . . 368
Figure 78. UART Timing With CTS . . . . . 369
Figure 79. UART Timing Without CTS . . . . . 370

# List of Tables

Table 1. Z8 Encore! XP F1680 Series Part Selection Guide . . . . . 2
Table 2. F1680 Series MCU Acronyms . . . . . 8
Table 3. Z8 Encore! XP F1680 Series Package Options . . . . . 10
Table 4. Signal Descriptions . . . . . 14
Table 5. Pin Characteristics (20-, 28-, 40- and 44-pin Devices). . . . . 17
Table 6. F1680 Series MCU Program Memory Maps . . . . . 20
Table 7. F1680 Series MCU Flash Memory Information Area Map . . . . . 22
Table 8. Register File Address Map . . . . . 23
Table 9. Reset and Stop Mode Recovery Characteristics and Latency . . . . . 32
Table 10. Reset Sources and Resulting Reset Type . . . . . 33
Table 11. Stop Mode Recovery Sources and Resulting Action . . . . . 38
Table 12. Reset Status Register (RSTSTAT) . . . . . 40
Table 13. Reset Status Per Event . . . . . 41
Table 14. Power Control Register 0 (PWRCTL0) . . . . . 44
Table 15. Setup Condition for LVD and VBO Circuits in Different Operation Modes 45
Table 16. Port Availability by Device and Package Type . . . . . 46
Table 17. Port Alternate Function Mapping, 20-Pin Parts1,2. . . . . 49
Table 18. Port Alternate Function Mapping, 28-Pin Parts1,2. . . . . 51
Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts1,2 . . . . . 54
Table 20. GPIO Port Registers and Subregisters . . . . . 58
Table 21. Port A–E GPIO Address Registers (PxADDR) . . . . . 59
Table 22. Port A–E Control Registers (PxCTL). . . . . 60
Table 23. Port A–E Data Direction Subregisters (PxDD) . . . . . 60
Table 24. Port A–E Alternate Function Subregisters (PxAF). . . . . 61
Table 25. Port A–E Output Control Subregisters (PxOC) . . . . . 62
Table 26. Port A–E High Drive Enable Subregisters (PxHDE) . . . . . 62
Table 27. Port A–E Stop Mode Recovery Source Enable Subregisters (PxSMRE). . . 63
Table 28. Port A–E Pull-Up Enable Subregisters (PxPUE) . . . . . 63

Table 29. Port A–E Alternate Function Set 1 Subregisters (PxAFS1) . . . . . 64
Table 30. Port A–E Alternate Function Set 2 Subregisters (PxAFS2) . . . . . 65
Table 31. Port A–E Input Data Registers (PxIN) . . . . . 65
Table 32. Port A–E Output Data Register (PxOUT) . . . . . 66
Table 33. LED Drive Enable (LEDEN) . . . . . 66
Table 34. LED Drive Level High Bit Register (LEDLVLH) . . . . . 67
Table 35. LED Drive Level Low Bit Register (LEDLVLL) . . . . . 67
Table 36. Trap and Interrupt Vectors in Order of Priority . . . . . 69
Table 37. Interrupt Request 0 Register (IRQ0) . . . . . 73
Table 38. Interrupt Request 1 Register (IRQ1) . . . . . 74
Table 39. Interrupt Request 2 Register (IRQ2) . . . . . 75
Table 40. IRQ0 Enable and Priority Encoding . . . . . 76
Table 41. IRQ0 Enable High Bit Register (IRQ0ENH) . . . . . 76
Table 42. IRQ0 Enable Low Bit Register (IRQ0ENL) . . . . . 77
Table 43. IRQ1 Enable and Priority Encoding . . . . . 77
Table 44. IRQ1 Enable High Bit Register (IRQ1ENH) . . . . . 78
Table 45. IRQ2 Enable and Priority Encoding . . . . . 79
Table 46. IRQ1 Enable Low Bit Register (IRQ1ENL) . . . . . 79
Table 47. IRQ2 Enable High Bit Register (IRQ2ENH) . . . . . 80
Table 48. IRQ2 Enable Low Bit Register (IRQ2ENL) . . . . . 81
Table 49. Interrupt Edge Select Register (IRQES) . . . . . 82
Table 50. Shared Interrupt Select Register (IRQSS) . . . . . 82
Table 51. Interrupt Control Register (IRQCTL) . . . . . 83
Table 52. Timer Operating Modes . . . . . 87
Table 53. TRIGGERED ONE-SHOT Mode Initialization Example . . . . . 89
Table 54. DEMODULATION Mode Initialization Example . . . . . 105
Table 55. Timer 0–2 High Byte Register (TxH) . . . . . 109
Table 56. Timer 0–2 Low Byte Register (TxL) . . . . . 109
Table 57. Timer 0–2 Reload High Byte Register (TxRH) . . . . . 110
Table 58. Timer 0–2 Reload Low Byte Register (TxRL) . . . . . 110

Table 59. Timer 0–2 PWM0 High Byte Register (TxPWM0H) . . . . . 110
Table 60. Timer 0-2 PWM1 High Byte Register (TxPWM1H) . . . . . 111
Table 61. Timer 0–2 PWM1 Low Byte Register (TxPWM1L) . . . . . 111
Table 62. Timer 0–2 PWM0 Low Byte Register (TxPWM0L) . . . . . 111
Table 63. Timer 0–2 Control 0 Register (TxCTL0) . . . . . 112
Table 64. Timer 0–2 Control 1 Register (TxCTL1) . . . . . 113
Table 65. Timer 0–2 Control 2 Register (TxCTL2) . . . . . 117
Table 66. Timer 0–2 Status Register (TxSTAT) . . . . . 118
Table 67. Timer 0–2 Noise Filter Control Register (TxNFC) . . . . . 119
Table 68. Timer Count Modes . . . . . 122
Table 69. Multi-Channel Timer Address Map . . . . . 129
Table 70. Multi-Channel Timer High and Low Byte Registers (MCTH, MCTL) . . . 130
Table 71. Multi-Channel Timer Reload High and Low Byte Registers (MCTRH, MCTRL) . . . . . 131
Table 72. Multi-Channel Timer Subaddress Register (MCTSA) . . . . . 132
Table 73. Multi-Channel Timer Subregister x (MCTSRx) . . . . . 132
Table 74. Multi-Channel Timer Control 0 Register (MCTCTL0) . . . . . 132
Table 75. Multi-Channel Timer Control 1 Register (MCTCTL1) . . . . . 134
Table 76. Multi-Channel Timer Channel Status 0 Register (MCTCHS0) . . . . . 135
Table 77. Multi-Channel Timer Channel Status 1 Register (MCTCHS1) . . . . . 136
Table 78. Multi-Channel Timer Channel Control Register (MCTCHyCTL) . . . . . 137
Table 79. Multi-Channel Timer Channel-y High Byte Registers (MCTCHyH)* . . . 139
Table 80. Watchdog Timer Approximate Time-Out Delays . . . . . 141
Table 81. Watchdog Timer Reload High Byte Register (WDTH = FF2H) . . . . . 143
Table 82. Watchdog Timer Reload Low Byte Register . . . . . 143
Table 83. LIN-UART Transmit Data Register . . . . . 163
Table 84. LIN-UART Receive Data Register . . . . . 164
Table 85. LIN-UART Status 0 Register—Standard UART Mode (U0STAT0 = F41H) . . . . . 165
Table 86. LIN-UART Status 0 Register—LIN Mode (U0STAT0 = F41H) . . . . . 166
Table 87. LIN-UART Mode Select and Status Register (U0MDSTAT = F44H) . . . 168

Table 88. Mode Status Fields . . . . . 169
Table 89. LIN-UART Control 0 Register (U0CTL0 = F42H) . . . . . 170
Table 90. Multiprocessor Control Register (U0CTL1 = F43H with MSEL = 000b) . 172
Table 91. Noise Filter Control Register (U0CTL1 = F43H with MSEL = 001b) . . . 174
Table 92. LIN Control Register (U0CTL1 = F43H with MSEL = 010b). . . . . 175
Table 93. LIN-UART Address Compare Register (U0ADDR = F45H) . . . . . 177
Table 94. LIN-UART Baud Rate High Byte Register (U0BRH = F46H) . . . . . 177
Table 95. LIN-UART Baud Rate Low Byte Register (U0BRL = F47H). . . . . 178
Table 96. LIN-UART Baud Rates, 20.0 MHz System Clock . . . . . 179
Table 97. LIN-UART Baud Rates, 10.0 MHz System Clock . . . . . 179
Table 98. LIN-UART Baud Rates, 5.5296 MHz System Clock . . . . . 180
Table 99. LIN-UART Baud Rates, 3.579545 MHz System Clock . . . . . 180
Table 100. LIN-UART Baud Rates, 1.8432 MHz System Clock . . . . . 181
Table 101. ADC Control Register 0 (ADCCTL0) . . . . . 189
Table 102. ADC Raw Data High Byte Register (ADCRD_H). . . . . 191
Table 103. ADC Data High Byte Register (ADCD_H) . . . . . 191
Table 104. ADC Data Low Bits Register (ADCD_L) . . . . . 192
Table 105. Sample Settling Time (ADCSST). . . . . 193
Table 106. Sample Time (ADCST) . . . . . 194
Table 107. ADC Clock Prescale Register (ADCCP) . . . . . 195
Table 108. ESPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation . . 201
Table 109. ESPI Data Register (ESPIDATA) . . . . . 214
Table 110. ESPI Transmit Data Command and Receive Data Buffer Control Register (ESPITDCR) . . . . . 214
Table 111. ESPI Control Register. . . . . 215
Table 112. ESPI Mode Register (ESPIMODE) . . . . . 217
Table 113. ESPI Status Register (ESPISTAT) . . . . . 219
Table 114. ESPI State Register (ESPISTATE). . . . . 220
Table 115. ESPISTATE Values . . . . . 221
Table 116. ESPI Baud Rate High Byte Register (ESPIBRH) . . . . . 222
Table 117. ESPI Baud Rate Low Byte Register (ESPIBRL) . . . . . 222

Table 118. I2C Master/Slave Controller Registers . . . . . 224
Table 119. I2C Data Register (I2CDATA = F50H) . . . . . 244
Table 120. I2C Interrupt Status Register (I2CISTAT = F51H) . . . . . 245
Table 121. I2C Control Register (I2CCTL) . . . . . 247
Table 122. I2C Baud Rate High Byte Register (I2CBRH = 53H) . . . . . 248
Table 123. I2C Baud Rate Low Byte Register (I2CBRL = F54H). . . . . 249
Table 124. I2C State Register (I2CSTATE)—Description when DIAG = 1 . . . . . 250
Table 125. I2C State Register (I2CSTATE)—Description when DIAG = 0 . . . . . 251
Table 126. I2CSTATE_L . . . . . 252
Table 127. I2CSTATE_H . . . . . 252
Table 128. I2C Mode Register (I2C Mode = F56H) . . . . . 253
Table 129. I2C Slave Address Register (I2CSLVAD = 57H) . . . . . 255
Table 130. Comparator 0 Control Register (CMP0). . . . . 257
Table 131. Comparator 1 Control Register (CMP1). . . . . 258
Table 132. Z8 Encore! XP F1680 Series Flash Memory Configurations. . . . . 262
Table 133. Flash Code Protection Using the Flash Option Bit. . . . . 268
Table 134. Flash Control Register (FCTL). . . . . 272
Table 135. Flash Status Register (FSTAT). . . . . 272
Table 136. Flash Page Select Register (FPS) . . . . . 273
Table 137. Flash Sector Protect Register (FPROT) . . . . . 274
Table 138. Flash Frequency High Byte Register (FFREQH). . . . . 275
Table 139. Flash Frequency Low Byte Register (FFREQL) . . . . . 275
Table 140. Flash Option Bits at Program Memory Address 0000H. . . . . 278
Table 141. Flash Option Bits at Program Memory Address 0001H. . . . . 280
Table 142. Trim Bit Data Register (TRMDR) . . . . . 281
Table 143. Trim Bit Address Register (TRMADR) . . . . . 281
Table 144. Trim Bit Address Map . . . . . 281
Table 145. Trim Bit Address Description. . . . . 282
Table 146. Trim Option Bits at Address 0000H (TTEMP0) . . . . . 282
Table 147. Trim Option Bits at 0001H (TTEMP1) . . . . . 283

Table 148. Trim Option Bits at 0002H (TIPO) . . . . . 283
Table 149. Trim Option Bits at Address 0003H (TLVD_VBO) . . . . . 284
Table 150. LVD_Trim Values . . . . . 284
Table 151. Trim Option Bits at 0004H (TCOMP_ADC) . . . . . 286
Table 152. Truth Table of HYS . . . . . 286
Table 153. Trim Option Bits at 0005H (TVREF) . . . . . 287
Table 154. Trim Option Bits at 0006H (TBG) . . . . . 287
Table 155. Trim Option Bits at 0007H (TFilter0) . . . . . 288
Table 156. Trim Option Bits at 0008H (TFilter1) . . . . . 288
Table 157. Temperature Sensor Calibration High Byte at FE60H (TEMPCALH) . . . 289
Table 158. Temperature Sensor Calibration Low Byte at FE61H (TEMPCALL) . . . . 289
Table 159. Write Status Byte . . . . . 291
Table 160. Read Status Byte . . . . . 292
Table 161. NVDS Read Time . . . . . 293
Table 162. OCD Baud-Rate Limits . . . . . 299
Table 163. On-Chip Debugger Commands . . . . . 304
Table 164. OCD Control Register (OCDCTL) . . . . . 310
Table 165. OCD Status Register (OCDSTAT) . . . . . 312
Table 166. OCD Line Control Register (OCDLCR) . . . . . 313
Table 167. Baud Reload Register . . . . . 314
Table 168. Oscillator Configuration and Selection . . . . . 316
Table 169. Peripheral Clock Source and Usage . . . . . 318
Table 170. Oscillator Control 0 Register (OSCCTL0) . . . . . 319
Table 171. Oscillator Control 1 Register (OSCCTL1) . . . . . 320
Table 172. Recommended Crystal Oscillator Specifications . . . . . 323
Table 173. Recommended Crystal Oscillator Specifications . . . . . 326
Table 174. Assembly Language Syntax Example 1 . . . . . 329
Table 175. Assembly Language Syntax Example 2 . . . . . 329
Table 176. Notational Shorthand . . . . . 330
Table 177. Additional Symbols . . . . . 331

Table 178. Arithmetic Instructions . . . . . 332
Table 179. Bit Manipulation Instructions . . . . . 333
Table 180. Block Transfer Instructions . . . . . 333
Table 181. CPU Control Instructions . . . . . 333
Table 182. Logical Instructions . . . . . 334
Table 183. Load Instructions . . . . . 334
Table 184. Program Control Instructions . . . . . 335
Table 185. Rotate and Shift Instructions. . . . . 335
Table 186. eZ8 CPU Instruction Summary. . . . . 336
Table 187. Op Code Map Abbreviations . . . . . 346
Table 188. Absolute Maximum Ratings* . . . . . 349
Table 189. DC Characteristics . . . . . 350
Table 190. Supply Current Characteristics . . . . . 352
Table 191. AC Characteristics . . . . . 357
Table 192. Power-On Reset and Voltage Brown-Out Electrical Characteristics and Timing . . . . . 358
Table 193. Flash Memory Electrical Characteristics and Timing . . . . . 359
Table 194. Watchdog Timer Electrical Characteristics and Timing. . . . . 359
Table 195. Non-Volatile Data Storage . . . . . 359
Table 196. Analog-to-Digital Converter Electrical Characteristics and Timing . . . . . 360
Table 197. Comparator Electrical Characteristics . . . . . 361
Table 198. Temperature Sensor Electrical Characteristics . . . . . 361
Table 199. Low Power Operational Amplifier Characteristics . . . . . 362
Table 200. IPO Electrical Characteristics . . . . . 362
Table 201. Low Voltage Detect Electrical Characteristics . . . . . 363
Table 202. Crystal Oscillator Characteristics . . . . . 364
Table 203. Low Power 32 kHz Secondary Oscillator Characteristics . . . . . 365
Table 204. GPIO Port Input Timing . . . . . 366
Table 205. GPIO Port Output Timing. . . . . 367
Table 206. On-Chip Debugger Timing . . . . . 368
Table 207. UART Timing with CTS . . . . . 369

Table 208. UART Timing Without CTS . . . . . 370
Table 209. Ordering Information for the Z8 Encore! XP F1680 Series of MCUs . . . . 372
Table 210. Package and Pin Count Description . . . . . 376

# Chapter 1. Overview

Zilog's F1680 Series of MCUs is based on Zilog's advanced 8-bit eZ8 CPU core. This microcontroller, a member of the Z8 Encore! XP® product line, is optimized for low-power applications and supports 1.8V to 3.6V of low-voltage operation with extremely low Active, Halt and Stop Mode currents, plus it offers a wide assortment of speed and low-power options. In addition, the feature-rich analog and digital peripherals of the Z8 Encore! XP F1680 Series of MCUs makes them suitable for a variety of applications including safety and security, utility metering, digital power supervisory, hand-held electronic devices and general motor control applications.

For simplicity, the remainder of this document refers to the entire Z8 Encore! XP F1680 Series of MCUs as the F1680 Series MCU.

## 1.1. Features

Key features of the F1680 Series MCU include:

- 20MHz eZ8 CPU core
- 8KB, 16KB, or 24KB Flash memory with in-circuit programming capability
- 1KB or 2KB Register RAM
- 1KB Program RAM for program code shadowing and data storage (optional)
- 128B or 256B 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)
- $I^2C$ 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 Brown-Out (VBO) protection
- Built-in Low-Voltage Detection (LVD) with programmable voltage threshold
- 32kHz secondary oscillator for Timers
- Internal Precision Oscillator (IPO) with output frequency in the range of 43.2kHz to 11.0592MHz
- Crystal oscillator with three power settings and external RC network option
- Wide operation voltage range: 1.8V–3.6V
- 20-, 28-, 40- and 44-pin packages
- 0°C to +70°C (standard) and –40°C to +105°C (extended) operating temperature ranges

## 1.2. Part Selection Guide

Table 1 displays basic features and package styles available for each of the F1680 Series MCUs.

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

| Part Number | Flash (KB) | RAM (B) | Program RAM (B) | NVDS (B) | I/O | ADC Inputs | SPI | $I^2C$ | 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 |

## 1.3. Block Diagram

Figure 1 displays the architecture of the F1680 Series MCU.

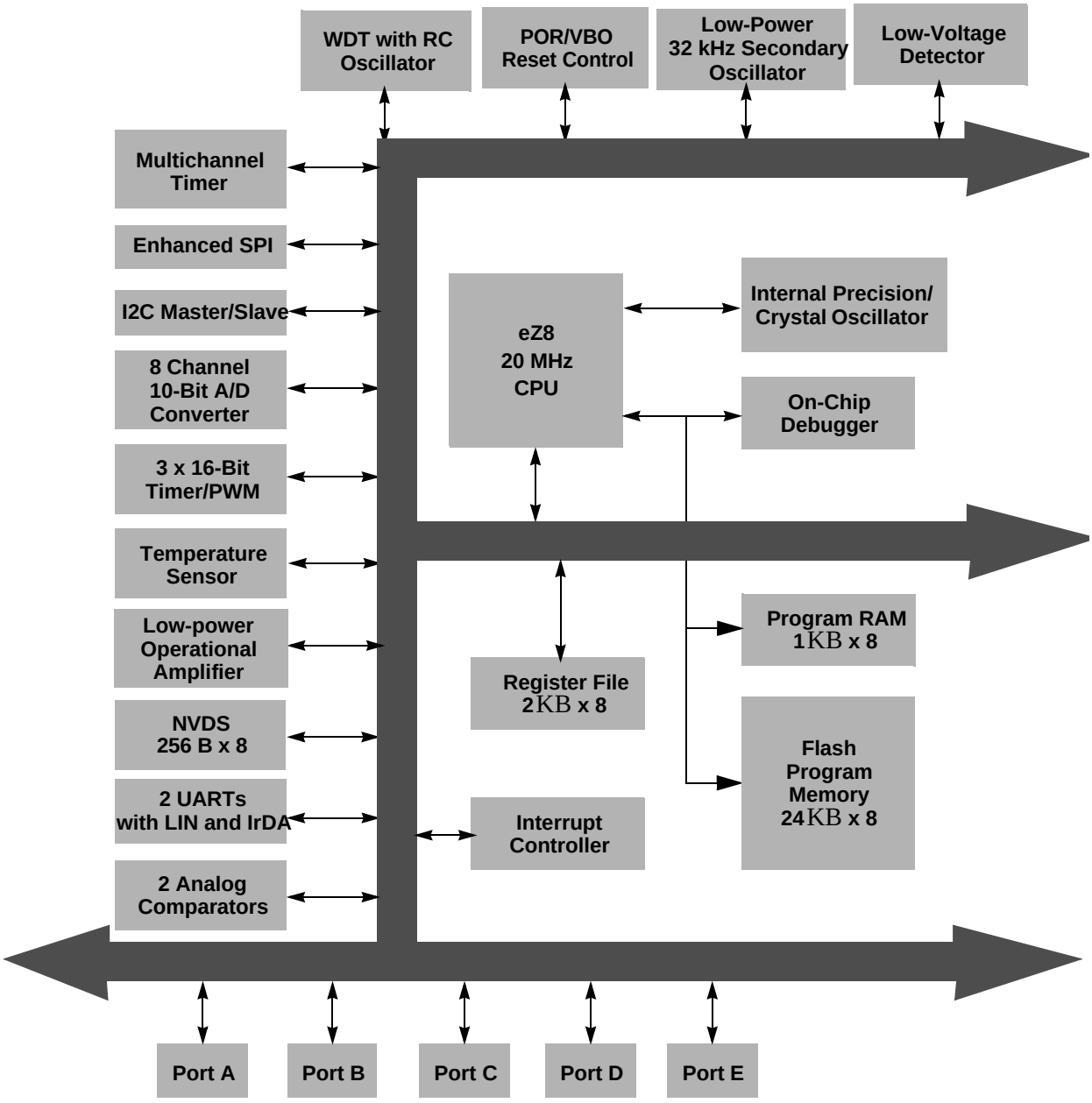



**Figure 1. F1680 Series MCU Block Diagram**

# 1.4. An Overview of the eZ8 CPU and its Peripherals

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® 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 subroutine calls and interrupts more than hardware stacks
- Compatible with existing Z8 code
- Expanded internal Register File allows access up to 4KB
- New instructions improve execution efficiency for code developed using higher-level programming languages including C
- Pipelined instruction fetch and execution
- New instructions for improved performance including BIT, BSWAP, BTJ, CPC, LDC, LDCI, LEA, MULT and SRL
- New instructions support 12-bit linear addressing of the register file
- Up to 10 MIPS operation
- C-Compiler friendly
- 2 to 9 clock cycles per instruction

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

## 1.4.1. General-Purpose Input/Output

The F1680 MCU 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.

## 1.4.2. Flash Controller

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

### 1.4.3. Non-Volatile Data Storage

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

### 1.4.4. 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 (43.2kHz to 11.0592MHz) and is available with factory-trimmed calibration data.

### 1.4.5. Crystal Oscillator

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

### 1.4.6. Secondary Oscillator

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

### 1.4.7. 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.

### 1.4.8. 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.

### 1.4.9. 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.

## 1.4.10. 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.

## 1.4.11. Low-Voltage Detector

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

## 1.4.12. Enhanced SPI

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

## 1.4.13. 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 20KBits.

## 1.4.14. Master/Slave $I^2C$

The inter-integrated circuit ($I^2C$) controller makes the F1680 Series MCU compatible with the $I^2C$ protocol. The $I^2C$ controller consists of two bidirectional bus lines:

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

It also supports Master, Slave and Multi-Master Operations

## 1.4.15. 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.

## 1.4.16. 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.

## 1.4.17. 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.

## 1.4.18. Reset Controller

The F1680 Series MCU is reset using the $\overline{\text{RESET}}$ pin, POR, WDT time-out, STOP Mode exit, or VBO warning signal. The $\overline{\text{RESET}}$ pin is bidirectional, that is, it functions as reset source as well as a reset indicator.

## 1.4.19. On-Chip Debugger

The F1680 Series MCU features 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.

## 1.4.20. 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.

## 1.5. Acronyms and Expansions

This document uses the acronyms and expansions listed in Table 2.

**Table 2. F1680 Series MCU Acronyms**

| 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 Brown-Out |
| IPO | Internal Precision Oscillator |
| UART | Universal Asynchronous Receiver/Transmitter |
| IrDA | Infrared Data Association |
| $I^2C$ | 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 |

**Table 2. F1680 Series MCU Acronyms (Continued)**

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

# Chapter 2. Pin Description

The F1680 Series MCU is 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 about the physical package specifications, see the Packaging chapter on page 371.

## 2.1. Available Packages

Table 3 lists the package styles available for each device in the Z8 Encore! XP F1680 Series product line.

**Table 3. 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 |

## 2.2. Pin Configurations

Figures 2 through 5 display the pin configurations of all the packages available in the F1680 Series MCU. For description of the signals, see Table 4 on page 14.

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 $\overline{\text{RESET}}$ alternate function.

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

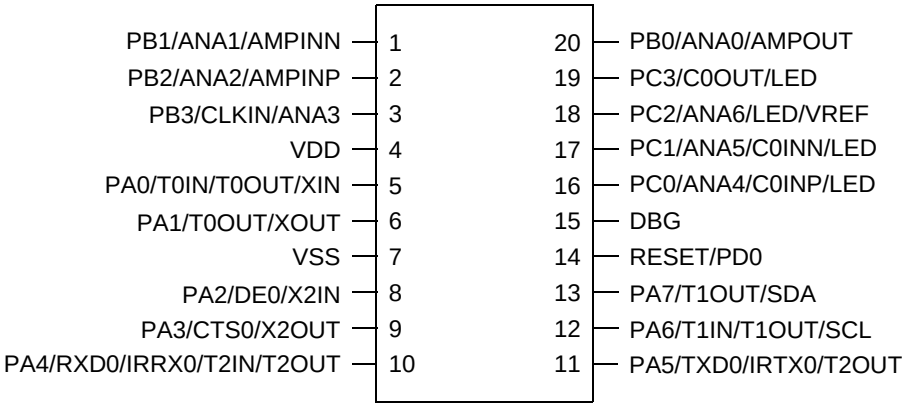



**Figure 2. Z8F2480, Z8F1680 and Z8F0880 in 20-Pin SOIC, SSOP or PDIP Packages**

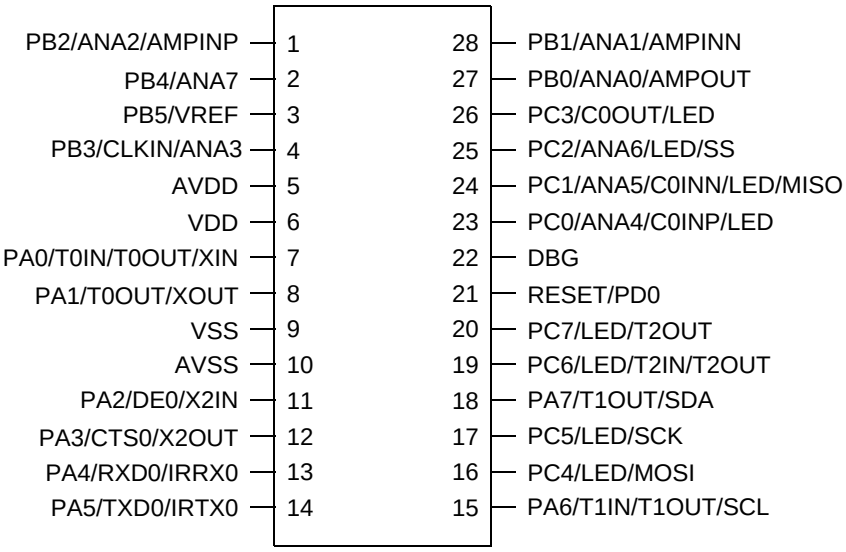



**Figure 3. Z8F2480, Z8F1680 and Z8F0880 in 28-Pin SOIC, SSOP or PDIP Packages**

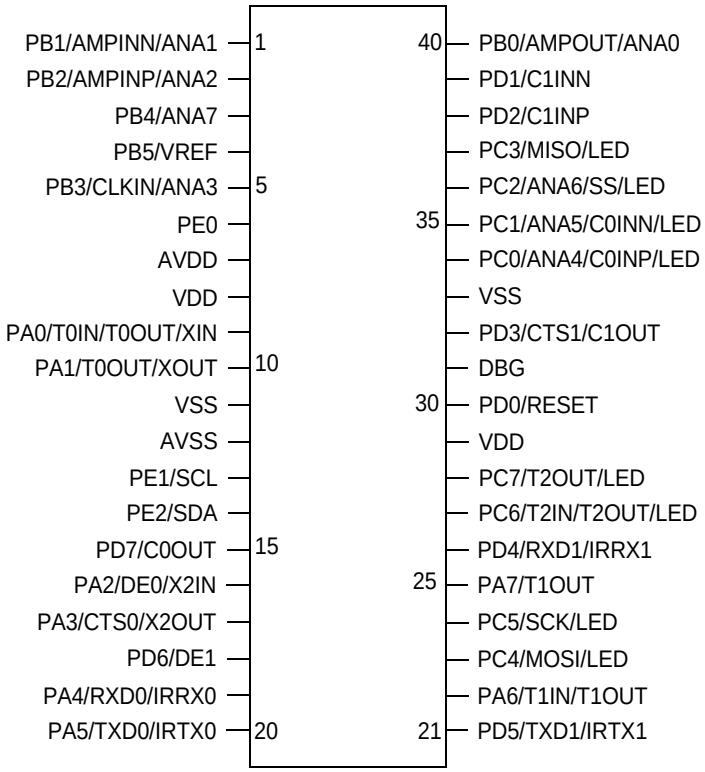



**Figure 4. Z8F2480, Z8F1680 and Z8F0880 in 40-Pin Dual Inline Package (PDIP)**

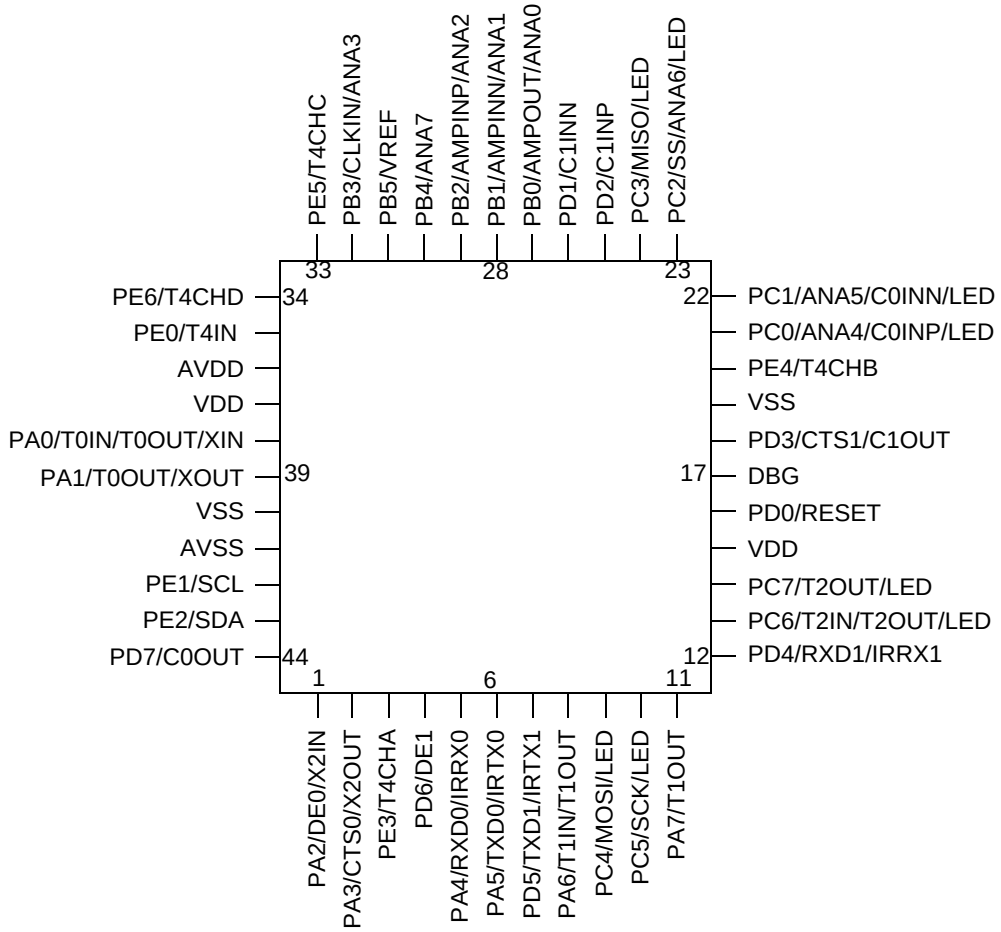



**Figure 5. Z8F2480, Z8F1680 and Z8F0880 in 44-Pin Low-Profile Quad Flat Package (LQFP) or Quad Flat No Lead (QFN)**

## 2.3. Signal Descriptions

Table 4 describes the signals for each block on the F1680 Series MCU. To determine the signals available for specific package styles, see the Pin Configurations chapter on page 10.

**Table 4. Signal Descriptions**

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **General-Purpose I/O Ports A–E** | | |
| 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. |
| **LIN-UART Controllers** | | |
| 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. |
| **$I^2C$ Controller** | | |
| SCL | I/O | $I^2C$ Serial Clock: The $I^2C$ Master supplies this signal. If the F1680 Series MCU is the $I^2C$ Master, this pin is an output. If it is the $I^2C$ slave, this pin is an input. When the GPIO pin is configured as an alternate function to enable the SCL function, this pin is open-drain. |
| SDA | I/O | Serial Data: This open-drain pin transfers data between the $I^2C$ and an external $I^2C$ Master/Slave. When the GPIO pin is configured as an alternate function to enable the SDA function, this pin is open-drain. |
| **ESPI Controller** | | |
| $\overline{SS}$ | I/O | Slave Select: This signal can be an output or an input. If the F1680 Series MCU is the SPI master, this pin can be configured as the Slave Select output. If it is the SPI slave, this pin is the input slave select. |

**Table 4. Signal Descriptions (Continued)**

| Signal Mnemonic | I/O | Description |
|---|---|---|
| SCK | I/O | SPI Serial Clock: The SPI master supplies this signal. If the F1680 Series MCU is the SPI master, this pin is an output. 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. |
| **Timers** | | |
| T0OUT/T1OUT/ T2OUT | O | Timer Output 0–2: These signals are output from the timers. |
| $\overline{\text{T0OUT}}$/$\overline{\text{T1OUT}}$/ $\overline{\text{T2OUT}}$ | 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 $\overline{\text{T0OUT}}$/T1OUT/ $\overline{\text{T2OUT}}$ signals. |
| **Multi-Channel Timers** | | |
| 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. |
| **Comparators** | | |
| 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. |
| **Analog** | | |
| 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. |
| **Low-Power Operational Amplifier** | | |
| 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. |

**Table 4. Signal Descriptions (Continued)**

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **Oscillators** | | |
| XIN | I | External Crystal Input: The input pin to the crystal oscillator. A crystal can be connected between the pin and the XOUT pin to form the oscillator. In addition, this pin is used with external RC networks or external clock drivers to provide the system clock. |
| XOUT | O | External Crystal Output: This pin is the output of the crystal oscillator. A crystal can be connected between it and the XIN pin to form the oscillator. |
| X2IN | I | Watch Crystal Input: 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. |
| **Clock Input** | | |
| CLKIN | I | Clock Input Signal: This pin can be used to input a TTL-level signal to be used as the system clock. |
| **LED Drivers** | | |
| LED | O | Direct LED Drive Capability: All Port C pins have the capability to drive an LED without any other external components. These pins have programmable drive strengths set by the GPIO block. |
| **On-Chip Debugger** | | |
| DBG | I/O | Debug: This signal is the control and data input and output of the On-Chip Debugger.<br>**Caution:** The DBG pin is open-drain and requires an external pull-up resistor to ensure proper operation. |
| **Reset** | | |
| 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. |
| **Power Supply** | | |
| $V_{DD}$ | I | Digital Power Supply. |
| $AV_{DD}$ | I | Analog Power Supply. |
| $V_{SS}$ | I | Digital Ground. |
| $AV_{SS}$ | I | Analog Ground. |

Note: The $AV_{DD}$ and $AV_{SS}$ signals are available only in 28-pin, 40-pin and 44-pin packages.

## 2.4. Pin Characteristics

Table 5 provides detailed information about the characteristics of each pin available on the F1680 Series MCU 20-, 28-, 40- and 44-pin devices. Data provided in Table 5 is sorted alphabetically by the pin symbol mnemonic.

**Table 5. Pin Characteristics (20-, 28-, 40- and 44-pin Devices)**

| Symbol Mnemonic | Direction | Reset Direction | Active Low or Active High | Tristate Output | Internal Pull-up or Pull-down | Schmitt Trigger Input | Open Drain Output | 5V Tolerance |
|---|---|---|---|---|---|---|---|---|
| $AV_{DD}$ | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| $AV_{SS}$ | 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, 5V tolerant inputs unless pull-ups are enabled |
| PB[5:0] | I/O | I | N/A | Yes | Programmable pull-up | Yes | Yes, programmable | Yes, 5V tolerant inputs unless pull-ups are enabled |
| PC[7:0] | I/O | I | N/A | Yes | Programmable pull-up | Yes | Yes, programmable | Yes, 5V tolerant inputs unless pull-ups are enabled |
| PD[7:1] | I/O | I | N/A | Yes | Programmable pull-up | Yes | Yes, programmable | Yes, 5V tolerant inputs unless pull-ups are enabled |

**Table 5. 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 | 5V Tolerance |
|---|---|---|---|---|---|---|---|---|
| PE[6:0] | I/O | I | N/A | Yes | Programmable pull-up | Yes | Yes, programmable | Yes, 5V tolerant inputs unless pull-ups are enabled |
| 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, 5V tolerant inputs unless pull-ups are enabled |
| $V_{DD}$ | N/A | N/A | N/A | N/A | | | N/A | N/A |
| $V_{SS}$ | N/A | N/A | N/A | N/A | | | N/A | N/A |

# Chapter 3. Address Space

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 about the eZ8 CPU and its address space, refer to the eZ8 CPU Core User Manual (UM0128), available for download at www.zilog.com.

## 3.1. Register File

The Register File address space in the Z8 Encore!® MCU is 4KB (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 4KB 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. Zilog does not recommend writing to the reserved Register File addresses because doing so can produce unpredictable results.

The on-chip Register RAM always begins at address 000H in the Register File address space. The F1680 Series MCU contains 1KB or 2KB 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, the F1680 Series MCU contains 1KB of on-chip Program RAM. Normally it is used as Program RAM and is present in the Program Memory address space (see the Program Memory section on page 20). However, it can also be used as additional Register RAM present in the Register File address space 800H–BFFH (1KB Program RAM, 2KB Register RAM), or 400H–7FFH (1KB Program RAM, 1KB Register RAM), if you do not

need to use this on-chip Program RAM to shadow Interrupt Service Routines (ISR). For details, see the PRAM_M section on page 278.

## 3.2. Program Memory

The eZ8 CPU supports 64KB of Program Memory address space. The F1680 Series MCU contains 8KB to 24KB of on-chip Flash memory in the Program Memory address space, depending on the device.

In addition, the F1680 Series MCU contains up to 1KB 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 a 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 6 describes the Program Memory maps for the F1680 Series MCU.

**Table 6. F1680 Series MCU Program Memory Maps**

| Program Memory Address (Hex) | Function |
|---|---|
| **Z8F2480 Device** | |
| 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 | 1KB PRAM |

Note: *See Table 36 on page 69 for a list of interrupt vectors and traps.

**Table 6. F1680 Series MCU Program Memory Maps (Continued)**

| Program Memory Address (Hex) | Function |
|---|---|
| **Z8F1680 Device** | |
| 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 | 1KB PRAM |
| **Z8F0880 Device** | |
| 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 | 1KB PRAM |

Note: *See Table 36 on page 69 for a list of interrupt vectors and traps.

## 3.3. Data Memory

The F1680 Series MCU does not use the eZ8 CPU's 64KB Data Memory address space.

## 3.4. Flash Information Area

Table 7 describes the F1680 Series MCU Flash Information Area. This 512-byte 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 512bytes at addresses FE00H to FFFFH. When the Information Area access is enabled, all reads from these Program Memory addresses return the Information Area data rather than the Program Memory data. Access to the Flash Information Area is read-only.

**Table 7. F1680 Series MCU Flash Memory Information Area Map**

| Program Memory Address (Hex) | Function |
|---|---|
| FE00–FE3F | Zilog option bits |
| FE40–FE53 | Part Number:<br>20-character ASCII alphanumeric code<br>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 |

# Chapter 4. Register Map

Table 8 provides an address map to the register file contained in all Z8 Encore! XP F1680 Series devices. Not all devices and package styles in this product series support the ADC, nor all of the GPIO ports. Therefore, consider the registers for unimplemented peripherals to be reserved.

**Table 8. Register File Address Map**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex)[1] | Page # |
|---|---|---|---|---|
| **General Purpose RAM** | | | | |
| **Z8F2480 Device** | | | | |
| 000–7FF | General-Purpose Register File RAM | — | XX | |
| 800–EFF | Reserved[2] | — | XX | |
| **Z8F1680 Device** | | | | |
| 000–7FF | General-Purpose Register File RAM | — | XX | |
| 800–EFF | Reserved[2] | — | XX | |
| **Z8F0880 Device** | | | | |
| 000–3FF | General-Purpose Register File RAM | — | XX | |
| 400–EFF | Reserved[2] | — | XX | |
| **Special Purpose Registers** | | | | |
| **Timer 0** | | | | |
| F00 | Timer 0 High Byte | T0H | 00 | 109 |
| F01 | Timer 0 Low Byte | T0L | 01 | 109 |
| F02 | Timer 0 Reload High Byte | T0RH | FF | 110 |
| F03 | Timer 0 Reload Low Byte | T0RL | FF | 110 |
| F04 | Timer 0 PWM0 High Byte | T0PWM0H | 00 | 110 |
| F05 | Timer 0 PWM0 Low Byte | T0PWM0L | 00 | 111 |
| F06 | Timer 0 Control 0 | T0CTL0 | 00 | 112 |
| F07 | Timer 0 Control 1 | T0CTL1 | 00 | 113 |

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). 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.

**Table 8. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex)[1] | Page # |
|---|---|---|---|---|
| F20 | Timer 0 PWM1 High Byte | T0PWM1H | 00 | 111 |
| F21 | Timer 0 PWM1 Low Byte | T0PWM1L | 00 | 111 |
| F22 | Timer 0 Control 2 | T0CTL2 | 00 | 117 |
| F23 | Timer 0 Status | T0STA | 00 | 118 |
| F2C | Timer 0 Noise Filter Control | T0NFC | 00 | 119 |
| **Timer 1** | | | | |
| F08 | Timer 1 High Byte | T1H | 00 | 109 |
| F09 | Timer 1 Low Byte | T1L | 01 | 109 |
| F0A | Timer 1 Reload High Byte | T1RH | FF | 110 |
| F0B | Timer 1 Reload Low Byte | T1RL | FF | 110 |
| F0C | Timer 1 PWM0 High Byte | T1PWM0H | 00 | 110 |
| F0D | Timer 1 PWM0 Low Byte | T1PWM0L | 00 | 111 |
| F0E | Timer 1 Control 0 | T1CTL0 | 00 | 112 |
| F0F | Timer 1 Control 1 | T1CTL1 | 00 | 113 |
| F24 | Timer 1 PWM1 High Byte | T1PWM1H | 00 | 111 |
| F25 | Timer 1 PWM1 Low Byte | T1PWM1L | 00 | 111 |
| F26 | Timer 1 Control 2 | T1CTL2 | 00 | 117 |
| F27 | Timer 1 Status | T1STA | 00 | 118 |
| F2D | Timer 1 Noise Filter Control | T1NFC | 00 | 119 |
| **Timer 2** | | | | |
| F10 | Timer 2 High Byte | T2H | 00 | 109 |
| F11 | Timer 2 Low Byte | T2L | 01 | 110 |
| F12 | Timer 2 Reload High Byte | T2RH | FF | 110 |
| F13 | Timer 2 Reload Low Byte | T2RL | FF | 110 |
| F14 | Timer 2 PWM0 High Byte | T2PWM0H | 00 | 110 |
| F15 | Timer 2 PWM0 Low Byte | T2PWM0L | 00 | 111 |
| F16 | Timer 2 Control 0 | T2CTL0 | 00 | 112 |

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). 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.

**Table 8. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex)[1] | Page # |
|---|---|---|---|---|
| F17 | Timer 2 Control 1 | T2CTL1 | 00 | 113 |
| F28 | Timer 2 PWM1 High Byte | T2PWM1H | 00 | 111 |
| F29 | Timer 2 PWM1 Low Byte | T2PWM1L | 00 | 111 |
| F2A | Timer 2 Control 2 | T2CTL2 | 00 | 117 |
| F2B | Timer 2 Status | T2STA | 00 | 118 |
| F2E | Timer 2 Noise Filter Control | T2NFC | 00 | 119 |
| F2F–F3F | Reserved | — | XX | |
| **LIN UART 0** | | | | |
| F40 | LIN UART0 Transmit Data | U0TXD | XX | 163 |
| | LIN UART0 Receive Data | U0RXD | XX | 164 |
| F41 | LIN UART0 Status 0—Standard UART Mode | U0STAT0 | 0000011Xb | 165 |
| | LIN UART0 Status 0—LIN Mode | U0STAT0 | 00000110b | 166 |
| F42 | LIN UART0 Control 0 | U0CTL0 | 00 | 170 |
| F43 | LIN UART0 Control 1—Multiprocessor Control | U0CTL1 | 00 | 172 |
| | LIN UART0 Control 1—Noise Filter Control | U0CTL1 | 00 | 174 |
| | LIN UART0 Control 1—LIN Control | U0CTL1 | 00 | 175 |
| F44 | LIN UART0 Mode Select and Status | U0MDSTAT | 00 | 168 |
| F45 | UART0 Address Compare | U0ADDR | 00 | 177 |
| F46 | UART0 Baud Rate High Byte | U0BRH | FF | 177 |
| F47 | UART0 Baud Rate Low Byte | U0BRL | FF | 178 |
| **LIN UART 1** | | | | |
| F48 | LIN UART1 Transmit Data | U1TXD | XX | 163 |
| | LIN UART1 Receive Data | U1RXD | XX | 164 |
| F49 | LIN UART1 Status 0—Standard UART Mode | U1STAT0 | 0000011Xb | 165 |
| | LIN UART1 Status 0—LIN Mode | U1STAT0 | 00000110b | 166 |
| F4A | LIN UART1 Control 0 | U1CTL0 | 00 | 170 |

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). 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.

**Table 8. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex)[1] | Page # |
|---|---|---|---|---|
| F4B | LIN UART1 Control 1—Multiprocessor Control | U1CTL1 | 00 | 172 |
| | LIN UART1 Control 1—Noise Filter Control | U1CTL1 | 00 | 174 |
| | LIN UART1 Control 1—LIN Control | U1CTL1 | 00 | 175 |
| F4C | LIN UART1 Mode Select and Status | U1MDSTAT | 00 | 168 |
| F4D | UART1 Address Compare | U1ADDR | 00 | 177 |
| F4E | UART1 Baud Rate High Byte | U1BRH | FF | 177 |
| F4F | UART1 Baud Rate Low Byte | U1BRL | FF | 178 |
| **$I^2C$** | | | | |
| F50 | $I^2C$ Data | I2CDATA | 00 | 244 |
| F51 | $I^2C$ Interrupt Status | I2CISTAT | 80 | 245 |
| F52 | $I^2C$ Control | I2CCTL | 00 | 247 |
| F53 | $I^2C$ Baud Rate High Byte | I2CBRH | FF | 248 |
| F54 | $I^2C$ Baud Rate Low Byte | I2CBRL | FF | 249 |
| F55 | $I^2C$ State | I2CSTATE | 02 | 251 |
| F56 | $I^2C$ Mode | I2CMODE | 00 | 252 |
| F57 | $I^2C$ Slave Address | I2CSLVAD | 00 | 255 |
| F58-F5F | Reserved | — | XX | |
| **Enhanced Serial Peripheral Interface (ESPI)** | | | | |
| F60 | ESPI Data | ESPIDATA | XX | 214 |
| F61 | ESPI Transmit Data Command | ESPITDCR | 00 | 214 |
| F62 | ESPI Control | ESPICTL | 00 | 215 |
| F63 | ESPI Mode | ESPIMODE | 00 | 217 |
| F64 | ESPI Status | ESPISTAT | 01 | 219 |
| F65 | ESPI State | ESPISTATE | 00 | 220 |
| F66 | ESPI Baud Rate High Byte | ESPIBRH | FF | 220 |
| F67 | ESPI Baud Rate Low Byte | ESPIBRL | FF | 220 |
| F68–F6F | Reserved | — | XX | |

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). 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.

**Table 8. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex)[1] | Page # |
|---|---|---|---|---|
| **Analog-to-Digital Converter (ADC)** | | | | |
| F70 | ADC Control 0 | ADCCTL0 | 00 | 189 |
| F71 | ADC Raw Data High Byte | ADCRD_H | 80 | 191 |
| F72 | ADC Data High Byte | ADCD_H | XX | 191 |
| F73 | ADC Data Low Bits | ADCD_L | XX | 192 |
| F74 | ADC Sample Settling Time | ADCSST | FF | 193 |
| F75 | Sample Time | ADCST | XX | 194 |
| F76 | ADC Clock Prescale Register | ADCCP | 00 | 195 |
| F77–F7F | Reserved | — | XX | |
| **Low-Power Control** | | | | |
| F80 | Power Control 0 | PWRCTL0 | 80 | 44 |
| F81 | Reserved | — | XX | |
| **LED Controller** | | | | |
| F82 | LED Drive Enable | LEDEN | 00 | 66 |
| F83 | LED Drive Level High Bit | LEDLVLH | 00 | 67 |
| F84 | LED Drive Level Low Bit | LEDLVLL | 00 | 67 |
| F85 | Reserved | — | XX | |
| **Oscillator Control** | | | | |
| F86 | Oscillator Control 0 | OSCCTL0 | A0 | 319 |
| F87 | Oscillator Control 1 | OSCCTL1 | 00 | 320 |
| F88–F8F | Reserved | | | |
| **Comparator 0** | | | | |
| F90 | Comparator 0 Control | CMP0 | 14 | 257 |
| **Comparator 1** | | | | |
| F91 | Comparator 1 Control | CMP1 | 14 | 258 |
| F92–F9F | Reserved | — | XX | |

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). 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.

**Table 8. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex)[1] | Page # |
|---|---|---|---|---|
| **Multi-Channel Timer** | | | | |
| FA0 | MCT High Byte | MCTH | 00 | 130 |
| FA1 | MCT Low Byte | MCTL | 00 | 130 |
| FA2 | MCT Reload High Byte | MCTRH | FF | 131 |
| FA3 | MCT Reload Low Byte | MCTRL | FF | 131 |
| FA4 | MCT Subaddress | MCTSA | XX | 132 |
| FA5 | MCT Subregister 0 | MCTSR0 | XX | 132 |
| FA6 | MCT Subregister 1 | MCTSR1 | XX | 132 |
| FA7 | MCT Subregister 2 | MCTSR2 | XX | 132 |
| FA8–FBF | Reserved | — | XX | |
| **Interrupt Controller** | | | | |
| FC0 | Interrupt Request 0 | IRQ0 | 00 | 73 |
| FC1 | IRQ0 Enable High Bit | IRQ0ENH | 00 | 76 |
| FC2 | IRQ0 Enable Low Bit | IRQ0ENL | 00 | 77 |
| FC3 | Interrupt Request 1 | IRQ1 | 00 | 74 |
| FC4 | IRQ1 Enable High Bit | IRQ1ENH | 00 | 78 |
| FC5 | IRQ1 Enable Low Bit | IRQ1ENL | 00 | 79 |
| FC6 | Interrupt Request 2 | IRQ2 | 00 | 75 |
| FC7 | IRQ2 Enable High Bit | IRQ2ENH | 00 | 80 |
| 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 |
| **GPIO Port A** | | | | |
| FD0 | Port A Address | PAADDR | 00 | 58 |
| FD1 | Port A Control | PACTL | 00 | 60 |

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). 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.

Table 8. Register File Address Map (Continued)

| Address (Hex) | Register Description | Mnemonic | Reset (Hex)[1] | Page # |
|---|---|---|---|---|
| FD2 | Port A Input Data | PAIN | XX | 60 |
| FD3 | Port A Output Data | PAOUT | 00 | 60 |
| **GPIO Port B** | | | | |
| FD4 | Port B Address | PBADDR | 00 | 58 |
| FD5 | Port B Control | PBCTL | 00 | 60 |
| FD6 | Port B Input Data | PBIN | XX | 60 |
| FD7 | Port B Output Data | PBOUT | 00 | 60 |
| **GPIO Port C** | | | | |
| FD8 | Port C Address | PCADDR | 00 | 58 |
| FD9 | Port C Control | PCCTL | 00 | 60 |
| FDA | Port C Input Data | PCIN | XX | 60 |
| FDB | Port C Output Data | PCOUT | 00 | 60 |
| **GPIO Port D** | | | | |
| FDC | Port D Address | PDADDR | 00 | 58 |
| FDD | Port D Control | PDCTL | 00 | 60 |
| FDE | Port D Input Data | PDIN | XX | 60 |
| FDF | Port D Output Data | PDOUT | 00 | 60 |
| **GPIO Port E** | | | | |
| FE0 | Port E Address | PEADDR | 00 | 58 |
| FE1 | Port E Control | PECTL | 00 | 60 |
| FE2 | Port E Input Data | PEIN | XX | 60 |
| FE3 | Port E Output Data | PEOUT | 00 | 60 |
| FE4–FEF | Reserved | — | XX | |
| **Reset** | | | | |
| FF0 | Reset Status | RSTSTAT | XX | 40 |
| FF1 | Reserved | — | XX | |

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). 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.

**Table 8. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex)[1] | Page # |
|---|---|---|---|---|
| **Watchdog Timer** | | | | |
| FF2 | Watchdog Timer Reload High Byte | WDTH | FF | 143 |
| FF3 | Watchdog Timer Reload Low Byte | WDTL | FF | 143 |
| FF4–FF5 | Reserved | — | XX | |
| **Trim Bit Control** | | | | |
| FF6 | Trim Bit Address | TRMADR | 00 | 281 |
| FF7 | Trim Data | TRMDR | XX | 281 |
| **Flash Memory Controller** | | | | |
| FF8 | Flash Control | FCTL | 00 | 272 |
| | Flash Status | FSTAT | 00 | 272 |
| FF9 | Flash Page Select | FPS | 00 | 273 |
| | Flash Sector Protect | FPROT | 00 | 274 |
| FFA | Flash Programming Frequency High Byte | FFREQH | 00 | 275 |
| FFB | Flash Programming Frequency Low Byte | FFREQL | 00 | 275 |
| **eZ8 CPU** | | | | |
| FFC | Flags | — | XX | refer to the eZ8 CPU Core User Manual (UM0128) |
| FFD | Register Pointer | RP | XX | |
| FFE | Stack Pointer High Byte | SPH | XX | |
| FFF | Stack Pointer Low Byte | SPL | XX | |

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). 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.

# Chapter 5. Reset, Stop Mode Recovery and Low-Voltage Detection

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

- Power-On Reset (POR)
- Voltage Brown-Out (VBO) protection
- Watchdog Timer (WDT) time-out (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 time-out
- 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

## 5.1. Reset Types

The F1680 Series MCU provides various types of Reset operation. Stop Mode Recovery is considered a form of Reset. Table 9 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 9. Reset and Stop Mode Recovery Characteristics and Latency**

| | Reset Characteristics and Latency | | |
|---|---|---|---|
| **Reset Type** | **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 + 50ms Wait time |
| System Reset with Crystal Oscillator Enabled | Reset (as applicable) | Reset | 568–10068 Internal Precision Oscillator Cycles after IPO starts up; see Table 141 on page 280 for a description of the EXTLTMG user option bit. |
| 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 4µs to start up. When the reset type is a System Reset, the F1680 Series MCU 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 Table 141 on page 280 for a description of the EXTLTMG user option bit. When the reset type is a Stop Mode Recovery, the F1680 Series MCU goes to NORMAL Mode immediately after 4 IPO cycles. The total Stop Mode Recovey delay is less than 6µs. 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 can 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.

## 5.2. Reset Sources

Table 10 lists the possible sources of a System Reset.

**Table 10. 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 Brown-Out | Reset delay begins after supply voltage exceeds VBO level |
| | Watchdog Timer time-out when configured for Reset | None |
| | $\overline{\text{RESET}}$ pin assertion | All reset pulses less than three system clocks in width are ignored, see the Electrical Characteristics chapter on page 349. |
| | 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 Brown-Out | Reset delay begins after supply voltage exceeds VBO level |
| | $\overline{\text{RESET}}$ pin assertion | All reset pulses less than the specified analog delay is ignored, see the Electrical Characteristics chapter on page 349. |
| | DBG pin driven Low | None |

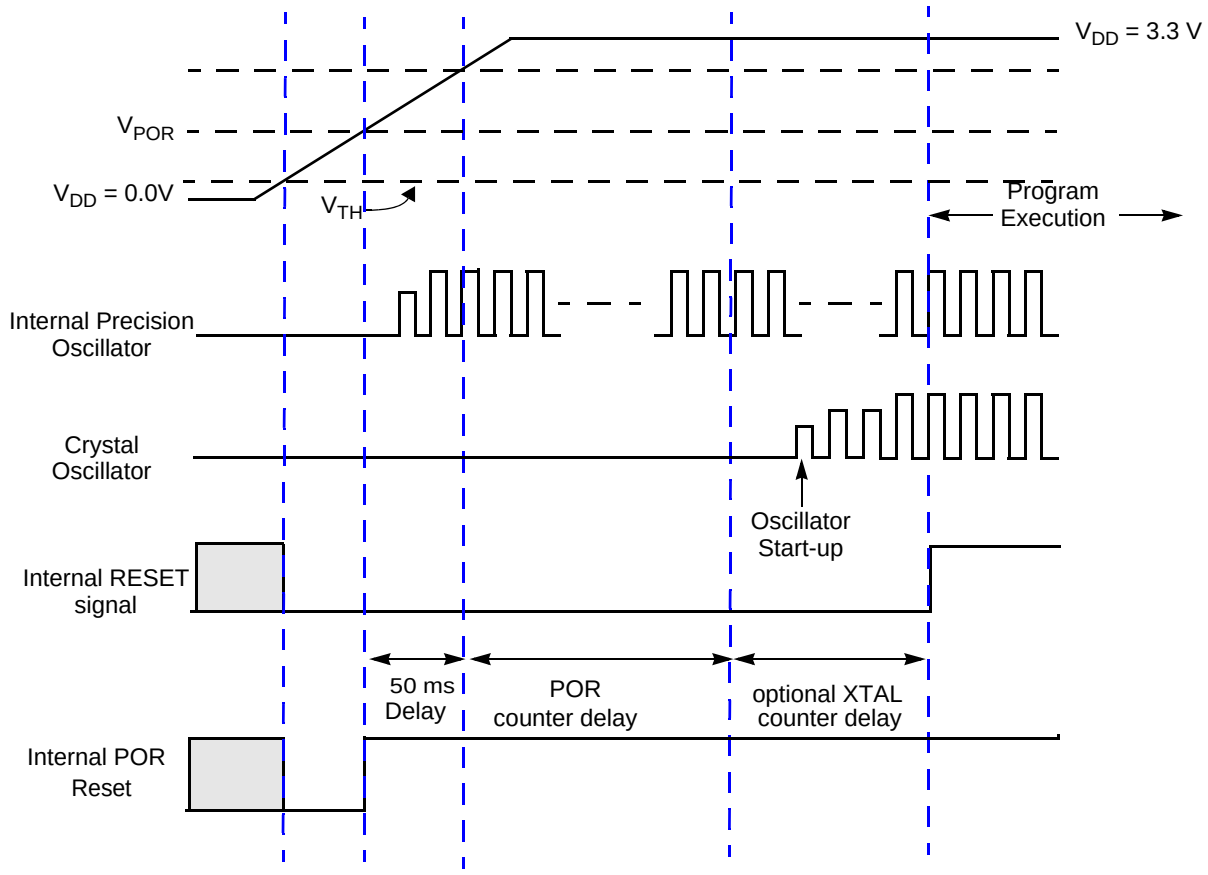
## 5.2.1. Power-On Reset

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 35 displays this POR timing.

After the F1680 Series MCU 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 the Electrical Characteristics chapter on page 349.




**Notes**
1. Not to Scale.
2. Internal Reset and POR Reset are active Low.

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

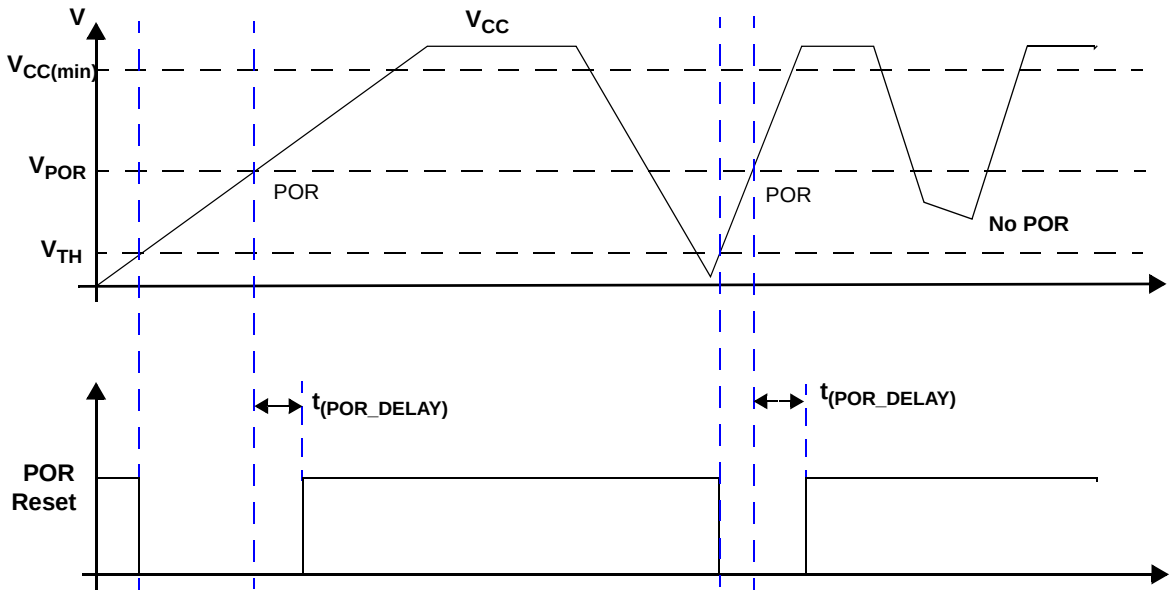


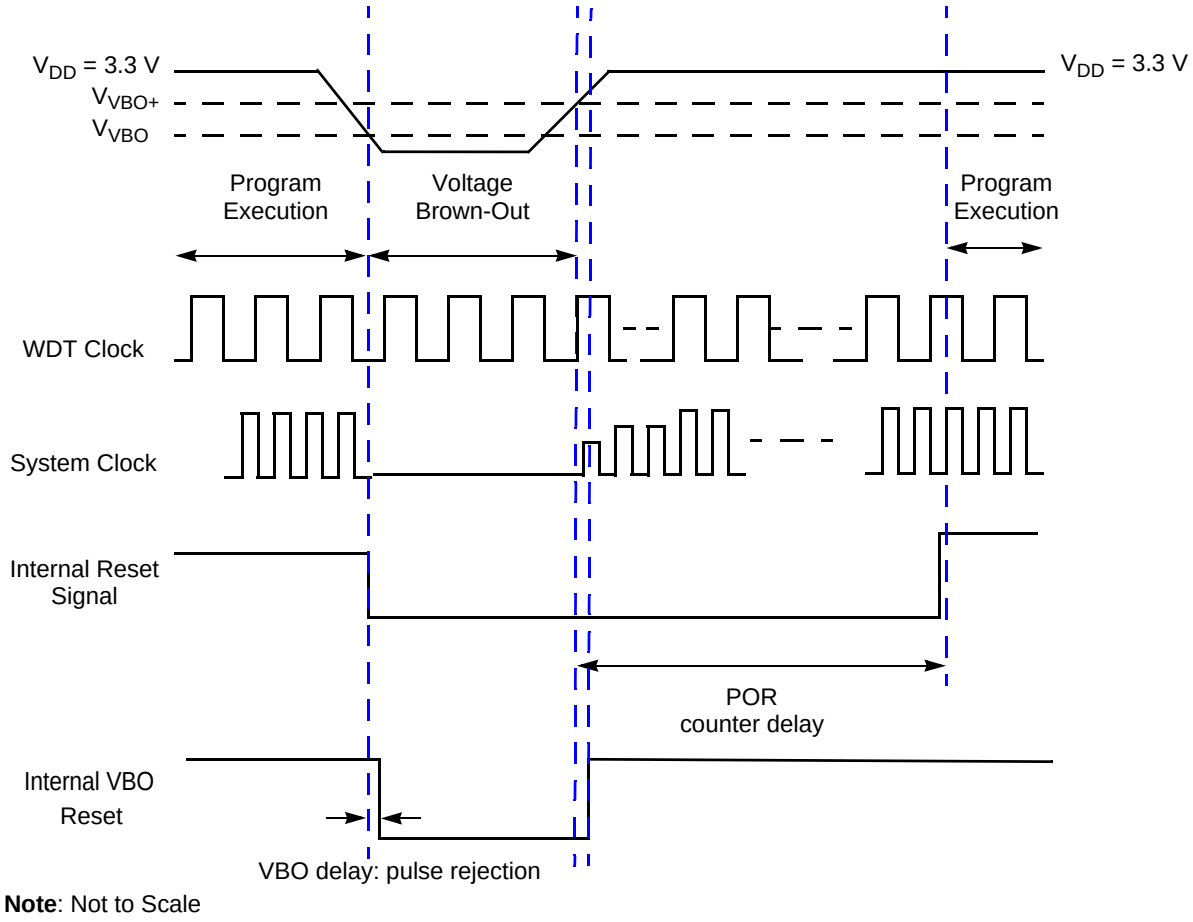

**Figure 7. Power-On Reset Timing**

## 5.2.2. Voltage Brown-Out Reset

The F1680 Series MCU 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 $V_{VBO+}$($V_{VBO}$+$V_{HYS}$) 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 36 displays VBO Reset operation.

For VBO threshold voltages ($V_{VBO}$) and VBO hysteresis ($V_{HYS}$), see the Electrical Characteristics chapter on page 349.

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 the Flash Option Bits chapter on page 276 and the Power Control Register Definitions section on page 44.




**Note**: Not to Scale

**Figure 8. Voltage Brown-Out Reset Operation**

## 5.2.3. Watchdog Timer Reset

If the device is operating in NORMAL or STOP modes, the WDT initiates a System Reset at time-out if the WDT_RES Flash option bit is programmed to 1 (which 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 time-out. The WDT status bit in the Reset Status Register is set to signify that the reset was initiated by the WDT.

### 5.2.4. External Reset Input

The $\overline{\text{RESET}}$ pin has a Schmitt-triggered input and an internal pull-up resistor. When the $\overline{\text{RESET}}$ 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 can 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 F1680 Series MCU remains in the Reset state. If the $\overline{\text{RESET}}$ pin is held Low beyond the System Reset time-out, the device exits the Reset state 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 EXT status bit in the RSTSTAT Register is set to 1.

### 5.2.5. External Reset Indicator

During System Reset or when enabled by the GPIO logic (see the Port A–E Control Registers section on page 60), 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 the F1680 Series MCU 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 9 on page 32 has elapsed.

### 5.2.6. On-Chip Debugger Initiated Reset

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

## 5.3. Stop Mode Recovery

STOP Mode is entered by execution of a stop instruction by the eZ8 CPU. For detailed STOP Mode information, see the Low-Power Modes section on page 42. 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 stop bit in the Reset Status Register is set to 1. Table 11 lists the Stop Mode Recovery sources and resulting actions. The text following provides more detailed information about each of the Stop Mode Recovery sources.

**Table 11. Stop Mode Recovery Sources and Resulting Action**

| Operating Mode | Stop Mode Recovery Source | Action |
|---|---|---|
| STOP Mode | Watchdog Timer time-out when configured for Reset | Stop Mode Recovery |
| | Watchdog Timer time-out 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 $\overline{\text{RESET}}$ Pin | System Reset |
| | Debug Pin driven Low | System Reset |

## 5.3.1. Stop Mode Recovery Using Watchdog Timer Time-Out

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 time-out and the F1680 Series MCU is configured to respond to interrupts. The eZ8 CPU services the WDT interrupt request following the normal Stop Mode Recovery sequence.

## 5.3.2. Stop Mode Recovery Using Timer Interrupt

If a Timer with 32K 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 F1680 Series MCU is configured to respond to interrupts, the

eZ8 CPU services the Timer interrupt request following the normal Stop Mode Recovery sequence.

### 5.3.3. 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 F1680 Series MCU is configured to respond to interrupts, the eZ8 CPU services the comparator interrupt request following the normal Stop Mode Recovery sequence.

### 5.3.4. Stop Mode Recovery Using GPIO Port Pin Transition

Each of the GPIO port pins can be configured as a Stop Mode Recovery input source. On any GPIO pin enabled as a Stop Mode 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 until 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).

### 5.3.5. Stop Mode Recovery Using External $\overline{\text{RESET}}$ Pin

When the F1680 Series MCU is in STOP Mode and the external $\overline{\text{RESET}}$ pin is driven Low, a System Reset occurs. Because of a glitch filter operating on the $\overline{\text{RESET}}$ pin, the Low pulse must be greater than the minimum width specified, or it is ignored. For details, see the Electrical Characteristics chapter on page 349.

## 5.4. 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 about the available Low-Voltage Detection (LVD) threshold levels, see the Trim Option Bits at Address 0000H (TTEMP0) section on page 282.

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 (see the Interrupt Vectors and Priority section on page 71). 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 the Power Control Register Definitions section on page 44.

## 5.5. Reset Register Definitions

The following sections define the Reset registers.

### 5.5.0.1. Reset Status Register

The Reset Status (RSTSTAT) Register, shown in Table 12, is a read-only register that indicates the source of the most recent Reset event, Stop Mode Recovery event and/or WDT time-out. Reading this register resets the upper 4 bits to 0.

This register shares its address with the Reset Status Register, which is write-only.

**Table 12. Reset Status Register (RSTSTAT)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POR/VBO | STOP | WDT | EXT | Reserved | | | LVD |
| Reset | See Table 13 | | | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Address | FF0H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>POR/VBO | **Power-On initiated VBO Reset or general VBO Reset Indicator**<br>If this bit is set to 1, a POR or VBO Reset event occurs. This bit is reset to 0, if a WDT time-out or Stop Mode Recovery occurs. This bit is also reset to 0 when the register is read. |
| [6]<br>STOP | **Stop Mode Recovery Indicator**<br>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 time-out. If the stop bit is 1 and the WDT bit is 0, the Stop Mode Recovery is not caused by a WDT time-out. This bit is reset by Power-On Reset or WDT time-out that occurred while not in STOP Mode. Reading this register also resets this bit. |
| [5]<br>WDT | **Watchdog Timer time-out Indicator**<br>If this bit is set to 1, a WDT time-out 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. |

| Bit | Description (Continued) |
|---|---|
| [4]<br>EXT | **External Reset Indicator**<br>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. |
| [3:1] | Reserved; must be 0. |
| [0]<br>LVD | **Low-Voltage Detection Indicator**<br>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. |

**Table 13. Reset Status Per Event**

| Reset or Stop Mode Recovery Event | POR | STOP | WDT | EXT |
|---|---|---|---|---|
| Power-On Reset or VBO Reset | 1 | 0 | 0 | 0 |
| Reset using $\overline{\text{RESET}}$ pin assertion | 0 | 0 | 0 | 1 |
| Reset using Watchdog Timer time-out | 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 time-out | 0 | 1 | 1 | 0 |

# Chapter 6. Low-Power Modes

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.

## 6.1. 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 32kHz secondary oscillator continues operating
- If enabled for operation in STOP Mode, the Timer logic continues to operate with 32kHz 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 enabled by the Power Control Register
- Low-Power Operational Amplifier and comparator continue to operate if enabled by the Power Control Register
- 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 about Stop Mode Recovery, see the Reset, Stop Mode Recovery and Low-Voltage Detection chapter on page 31.

## 6.2. 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 time-out (Interrupt or Reset)
- Power-On Reset
- Voltage Brown-Out Reset
- External $\overline{\text{RESET}}$ pin assertion

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

## 6.3. 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.

# 6.4. Power Control Register Definitions

The following sections describe the power control registers.

### 6.4.0.1. 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 14. 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 |
| Address | F80H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TRAM | **Low-Power Operational Amplifier Disable**<br>0 = Low-Power Operational Amplifier is enabled (this applies even in STOP Mode).<br>1 = Low-Power Operational Amplifier is disabled. |
| [6:5] | Reserved; must be 0. |
| [4]<br>LVD/VBO | **Low-Voltage Detection/Voltage Brown-Out Detector Disable**<br>0 = LVD/VBO Enabled.<br>1 = LVD/VBO Disabled.<br>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 15 on page 45 lists the setup condition for LVD and VBO circuits in different operation modes. |
| [3]<br>TEMP | **Temperature Sensor Disable**<br>0 = Temperature Sensor Enabled.<br>1 = Temperature Sensor Disabled. |

| Bit | Description |
|---|---|
| [2] | Reserved; must be 0. |
| [1]<br>COMP0 | **Comparator 0 Disable**<br>0 = Comparator 0 is Enabled (this applies even in STOP Mode).<br>1 = Comparator 0 is Disabled. |
| [0]<br>COMP1 | **Comparator 1 Disable**<br>0 = Comparator 1 is Enabled (this applies even in STOP Mode).<br>1 = Comparator 1 is Disabled. |

**Table 15. Setup Condition for LVD and VBO Circuits in Different Operation Modes**

| **LVD/VBO Bit Setup[1]** | | 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[2] | LVD "ON" | | LVD "OFF" | |
| | | VBO "ON" | | VBO "ON" | |
| | VBO_AO="0" or disabled[3] | 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.

# Chapter 7. General-Purpose Input/Output

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.

## 7.1. GPIO Port Availability by Device

Table 16 lists the port pins available with each device and package type.

**Table 16. 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 |

## 7.2. Architecture

Figure 9 displays 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.

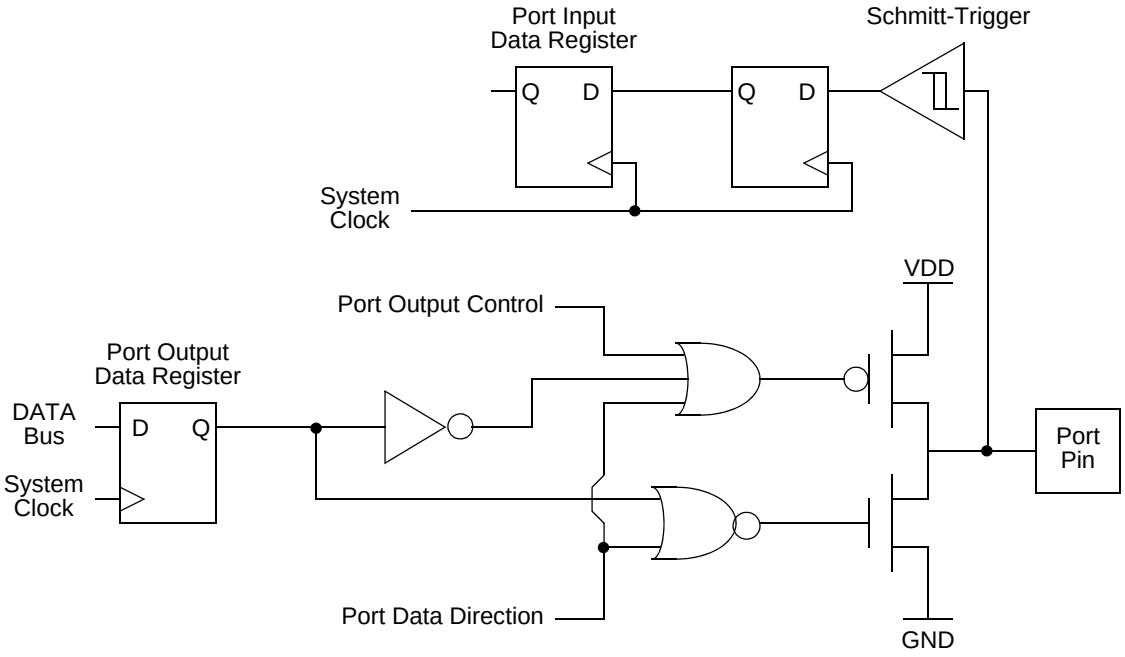



**Figure 9. GPIO Port Pin Block Diagram**

## 7.3. 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 sub-registers 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. Tables 17 through 19 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.

## 7.4. Direct LED Drive

The Port C pins provide a current synchronized output capable of driving an LED without requiring an external resistor. The output synchronizes current at programmable levels of 3mA, 7mA, 13mA and 20mA. This mode is enabled through the Alternate Function sub-register 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 can result in excessive total current. For the maximum total current for the applicable package, see the Electrical Characteristics chapter on page 349.

## 7.5. 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 input/open-drain output reset with an internal pull-up until user software reconfigures it as GPIO PD0. The Port D0 pin is output-only when in GPIO Mode, and must be configured as an output. PD0 supports the High Drive feature but not the Stop Mode Recovery feature.

## 7.6. 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 the Oscillator Control1 Register section on page 320), the GPIO settings are overridden and PA0 and PA1 is disabled.

## 7.7. 32kHz Secondary Oscillator Override

For systems using a 32kHz secondary oscillator, PA2 and PA3 is used to connect a watch crystal. When the 32kHz secondary oscillator is enabled (see the Oscillator Control1 Register section on page 320), the GPIO settings are overridden and PA2 and PA3 is disabled.

## 7.8. 5V Tolerance

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

## 7.9. 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 to the Oscillator Control Register (see page 320) such that the external oscillator is selected as the system clock.

**Table 17. Port Alternate Function Mapping, 20-Pin Parts[1,2]**

| Port | Pin | Mnemonic | Alternate Function Description | Alternate Function Set Register AFS1 |
|---|---|---|---|---|
| **Port A** | PA0 | T0IN/$\overline{\text{T0OUT}}$ | 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 | $\overline{\text{CTS0}}$ | 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/$\overline{\text{T2OUT}}$ | 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/$\overline{\text{T1OUT}}$ | Timer 1 Input/Timer 1 Output Complement | AFS1[6]: 0 |
| | | SCL | $I^2C$ Serial Clock | AFS1[6]: 1 |
| | PA7 | T1OUT | Timer 1 Output | AFS1[7]: 0 |
| | | SDA | $I^2C$ Serial Data | AFS1[7]: 1 |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Port A, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D pin, the Alternate Function Set registers are not implemented for Port D. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.

**Table 17. Port Alternate Function Mapping, 20-Pin Parts[1,2] (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 |
| **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 | Reserved | | AFS1[2]: 0 |
| | | VREF/ANA6/LED | Voltage Reference or ADC Analog Input or LED Drive | AFS1[2]: 1 |
| | PC3 | C0OUT | Comparator 0 Output | AFS1[3]: 0 |
| | | LED | LED drive | AFS1[3]: 1 |
| **Port D** | PD0 | $\overline{\text{RESET}}$ | External Reset | N/A |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Port A, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D pin, the Alternate Function Set registers are not implemented for Port D. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.

**Table 18. Port Alternate Function Mapping, 28-Pin Parts[1,2]**

| Port | Pin | Mnemonic | Alternate Function Description | Alternate Function Set Register AFS1 |
|---|---|---|---|---|
| **Port A** | PA0 | T0IN/$\overline{\text{T0OUT}}$ | 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 | $\overline{\text{CTS0}}$ | 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/$\overline{\text{T1OUT}}$ | Timer 1 Input/Timer 1 Output Complement | AFS1[6]: 0 |
| | | SCL | $I^2C$ Serial Clock | AFS1[6]: 1 |
| | PA7 | T1OUT | Timer 1 Output | AFS1[7]: 0 |
| | | SDA | $I^2C$ Serial Data | AFS1[7]: 1 |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A and B, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.

**Table 18. Port Alternate Function Mapping, 28-Pin Parts[1,2] (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 |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A and B, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.

**Table 18. Port Alternate Function Mapping, 28-Pin Parts[1,2] (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 | 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 | C0OUT | 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 |
| **Port D** | PD0 | $\overline{RESET}$ | External Reset | N/A |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A and B, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.

**Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts[1,2]**

| Port | Pin | Mnemonic | Alternate Function Description | Alternate Function Set Register AFS1 |
|---|---|---|---|---|
| **Port A** | PA0 | T0IN/$\overline{\text{T0OUT}}$ | 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 | $\overline{\text{CTS0}}$ | 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/$\overline{\text{T1OUT}}$ | 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 |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A–C, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.
3. This timer function is only available in the 44-pin package; its alternate functions are reserved in the 40-pin package.

**Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts[1,2] (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 |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A–C, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.
3. This timer function is only available in the 44-pin package; its alternate functions are reserved in the 40-pin package.

**Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts[1,2] (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 |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A–C, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.
3. This timer function is only available in the 44-pin package; its alternate functions are reserved in the 40-pin package.

**Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts[1,2] (Continued)**

| Port | Pin | Mnemonic | Alternate Function Description | Alternate Function Set Register AFS1 |
|---|---|---|---|---|
| **Port D** | PD0 | $\overline{\text{RESET}}$ | External Reset | N/A |
| | PD1 | C1INN | Comparator 1 Input (N) | |
| | PD2 | C1INP | Comparator 1 Input (P) | |
| | PD3 | $\overline{\text{CTS1}}$/C1OUT | UART 1 Clear to Send or Comparator 1 Output | |
| | PD4 | RXD1/IRRX1 | UART 1/IrDA 1 Receive Data | |
| | PD5 | TXD1/IRTX1 | UART 1/IrDA 1 Transmit Data | |
| | PD6 | DE1 | UART 1 Driver Enable | |
| | PD7 | C0OUT | Comparator 0 Output | |
| **Port E** | PE0 | T4IN[3] | | N/A |
| | | Reserved | | |
| | PE1 | SCL | $I^2C$ Serial Clock | |
| | | Reserved | | |
| | PE2 | SDA | $I^2C$ Serial Data | |
| | | Reserved | | |
| | PE3 | T4CHA[3] | | |
| | | Reserved | | |
| | PE4 | T4CHB[3] | | |
| | | Reserved | | |
| | PE5 | T4CHC[3] | | |
| | | Reserved | | |
| | PE6 | T4CHD[3] | | |
| | | Reserved | | |

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A–C, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.
3. This timer function is only available in the 44-pin package; its alternate functions are reserved in the 40-pin package.

## 7.10. 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 about interrupts using the GPIO pins, see the Interrupt Controller chapter on page 68.

## 7.11. GPIO Control Register Definitions

Four registers for each port provide access to GPIO control, input data and output data. Table 20 lists these port registers. Use Port A–E Address and Control registers together to provide access to subregisters for port configuration and control.

**Table 20. GPIO Port Registers and Subregisters**

| Port Register Mnemonic | Port Register Name |
|---|---|
| P*x*ADDR | Port A–E Address Register (Selects subregisters) |
| P*x*CTL | Port A–E Control Register (Provides access to subregisters) |
| P*x*IN | Port A–E Input Data Register |
| P*x*OUT | Port A–E Output Data Register |
| **Port Subregister Mnemonic** | **Port Register Name** |
| P*x*DD | Data Direction |
| P*x*AF | Alternate Function |
| P*x*OC | Output Control (Open-Drain) |
| P*x*HDE | High Drive Enable |
| P*x*SMRE | Stop Mode Recovery Source Enable |
| P*x*PUE | Pull-up Enable |
| PxAFS1 | Alternate Function Set 1 |
| PxAFS2 | Alternate Function Set 2 |

## 7.11.1. 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 21.

**Table 21. 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 |
| Address | FD0H, FD4H, FD8H, FDCH, FE0H | | | | | | | |

| Bit | Description | |
|---|---|---|
| [7:0] PADDR | **Port Address**<br>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. |

## 7.11.2. 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 22.

**Table 22. 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 |
| Address | FD1H, FD5H, FD9H, FDDH, FE1H | | | | | | | |

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

## 7.11.3. 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, as indicated in Table 23.

**Table 23. 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 |
| Address | If 01H in Port A–E Address Register, accessible through the Port A–E Control Register. | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>DD | **Data Direction**<br>These bits control the direction of the associated port pin. Port Alternate Function operation overrides the Data Direction register setting.<br>0 = Output. Data in the Port A–E Output Data Register is driven onto the port pin.<br>1 = Input. The port pin is sampled and the value written into the Port A–E Input Data Register. The output driver is tristated. |

## 7.11.4. Port A–E Alternate Function Subregisters

The Port A–E Alternate Function Subregister, shown in Table 24, 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 the pins; if disabled, the pins function as GPIO. If enabled, select one of the four alternate functions using Alternate Function Set Subregisters 1 and 2 as described in the Port A–E Alternate Function Set 1 Subregisters section on page 64 and the Port A–E Alternate Function Set 2 Subregisters section on page 64. To determine the alternate function associated with each port pin, see the GPIO Alternate Functions section on page 47.

**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 24. 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 | | | | | | | |
| Address | If 02H in Port A–D Address Register, accessible through the Port A–E Control Register. | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>AF | **Port Alternate Function enabled**<br>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.<br>1 = The alternate function selected through Alternate Function set subregisters are enabled. Port-pin operation is controlled by the alternate function. |

## 7.11.5. Port A–E Output Control Subregisters

The Port A–E Output Control Subregister, shown in Table 25, 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 25. 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 |
| Address | If 03H in Port A–E Address Register, accessible through the Port A–E Control Register | | | | | | | |

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

## 7.11.6. Port A–E High Drive Enable Subregisters

The Port A–E High Drive Enable Subregister, shown in Table 26, 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 26. 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 |
| Address | If 04H in Port A–E Address Register, accessible through the Port A–E Control Register | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PHDE | **Port High Drive Enabled**<br>0 = The Port pin is configured for standard output current drive.<br>1 = The Port pin is configured for high output current drive. |

## 7.11.7. Port A–E Stop Mode Recovery Source Enable Subregisters

The Port A–E Stop Mode Recovery Source Enable Subregister, shown in Table 27, is accessed through the 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 Stop Mode Recovery sources. During STOP Mode, any logic transition on a port pin enabled as a Stop Mode Recovery source initiates Stop Mode Recovery.

**Table 27. 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 |
| Address | If 05H in Port A–E Address Register, accessible through the Port A–E Control Register. | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PSMRE | **Port Stop Mode Recovery Source Enabled**<br>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.<br>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. |

## 7.11.8. Port A–E Pull-up Enable Subregisters

The Port A–E Pull-up Enable Subregister, shown in Table 28, 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 28. 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 |
| Address | If 06H in Port A–E Address Register, accessible through the Port A–E Control Register. | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PPUE | **Port Pull-up Enabled**<br>0 = The weak pull-up on the Port pin is disabled.<br>1 = The weak pull-up on the Port pin is enabled. |

## 7.11.9. Port A–E Alternate Function Set 1 Subregisters

The Port A–E Alternate Function Set1 Subregister, shown in Table 29, 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 the GPIO Alternate Functions section on page 47.

**Note:** Alternate function selection on the port pins must also be enabled as described in the Port A–E Alternate Function Subregisters section on page 61.

**Table 29. 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 |
| Address | If 07H in Port A–E Address Register, accessible through the Port A–E Control Register | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PAFS1 | **Port Alternate Function Set 1**<br>0 = Port Alternate Function selected, as defined in the GPIO Alternate Functions section on page 47.<br>1 = Port Alternate Function selected, as defined in the GPIO Alternate Functions section on page 47. |

## 7.11.10. Port A–E Alternate Function Set 2 Subregisters

The Port A–E Alternate Function Set 2 Subregister, shown in Table 30, 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 select the alternate function available at a port pin. Alternate Functions selected by setting or clearing bits of this register are defined in the GPIO Alternate Functions section on page 47.

**Note:** Alternate function selection on port pins must also be enabled as described in the Port A–E Alternate Function Subregisters section on page 61.

**Table 30. 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 |
| Address | If 08H in Port A–E Address Register, accessible through the Port A–E Control Register | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PAFS2 | **Port Alternate Function Set 2**<br>0 = Port Alternate Function selected as defined in the GPIO Alternate Functions section on page 47.<br>1 = Port Alternate Function selected as defined the GPIO Alternate Functions section on page 47. |

## 7.11.11. Port A–E Input Data Registers

Reading from the Port A–E Input Data registers, shown in Table 31, 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 31. 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 |
| Address | FD2H, FD6H, FDAH, FDEH, FE2H | | | | | | | |

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

## 7.11.12. Port A–E Output Data Register

The Port A–E Output Data Register, shown in Table 32, controls the output data to the pins.

**Table 32. Port A–E Output Data Register (P*x*OUT)**

| 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 |
| Address | FD3H, FD7H, FDBH, FDFH, FE3H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] POUT | **Port Output Data**<br>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.<br>0 = Drive a logical 0 (Low).<br>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. |

## 7.11.13. LED Drive Enable Register

The LED Drive Enable Register, shown in Table 33, 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 33. 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 |
| Address | F82H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] LEDEN | **LED Drive Enable**<br>These bits determine which Port C pins are connected to an internal current sink.<br>0 = Tristate the Port C pin.<br>1 = Connect controlled current synch to Port C pin. |

## 7.11.14. LED Drive Level Registers

Two LED Drive Level registers consist of the LED Drive Level High Bit Register (LEDLVLH[7:0]) and the LED Drive Level Low Bit Register (LEDLVLL[7:0]), as shown in Tables 34 and 35. Two control bits, LEDLVLH[x] and LEDLVLL[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 34. LED Drive Level High Bit Register (LEDLVLH)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | LEDLVLH | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | F83H | | | | | | | |

**Table 35. LED Drive Level Low Bit Register (LEDLVLL)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | LEDLVLL | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | F84H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>LEDLVLH,<br>LEDLVLL | **LED Drive Level High Bit**<br>**LED Drive Level Low Bit**<br>These bits are used to set the LED drive current. {LEDLVLH[x], LEDLVLL[x]}, in which x=Port C[0] to Port C[7]. Select one of the following four programmable current drive levels for each Port C pin.<br>00 = 3 mA<br>01 = 7 mA<br>10 = 13 mA<br>11 = 20 mA |

# Chapter 8. Interrupt Controller

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 36)
  - 15 on-chip peripheral interrupt sources (three interrupt vectors are shared; see Table 36)
- 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 about interrupt servicing by the eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), available on www.zilog.com.

## 8.1. Interrupt Vector Listing

Table 36 lists all of the interrupts available in order of priority. The interrupt vector is stored with the most-significant byte (MSB) at the even Program Memory address and the least-significant byte (LSB) at the following odd Program Memory address.

**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 36. 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 the Watchdog Timer chapter on page 140) |
| | 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 | $I^2C$ |
| | 0014H | SPI |
| | 0016H | ADC |
| | 0018H | Port A7, selectable rising or falling input edge or LVD (see the Reset, Stop Mode Recovery and Low-Voltage Detection chapter on page 31) |
| | 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 meaningful when considering identical interrupt levels. This priority varies depending on different interrupt level settings. See the Interrupt Vectors and Priority section on page 71 for details.

## 8.2. Architecture

Figure 10 displays the interrupt controller block diagram.

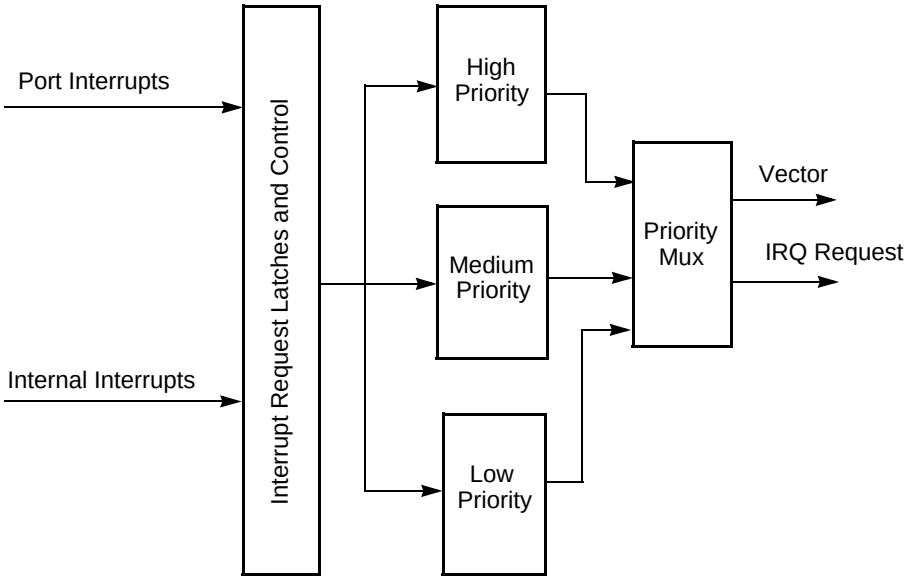



**Figure 10. Interrupt Controller Block Diagram**

## 8.3. Operation

This section describes the operational aspects of the following functions.

Master Interrupt Enable: see page 70

Interrupt Vectors and Priority: see page 71

Interrupt Assertion: see page 71

Software Interrupt Assertion: see page 72

### 8.3.1. 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

## 8.3.2. 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 36 on page 69. 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 36. 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.

## 8.3.3. Interrupt Assertion

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

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

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

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

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

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

```
ANDX IRQ0, MASK
```

### 8.3.4. Software Interrupt Assertion

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

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

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

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

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

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

```
ORX IRQ0, MASK
```

## 8.4. 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.

## 8.4.1. Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) Register, shown in Table 37, 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 37. Interrupt Request 0 Register (IRQ0)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T2I | T1I | T0I | U0RXI | U0TXI | I2CI | 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 |
| Address | FC0H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>T2I | **Timer 2 Interrupt Request**<br>0 = No interrupt request is pending for Timer 2.<br>1 = An interrupt request from Timer 2 is awaiting service. |
| [6]<br>T1I | **Timer 1 Interrupt Request**<br>0 = No interrupt request is pending for Timer 1.<br>1 = An interrupt request from Timer 1 is awaiting service. |
| [5]<br>T0I | **Timer 0 Interrupt Request**<br>0 = No interrupt request is pending for Timer 0.<br>1 = An interrupt request from Timer 0 is awaiting service. |
| [4]<br>U0RXI | **UART 0 Receiver Interrupt Request**<br>0 = No interrupt request is pending for the UART 0 receiver.<br>1 = An interrupt request from the UART 0 receiver is awaiting service. |
| [3]<br>U0TXI | **UART 0 Transmitter Interrupt Request**<br>0 = No interrupt request is pending for the UART 0 transmitter.<br>1 = An interrupt request from the UART 0 transmitter is awaiting service. |
| [2]<br>I2CI | **$I^2C$ Interrupt Request**<br>0 = No interrupt request is pending for the $I^2C$.<br>1 = An interrupt request from $I^2C$ is awaiting service. |
| [1]<br>SPII | **SPI Interrupt Request**<br>0 = No interrupt request is pending for the SPI.<br>1 = An interrupt request from the SPI is awaiting service. |
| [0]<br>ADCI | **ADC Interrupt Request**<br>0 = No interrupt request is pending for the ADC.<br>1 = An interrupt request from the ADC is awaiting service. |

## 8.4.2. Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) Register, shown in Table 38, 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 a 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 38. Interrupt Request 1 Register (IRQ1)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PA7VI | PA6CI | PA5CI | PAD4I | PAD3I | PAD2I | PAD1I | PA0I |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | FC3H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>PA7VI | **Port A7 or LVD Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port A7 or LVD.<br>1 = An interrupt request from GPIO Port A7 or LVD. |
| [6]<br>PA6CI | **Port A6 or Comparator 0 Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port A6 or Comparator 0.<br>1 = An interrupt request from GPIO Port A6 or Comparator 0. |
| [5]<br>PA5CI | **Port A5 or Comparator 1 Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port A5 or Comparator 1.<br>1 = An interrupt request from GPIO Port A5 or Comparator 1. |
| [4:1]<br>PADxI | **Port A or Port D Pin *x* Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port A or Port D pin *x*.<br>1 = An interrupt request from GPIO Port A or Port D pin *x* is awaiting service; *x* indicates the specific GPIO port pin number (1–4). |
| [0]<br>PA0I | **Port A Pin 0 Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port A0.<br>1 = An interrupt request from GPIO Port A0 is awaiting service.<br>For interrupt source select description, see the Shared Interrupt Select Register section on page 82. |

## 8.4.3. Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register, shown in Table 39, 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 39. 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 |
| Address | FC6H | | | | | | | |

| Bit | Description |
|---|---|
| [7] | Reserved; must be 0. |
| [6]<br>MCTI | **Multi-channel timer Interrupt Request**<br>0 = No interrupt request is pending for multi-channel timer.<br>1 = An interrupt request from multi-channel timer is awaiting service. |
| [5]<br>U1RXI | **UART 1 Receiver Interrupt Request**<br>0 = No interrupt request is pending for the UART 1 receiver.<br>1 = An interrupt request from the UART 1 receiver is awaiting service. |
| [4]<br>U1TXI | **UART 1 Transmitter Interrupt Request**<br>0 = No interrupt request is pending for the UART 1 transmitter.<br>1 = An interrupt request from the UART 1 transmitter is awaiting service. |
| [3:0]<br>PC*x*I | **Port C Pin *x* Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port C pin *x*.<br>1 = An interrupt request from GPIO Port C pin *x* is awaiting service; *x* indicates the specific GPIO Port C pin number (0–3). |

## 8.4.4. IRQ0 Enable High and Low Bit Registers

Table 40 describes the priority control for IRQ0. The IRQ0 Enable High and Low Bit registers, shown in Tables 41 and 42, form a priority-encoded enabling for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register.

**Table 40. 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 41. IRQ0 Enable High Bit Register (IRQ0ENH)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T2ENH | T1ENH | T0ENH | U0RENH | U0TENH | I2CENH | 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 |
| Address | FC1H | | | | | | | |

| Bit | Description |
|---|---|
| [7] T2ENH | Timer 2 Interrupt Request Enable High Bit |
| [6] T1ENH | Timer 1 Interrupt Request Enable High Bit |
| [5] T0ENH | Timer 0 Interrupt Request Enable High Bit |
| [4] U0RENH | UART 0 Receive Interrupt Request Enable High Bit |
| [3] U0TENH | UART 0 Transmit Interrupt Request Enable High Bit |
| [2] I2CENH | $I^2C$ Interrupt Request Enable High Bit |
| [1] SPIENH | SPI Interrupt Request Enable High Bit |
| [0] ADCENH | ADC Interrupt Request Enable High Bit |

**Table 42. 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/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | FC2H | | | | | | | |

| Bit | Description |
|---|---|
| [7] T2ENL | Timer 2 Interrupt Request Enable Low Bit |
| [6] T1ENL | Timer 1 Interrupt Request Enable Low Bit |
| [5] T0ENL | Timer 0 Interrupt Request Enable Low Bit |
| [4] U0RENL | UART 0 Receive Interrupt Request Enable Low Bit |
| [3] U0TENL | UART 0 Transmit Interrupt Request Enable Low Bit |
| [2] I2CENL | $I^2C$ Interrupt Request Enable Low Bit |
| [1] SPIENL | SPI Interrupt Request Enable Low Bit |
| [0] ADCENL | ADC Interrupt Request Enable Low Bit |

## 8.4.5. IRQ1 Enable High and Low Bit Registers

Table 43 lists the priority control for IRQ1. The IRQ1 Enable High and Low Bit registers, shown in Tables 44 and 45) form a priority-encoded enabling for interrupts in the Interrupt Request 1 Register. Priority is generated by setting bits in each register.

**Table 43. 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: An *x* indicates the register bits from 0–7.

**Table 44. 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 |
| Address | FC4H | | | | | | | |

| Bit | Description |
|---|---|
| [7] PA7VENH | Port A Bit[7] or LVD Interrupt Request Enable High Bit. |
| [6] PA6C0ENH | Port A Bit[6] or Comparator 0 Interrupt Request Enable High Bit. |
| [5] PA5C1ENH | Port A Bit[5] or Comparator 1 Interrupt Request Enable High Bit. |
| [4:1] PAD*x*ENH | Port A or Port D Bit[*x*] (*x*=1, 2, 3, 4) Interrupt Request Enable High Bit. |
| [0] PA0ENH | Port A Bit[0] Interrupt Request Enable High Bit. See the Shared Interrupt Select Register (IRQSS) on page 82 to determine a selection of either Port A or Port D as the interrupt source. |

**Table 45. 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 |
| Address | FC5H | | | | | | | |

| Bit | Description |
|---|---|
| [7] PA7VENL | Port A Bit[7] or LVD Interrupt Request Enable Low Bit. |
| [6] PA6C0ENL | Port A Bit[6] or Comparator 0 Interrupt Request Enable Low Bit. |
| [5] PA5C1ENL | Port A Bit[5] or Comparator 1 Interrupt Request Enable Low Bit. |
| [4:1] PAD*x*ENL | Port A or Port D Bit[*x*] (*x*=1, 2, 3, 4) Interrupt Request Enable Low Bit. |
| [0] PA0ENL | Port A Bit[0] Interrupt Request Enable Low Bit. |

## 8.4.6. IRQ2 Enable High and Low Bit Registers

Table 46 describes the priority control for IRQ2. The IRQ2 Enable High and Low Bit registers, shown in Tables 47 and 48 form a priority-encoded enabling for interrupts in the Interrupt Request 2 Register. Priority is generated by setting bits in each register.

**Table 46. 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: An *x* indicates the register bits from 0–7.

**Table 47. 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 |
| Address | FC7H | | | | | | | |

| Bit | Description |
|---|---|
| [7] | Reserved; must be 0. |
| [6] MCTENH | Multi-Channel Timer Interrupt Request Enable High Bit |
| [5] U1RENH | UART1 Receive Interrupt Request Enable High Bit |
| [4] U1TENH | UART1 Transmit Interrupt Request Enable High Bit |
| [3] C3ENH | Port C3 Interrupt Request Enable High Bit |
| [2] C2ENH | Port C2 Interrupt Request Enable High Bit |
| [1] C1ENH | Port C1 Interrupt Request Enable High Bit |
| [0] C0ENH | Port C0 Interrupt Request Enable High Bit |

**Table 48. 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 |
| Address | FC8H | | | | | | | |

| Bit | Description |
|---|---|
| [7] | Reserved; must be 0. |
| [6] MCTENL | Multi-Channel Timer Interrupt Request Enable Low Bit (MCTENL) |
| [5] U1RENL | UART1 Receive Interrupt Request Enable Low Bit (U1RENL) |
| [4] U1TENL | UART1 Transmit Interrupt Request Enable Low Bit (U1TENL) |
| [3] C3ENL | Port C3 Interrupt Request Enable Low Bit (C3ENL) |
| [2] C2ENL | Port C2 Interrupt Request Enable Low Bit (C2ENL) |
| [1] C1ENL | Port C1 Interrupt Request Enable Low Bit (C1ENL) |
| [0] C0ENL | Port C0 Interrupt Request Enable Low Bit (C0ENL) |

## 8.4.7. Interrupt Edge Select Register

The Interrupt Edge Select (IRQES) Register, shown in Table 49, 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 49. 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 |
| Address | FCDH | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>IES*x* | **Interrupt Edge Select *x***<br>0 = An interrupt request is generated on the falling edge of the PA*x* input or PD*x* input.<br>1 = An interrupt request is generated on the rising edge of the PA*x* input or PD*x* input; *x* indicates the specific GPIO port pin number (0–7). |

## 8.4.8. Shared Interrupt Select Register

The Shared Interrupt Select (IRQSS) Register, shown in Table 50, determines the source of the PAD*x*S interrupts. The Shared Interrupt Select Register selects between Port A and alternate sources for the individual interrupts.

**Table 50. 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 |
| Address | FCEH | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>PA7VS | **PA7/LVD Selection**<br>0 = PA7 is used for the interrupt for PA7VS interrupt request.<br>1 = The LVD is used for the interrupt for PA7VS interrupt request. |
| [6]<br>PA6CS | **PA6/Comparator 0 Selection**<br>0 = PA6 is used for the interrupt for PA6CS interrupt request.<br>1 = The Comparator 0 is used for the interrupt for PA6CS interrupt request. |

| Bit | Description |
|---|---|
| [5]<br>PA5CS | **PA5/Comparator 1 Selection**<br>0 = PA5 is used for the interrupt for PA5CS interrupt request.<br>1 = The Comparator 1 is used for the interrupt for PA5CS interrupt request. |
| [4:1]<br>PAD*x*S | **PAx/PDx Selection**<br>0 = PAx is used for the interrupt for PAx/PDx interrupt request<br>1 = PDx is used for the interrupt for PAx/PDx interrupt request; an *x* indicates the specific GPIO port pin number (1–4). |
| [0] | Reserved; must be 0. |

## 8.4.9. Interrupt Control Register

The Interrupt Control (IRQCTL) Register, shown in Table 51, contains the master enable bit for all interrupts.

**Table 51. 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 |
| Address | FCFH | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>IRQE | **Interrupt Request Enable**<br>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, a Reset, or by a direct register write of a 0 to this bit.<br>0 = Interrupts are disabled.<br>1 = Interrupts are enabled. |
| [6:0] | Reserved; must be 0. |

# Chapter 9. Timers

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 32kHz secondary oscillator

In addition to the timers described in this chapter, the Baud Rate Generator (BRG) of unused UART peripheral can also be used to provide basic timing functionality. For more information about using the Baud Rate Generator as additional timers, see the LIN-UART chapter on page 144.

## 9.1. Architecture

Figure 11 displays the architecture of the timers.

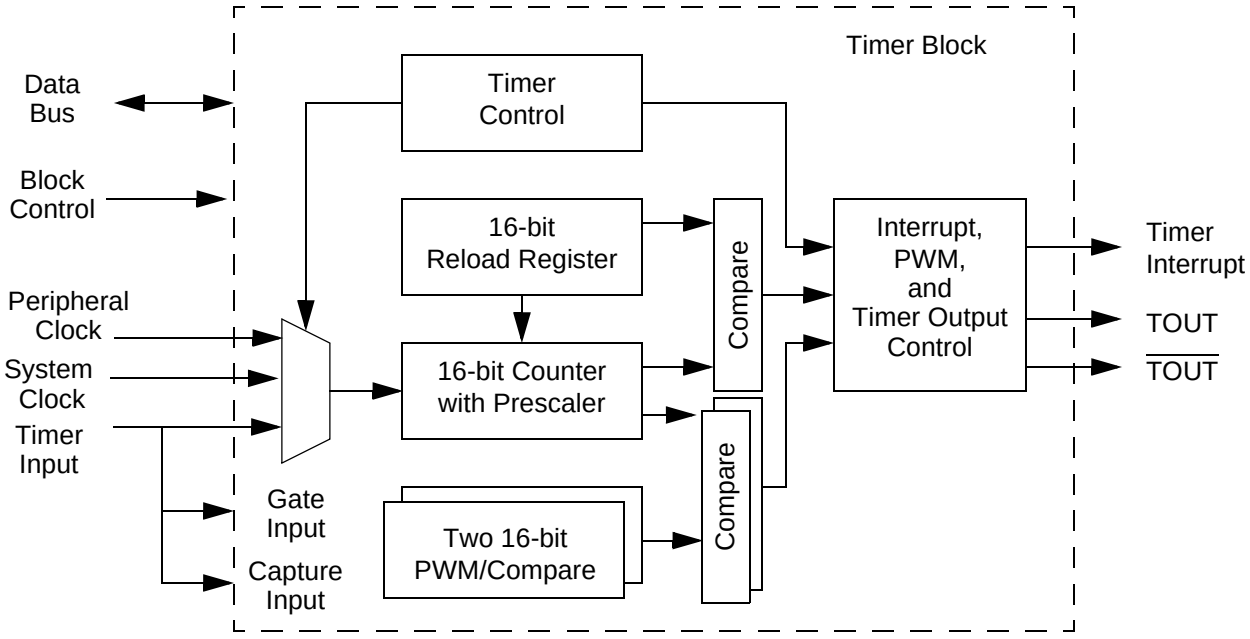



**Figure 11. Timer Block Diagram**

## 9.2. Operation

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

### 9.2.1. 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 the 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 (also include changing the timer clock source). Therefore, any write to the timer control registers cannot be performed when the timer is enabled and a peripheral clock is used.

When the timer uses a peripheral clock and the timer is enabled, any read from TxH or TxL is not recommended, because the results can be unpredictable. Disable the timer first, then read it. If the timer works in the CAPTURE, CAPTURE/COMPARE, CAPTURE RESTART or DEMODULATION modes, any read from TxPWM0H, TxPWM0L, TxPWM1H, TxPWM1L or TxSTAT must be performed after a capture interrupt occurs; otherwise, results can be unpredictable. The INPCAP bit of the Timer Control 0 Register is the same as these PWM registers. When the timer uses the main clock, you can write/read all timer registers at any time.

## 9.2.2. Low-Power Modes

Timers can operate in both HALT Mode and STOP Mode.

### 9.2.2.1. 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.

### 9.2.2.2. 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 after Stop Mode Recovery occurs.

#### 9.2.2.3. 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.

### 9.2.3. Timer Operating Modes

The timers can be configured to operate in the following modes, each of which is described in this section where indicated in Table 52.

**Table 52. Timer Operating Modes**

| Mode | Page Number |
|---|---|
| TRIGGERED ONE-SHOT Mode | 88 |
| CONTINUOUS Mode | 90 |
| COUNTER Mode | 91 |
| COMPARATOR COUNTER Mode | 92 |
| PWM SINGLE OUTPUT Mode | 93 |
| PWM DUAL Output Mode | 95 |
| CAPTURE Mode | 97 |
| CAPTURE RESTART Mode | 98 |
| COMPARE Mode | 100 |
| GATED Mode | 100 |
| CAPTURE/COMPARE Mode | 102 |
| DEMODULATION Mode | 103 |

#### 9.2.3.1. 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. Upon reaching the reload value, the timer generates an interrupt, and the count value in the Timer High and Low Byte registers is reset to `0001H`. Then, the timer is automatically disabled and stops counting.

Additionally, 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) upon timer reload. If it is appropriate to have the Timer Output make a permanent state change on

One-Shot time-out. 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.

Observe the following steps to configure a timer for ONE-SHOT Mode and to initiate 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 calculated using the following equation:

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

### 9.2.3.2. TRIGGERED ONE-SHOT Mode

In TRIGGERED ONE-SHOT Mode, the timer operates in the following sequence:

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. Upon 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 shown in the following equation:

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

Table 53 provides an example initialization sequence for configuring Timer 0 in TRIGGERED ONE-SHOT Mode and initiating operation.

**Table 53. TRIGGERED ONE-SHOT Mode Initialization Example**

| Register | Value | Comment |
|---|---|---|
| T0CTL0 | E0H | TMODE[3:0] = 1011B selects TRIGGERED ONE-SHOT Mode.<br>TICONFIG[1:0] = 11B enables interrupts on Timer reload only.<br>CSC = 0 selects the Timer Input (Trigger) from the GPIO pin.<br>PWMD[2:0] = 000B has no effect.<br>INPCAP = 0 has no effect.<br>TEN = 0 disables the timer.<br>TPOL = 0 enables triggering on rising edge of Timer. Input and sets Timer Out signal to 0.<br>PRES[2:0] = 000B sets prescaler to divide by 1.<br>TCLKS = 1 sets 32kHz peripheral clock as the Timer clock source. |
| T0CTL1 | 03H | |
| T0CTL2 | 01H | |
| T0H | 00H | Timer starting value = 0001H. |
| T0L | 01H | |
| T0RH | ABH | Timer reload value = ABCDH. |
| T0RL | CDH | |
| PAADDR | 02H | Selects Port A Alternate Function control register. |
| PACTL[1:0] | 11B | PACTL[0] enables Timer 0 Input Alternate function.<br>PACTL[1] enables Timer 0 Output Alternate function. |
| IRQ0ENH[5] | 0B | Disables the Timer 0 interrupt. |
| IRQ0ENL[5] | 0B | |

**Table 53. TRIGGERED ONE-SHOT Mode Initialization Example (Continued)**

| Register | Value | Comment |
|---|---|---|
| T0CTL1 | 83H | TEN = 1 enables the timer. All other bits remain in their appropriate 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.

### 9.2.3.3. 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. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001H` and counting resumes. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or High to Low) on timer reload.

Observe the following steps to configure a timer for CONTINUOUS Mode and initiate 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 value only affects the first pass in CONTINUOUS Mode. After the first timer reload in CONTINUOUS Mode, counting always begins at the reset value of `0001H`.
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. If appropriate, 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 calculated using 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 time-out period.

### 9.2.3.4. 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.

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or High to Low) at timer reload.

Observe the following steps to configure a timer for COUNTER Mode and initiate 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 is not required 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 value 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 calculated using the following equation:

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

### 9.2.3.5. COMPARATOR COUNTER Mode

In COMPARATOR COUNTER Mode, the timer counts output transitions from an analog comparator output. The assignment of a comparator to a timer is based on the TIMTRG bits in the CMP0 and CMP1 registers. 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.

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or High to Low) at timer reload.

Observe the following steps to configure a timer for COMPARATOR COUNTER Mode and initiate 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 appropriate comparator control register (COMP0 or COMP1) to set the TIMTRG bits that map the comparator to the timer.
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 High and Low Byte registers to set the starting count value. This value 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.
6. Write to the Timer Reload High and Low Byte registers to set the reload value.
7. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
8. If 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 COMPARATOR COUNTER Mode, the number of comparator output transitions since the timer start is calculated using the following equation:

Comparator Output Transitions = Current Count Value - Start Value

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

If the TPOL bit in the Timer Control 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.

Observe the following steps to configure a timer for PWM SINGLE OUTPUT Mode and initiate 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 value 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 appropriate, 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 calculated using the following equation:

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 time-out period.

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

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

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

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

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

### 9.2.3.7. 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 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 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. 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 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 set a time delay (0 to 128 timer clock cycles) when one PWM output transitions from High to Low and the other PWM output

transitions from a Low to High. This configuration ensures a time gap between the removal of one PWM output and the assertion of its complement.

Observe the following steps to configure a timer for PWM DUAL OUTPUT Mode and initiate 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 the 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 value 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 appropriate, 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 calculated using the following equation:

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 time-out period.

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

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

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

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

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

### 9.2.3.8. CAPTURE Mode

In CAPTURE Mode, the current timer count value is recorded when the appropriate external Timer Input transition occurs. The Capture count value is written to the Timer 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. Upon 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.

Observe the following steps to configure a timer for CAPTURE Mode and initiate 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:

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

### 9.2.3.9. CAPTURE RESTART Mode

In CAPTURE RESTART Mode, the current timer count value is recorded when the appropriate external Timer Input transition occurs. The Capture count value is written to the Timer PWM High and Low Byte registers. The Timer 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. 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 INPCAP bit in Timer Control 0 Register is cleared to indicate the timer interrupt is not due to an input capture event.

Observe the following steps to configure a timer for CAPTURE RESTART Mode and initiate 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 the 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:

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

### 9.2.3.10. 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 a 16-bit reload value. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0001H). Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) on Compare.

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

Observe the following steps to configure a timer for COMPARE Mode and initiate 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 appropriate, 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 calculated using the following equation:

### 9.2.3.11. 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

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

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.

Observe the following steps to configure a timer for GATED Mode and initiate 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 value 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.

### 9.2.3.12. CAPTURE/COMPARE Mode

In CAPTURE/COMPARE Mode, the timer begins counting on the first external Timer Input transition. The appropriate 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 appropriate 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. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. The INPCAP bit in Timer Control 0 Register is cleared to indicate the timer interrupt is not due to an input capture event.

Observe the following steps to configure a timer for CAPTURE/COMPARE Mode and initiate 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:

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

### 9.2.3.13. DEMODULATION Mode

In DEMODULATION Mode, the timer begins counting on the first external Timer Input transition. The appropriate 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 appropriate 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, a Capture is executed 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 in the Timer Status Register, PWMxEF, is set to indicate that 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 the Timer Status Register is set to indicate that the timer interrupt is due to a Reload event. Software can use this bit to determine if a Reload occurred prior to a Capture.

Observe the following steps to configure a timer for DEMODULATION Mode and initiate 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 the TMODEHI bit in the TxCTL0 Register
- Set the prescale value
- Set the TPOL bit to set the Capture edge (rising or falling) for the Timer Input. This setting applies only if the TPOLHI bit in the 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 54 provides an example initialization sequence for configuring Timer 0 in DEMODULATION Mode and initiating operation.

**Table 54. DEMODULATION Mode Initialization Example**

| Register | Value | Comment |
|---|---|---|
| T0CTL0 | C0H | TMODE[3:0] = 1100B selects DEMODULATION Mode.<br>TICONFIG[1:0] = 10B enables interrupt only on Capture events.<br>CSC = 0 selects the Timer Input from the GPIO pin.<br>PWMD[2:0] = 000B has no effect.<br>INPCAP = 0 has no effect.<br>TEN = 0 disables the timer.<br>PRES[2:0] = 000B sets prescaler to divide by 1.<br>TPOLHI,TPOL = 10 enables trigger and Capture on both rising and falling edges of Timer Input.<br>TCLKS = 1 enables 32kHz peripheral clock as timer clock source |
| T0CTL1 | 04H | |
| T0CTL2 | 11H | |
| T0H | 00H | Timer starting value = 0001H. |
| T0L | 01H | |
| T0RH | ABH | Timer reload value = ABCDH |
| T0RL | CDH | |
| T0PWM0H | 00H | Initial PWM0 value = 0000H |
| T0PWM0L | 00H | |
| T0PWM1H | 00H | Initial PWM1 value = 0000H |
| T0PWM1H | 00H | |
| T0NFC | C0H | NFEN = 1 enables noise filter<br>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.<br>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 remain in their appropriate settings. |
| Notes: | | |

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

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

## 9.2.4. 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.

## 9.2.5. Timer Output Signal Operation

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

## 9.2.6. 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

## 9.2.7. Architecture

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

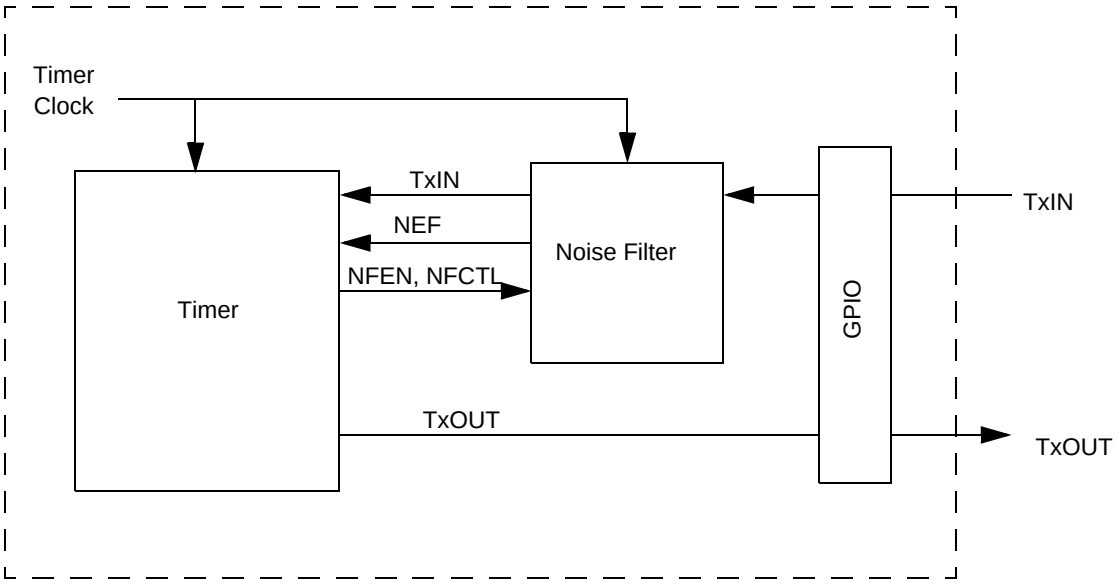



**Figure 12. Noise Filter System Block Diagram**

### 9.2.7.1. Operation

Figure 13 displays 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 the center of the bit time) does not mean that the sampled data is incorrect; rather, it is intended to be an indicator of the level of noise in the network.

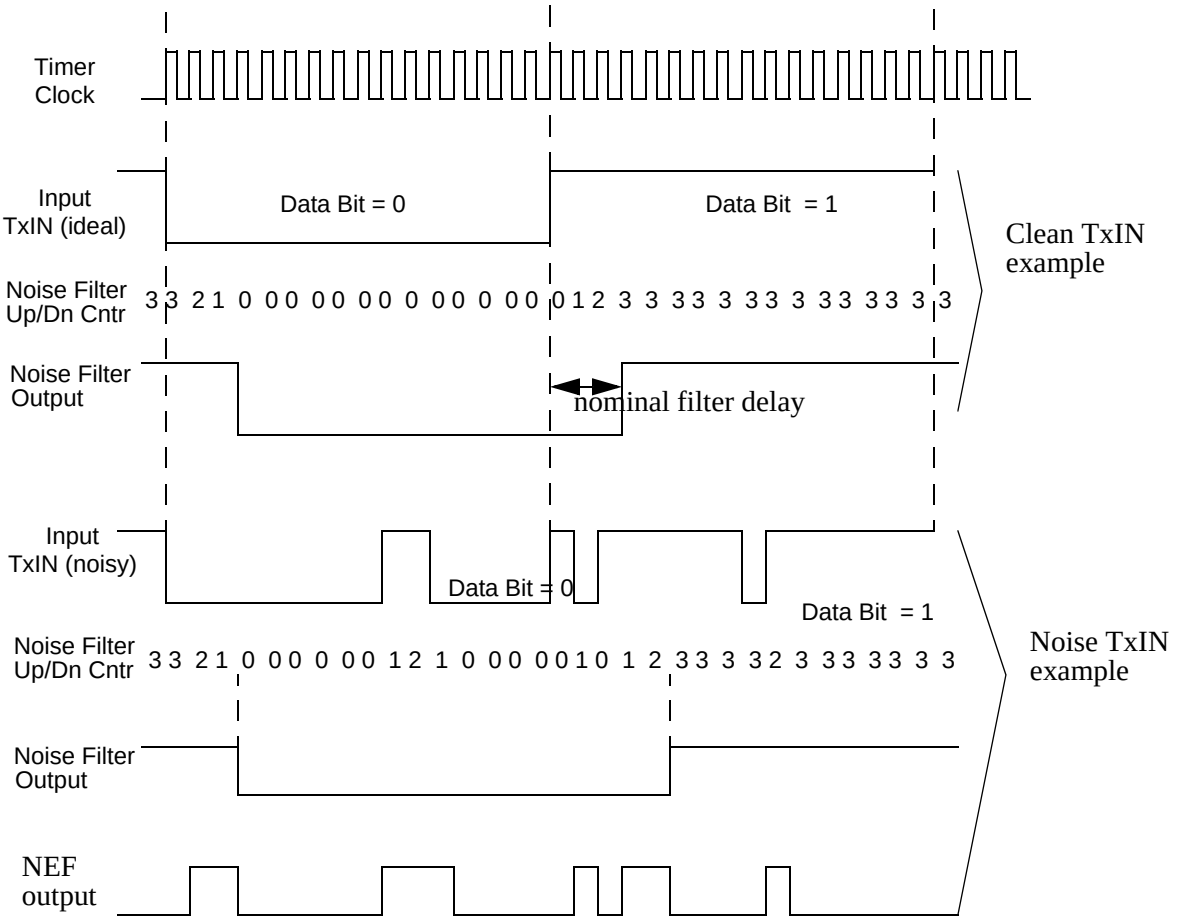



**Figure 13. Noise Filter Operation**

## 9.3. Timer Control Register Definitions

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

Timer 0–2 High and Low Byte Registers: see page 109

Timer Reload High and Low Byte Registers: see page 109

Timer 0–2 PWM0 High and Low Byte Registers: see page 110

Timer 0–2 PWM1 High and Low Byte Registers: see page 111

Timer 0–2 Control Registers: see page 112

Timer 0–2 Status Registers: see page 118

Timer 0–2 Noise Filter Control Register: see page 119

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

The Timer 0–2 High and Low Byte (TxH and TxL) registers, shown in Tables 55 and 56, 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; therefore 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 55. Timer 0–2 High Byte Register (TxH)**

| Bit | 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 |
| Address | F00H, F08H, F10H | | | | | | | |

**Table 56. Timer 0–2 Low Byte Register (TxL)**

| Bit | 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 |
| Address | F01H, F09H, F11H | | | | | | | |

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

## 9.3.2. Timer Reload High and Low Byte Registers

The Timer 0–2 Reload High and Low Byte (TxRH and TxRL) registers, shown in Tables 57 and 58, 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, this 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 57. Timer 0–2 Reload High Byte Register (TxRH)**

| Bit | 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 |
| Address | F02H, F0AH, F12H | | | | | | | |

**Table 58. Timer 0–2 Reload Low Byte Register (TxRL)**

| Bit | 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 |
| Address | F03H, F0BH, F13H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] TRH, TRL | **Timer Reload Register High and Low**<br>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. |

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

The Timer 0–2 PWM0 High and Low Byte (TxPWM0H and TxPWM0L) registers, shown in Tables 59 and 60, 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` occurs – that is, unless PWM0UE = 1.

**Table 59. Timer 0–2 PWM0 High Byte Register (TxPWM0H)**

| Bit | 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 |
| Address | F04H, F0CH, F14H | | | | | | | |

**Table 60. Timer 0–2 PWM0 Low Byte Register (TxPWM0L)**

| Bit | 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 |
| Address | F05H, F0DH, F15H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PWM0H, PWM0L | **Pulse Width Modulator 0 High and Low Bytes**<br>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).<br>The TxPWM0H and TxPWM0L registers also store the 16-bit captured timer value when operating in CAPTURE, CAPTURE/COMPARE and DEMODULATION Modes. |

## 9.3.4. Timer 0–2 PWM1 High and Low Byte Registers

The Timer 0–2 PWM1 High and Low Byte (TxPWM1H and TxPWM1L) registers, shown in Tables 61 and 62, store Capture values for DEMODULATION Mode.

**Table 61. Timer 0-2 PWM1 High Byte Register (TxPWM1H)**

| Bit | 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 |
| Address | F20H, F24H, F28H | | | | | | | |

**Table 62. Timer 0–2 PWM1 Low Byte Register (TxPWM1L)**

| Bit | 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 |
| Address | F21H, F25H, F29H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PWM1H, PWM1L | **Pulse Width Modulator 1 High and Low Bytes**<br>These two bytes, {PWM1H[7:0], PWM1L[7:0]}, store the 16-bit captured timer value for DEMODULATION Mode. |

## 9.3.5. Timer 0–2 Control Registers

The Timer Control registers are described in Tables 63 through 65.

### 9.3.5.1. Timer 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 63. Timer 0–2 Control 0 Register (TxCTL0)**

| Bit | 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 |
| Address | F06H, F0EH, F16H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TMODE[3] | **Timer Mode High Bit**<br>This bit, along with the TMODE[2:0] field in the TxCTL1 Register, determines the operating mode of the timer. This bit is the most significant bit of the timer mode selection value. For more details, see the description of the Timer 0–2 Control 1 Register (TxCTL1) on page 113. |
| [6:5]<br>TICONFIG | **Timer Interrupt Configuration**<br>This field configures timer interrupt definition.<br>0x = Timer Interrupt occurs on all defined Reload, Compare and Input Events.<br>10 = Timer Interrupt only on defined Input Capture/Deassertion Events.<br>11 = Timer Interrupt only on defined Reload/Compare Events. |
| [4]<br>CSC | **Cascade Timers**<br>0 = Timer Input signal comes from the pin.<br>1 = For Timer 0, Input signal is connected to Timer 2 output.<br>For Timer 1, Input signal is connected to Timer 0 output.<br>For Timer 2, Input signal is connected to Timer 1 output. |

| Bit | Description (Continued) |
|---|---|
| [3:1]<br>PWMD | **PWM Delay Value**<br>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.<br>000 = No delay<br>001 = 2 cycles delay<br>010 = 4 cycles delay<br>011 = 8 cycles delay<br>100 = 16 cycles delay<br>101 = 32 cycles delay<br>110 = 64 cycles delay<br>111 = 128 cycles delay |
| [0]<br>INPCAP | Input Capture Event<br>This bit indicates if the last timer interrupt is due to a Timer Input Capture Event.<br>0 = Previous timer interrupt is not a result of Timer Input Capture Event.<br>1 = Previous timer interrupt is a result of Timer Input Capture Event. |

### 9.3.5.2. Timer 0–2 Control 1 Register

The Timer 0–2 Control 1 (TxCTL1) registers enable and disable the timers, set the prescaler value and determine the timer operating mode. See Table 64.

**Table 64. Timer 0–2 Control 1 Register (TxCTL1)**

| Bit | 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 |
| Address | F07H, F0FH, F17H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TEN | **Timer Enable**<br>0 = Timer is disabled.<br>1 = Timer enabled to count. |

| Bit | Description (Continued) |
|---|---|
| [6]<br>TPOL | **Timer Input/Output Polarity**<br>Operation of this field is a function of the current operating modes of the timer.<br>**ONE-SHOT Mode**<br>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.<br>**CONTINUOUS Mode**<br>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.<br>**COUNTER Mode**<br>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.<br>0 = Count occurs on the rising edge of the Timer Input signal.<br>1 = Count occurs on the falling edge of the Timer Input signal.<br>**PWM SINGLE OUTPUT Mode**<br>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.<br>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.<br>**CAPTURE Mode**<br>0 = Count is captured on the rising edge of the Timer Input signal.<br>1 = Count is captured on the falling edge of the Timer Input signal.<br>**COMPARE Mode**<br>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.<br>**GATED Mode**<br>0 = Timer counts when the Timer Input signal is High (1) and interrupts are generated on the falling edge of the Timer Input.<br>1 = Timer counts when the Timer Input signal is Low (0) and interrupts are generated on the rising edge of the Timer Input.<br>**CAPTURE/COMPARE Mode**<br>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.<br>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. |

| Bit | Description (Continued) |
|---|---|
| [6] (cont'd) | **PWM DUAL OUTPUT Mode**<br>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).<br>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).<br>**CAPTURE RESTART Mode**<br>0 = Count is captured on the rising edge of the Timer Input signal.<br>1 = Count is captured on the falling edge of the Timer Input signal.<br>**COMPARATOR COUNTER Mode**<br>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.<br>**TRIGGERED ONE-SHOT Mode**<br>0 = Timer counting is triggered on the rising edge of the Timer Input signal.<br>1 = Timer counting is triggered on the falling edge of the Timer Input signal.<br>**DEMODULATION Mode**<br>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.<br>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.<br>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. |

| Bit | Description (Continued) |
|---|---|
| [5:3]<br>PRES | **Prescale Value**<br>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.<br>000 = Divide by 1<br>001 = Divide by 2<br>010 = Divide by 4<br>011 = Divide by 8<br>100 = Divide by 16<br>101 = Divide by 32<br>110 = Divide by 64<br>111 = Divide by 128 |
| [2:0]<br>TMODE[2:0] | **Timer Mode**<br>This field, along with the TMODE[3] bit in the TxCTL0 Register, determines the operating mode of the timer. TMODE[3:0] selects among the following modes:<br>0000 = ONE-SHOT Mode<br>0001 = CONTINUOUS Mode<br>0010 = COUNTER Mode<br>0011 = PWM SINGLE OUTPUT Mode<br>0100 = CAPTURE Mode<br>0101 = COMPARE Mode<br>0110 = GATED Mode<br>0111 = CAPTURE/COMPARE Mode<br>1000 = PWM DUAL OUTPUT Mode<br>1001 = CAPTURE RESTART Mode<br>1010 = COMPARATOR COUNTER Mode<br>1011 = TRIGGERED ONE-SHOT Mode<br>1100 = DEMODULATION Mode |

### 9.3.5.3. 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. See Table 65.

**Table 65. Timer 0–2 Control 2 Register (TxCTL2)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | PWM0UE | TPOLHI | Reserved | | | TCLKS* |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | F22H, F26H, F2AH | | | | | | | |

| Bit | Description |
|---|---|
| [7:6] | Reserved; must be 0. |
| [5]<br>PWM0UE | **PWM0 Update Enable**<br>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.<br>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.<br>1 = Writes to the Channel High and Low Byte registers are not buffered when TEN = 1. |
| [4]<br>TPOLHI | **Timer Input/Output Polarity High Bit**<br>This bit determines if timer count is triggered and captured on both edges of the input signal. This applies only to DEMODULATION Mode.<br>0 = Count is captured only on one edge in DEMODULATION Mode. In this case, edge polarity is determined by TPOL bit in the TxCTL1 Register.<br>1 = Count is triggered on any edge and captured on both rising and falling edges of the Timer Input signal in DEMODULATION Mode. |
| [3:1] | Reserved; must be 0. |
| [0]<br>TCLKS | **Timer Clock Source**<br>0 = System Clock.<br>1 = Peripheral Clock.* |

Note: *Before selecting the peripheral clock as the timer clock source, the peripheral clock must be enabled and oscillating.

## 9.3.6. Timer 0–2 Status Registers

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

**Table 66. Timer 0–2 Status Register (TxSTAT)**

| Bit | 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 |
| Address | F23H, F27H, F2BH | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>NEF | **Noise Event Flag**<br>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 can 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. |
| [6] | Reserved; must be 0. |
| [5:4]<br>PWMxEO | **PWM x Event Overrun**<br>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.<br>0 = No Overrun<br>1 = Capture/Compare Event Flag Overrun |
| [3]<br>RTOEF | **Reload Time-Out Event Flag**<br>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.<br>0 = No Reload Time-Out event occurred<br>1 = A Reload Time-Out event occurred |
| [2] | Reserved; must be 0. |
| [1:0]<br>PWMxEF | **PWM x Event Flag**<br>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.<br>0 = No Capture/Compare Event occurred for this PWM channel<br>1 = A Capture/Compare Event occurred for this PWM channel |

## 9.3.7. Timer 0–2 Noise Filter Control Register

The Timer 0–2 Noise Filter Control Register (TxNFC) enables and disables the Timer Noise Filter and sets the noise filter control.

**Table 67. Timer 0–2 Noise Filter Control Register (TxNFC)**

| Bit | 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 | | | |
| Address | F2CH, F2DH, F2EH | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>NFEN | **Noise Filter Enable**<br>0 = Noise Filter is disabled.<br>1 = Noise Filter is enabled. Receive data is preprocessed by the noise filter. |
| [6:4]<br>NFCTL | **Noise Filter Control**<br>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.<br>000 = 2-bit up/down counter<br>001 = 3-bit up/down counter<br>010 = 4-bit up/down counter<br>011 = 5-bit up/down counter<br>100 = 6-bit up/down counter<br>101 = 7-bit up/down counter<br>110 = 8-bit up/down counter<br>111 = 9-bit up/down counter |
| [3:0] | Reserved; must be 0. |

# Chapter 10. Multi-Channel Timer

The Multi-Channel timer, offered on all 44-pin F1680 Series parts, features 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 also includes the following features:

- 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

## 10.1. Architecture

Figure 14 displays the Multi-Channel Timer architecture.

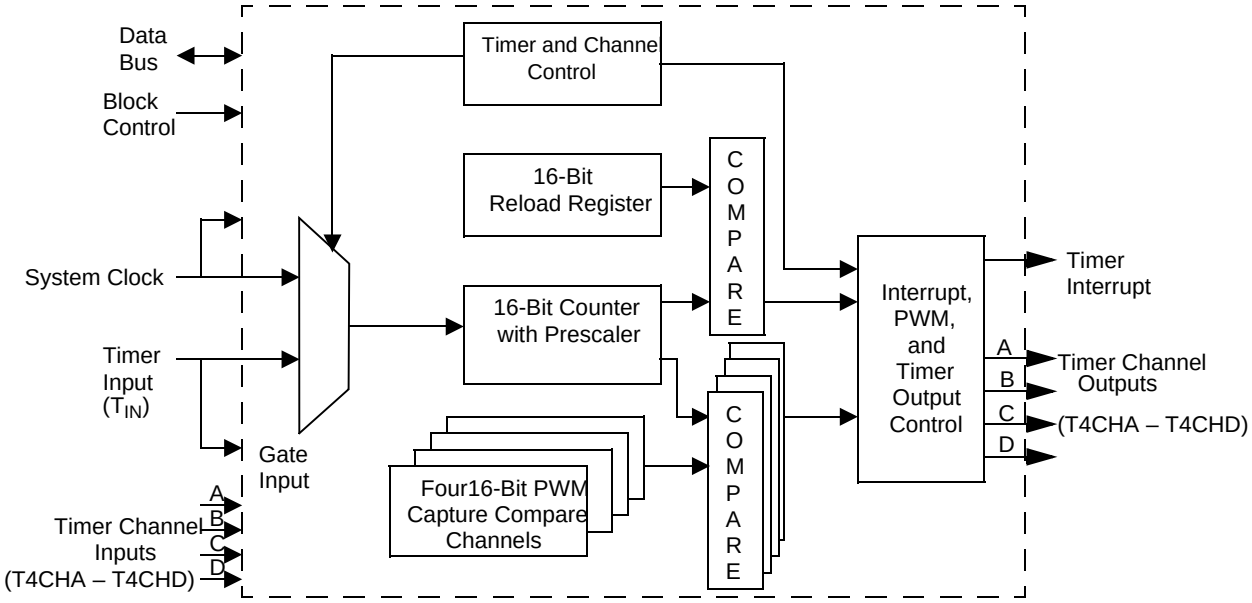



**Figure 14. Multi-Channel Timer Block Diagram**

## 10.2. Timer Operation

This section discusses the key features of the Multi-Channel Timer, including its counter, clock source, prescaler and counting modes.

### 10.2.1. 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.

### 10.2.2. Clock Source

The Multi-Channel Timer clock source can come from either the system clock or the alternate function $T_{IN}$ pin when the system clock is the clock source; the alternate function $T_{IN}$ input pin can perform a clock gating function. The TCLKS field in the MCTCTL0 Register selects the timer clock source. When using the $T_{IN}$ pin, the associated GPIO pin, T4CH, must be configured as an input. The $T_{IN}$ frequency cannot exceed one-fourth the system clock frequency.

## 10.2.3. 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 Multi-Channel Timer end-of-cycle count. The prescaler has no effect when the $T_{IN}$ is selected as the clock source.

## 10.2.4. Multi-Channel Timer Start

The Multi-Channel Timer starts counting when the TEN bit in the 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.

## 10.2.5. 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 the MCTCTL1 Register. The timer modes are described below in Table 68.

**Table 68. 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 | |

## 10.2.6. 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 shown 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.

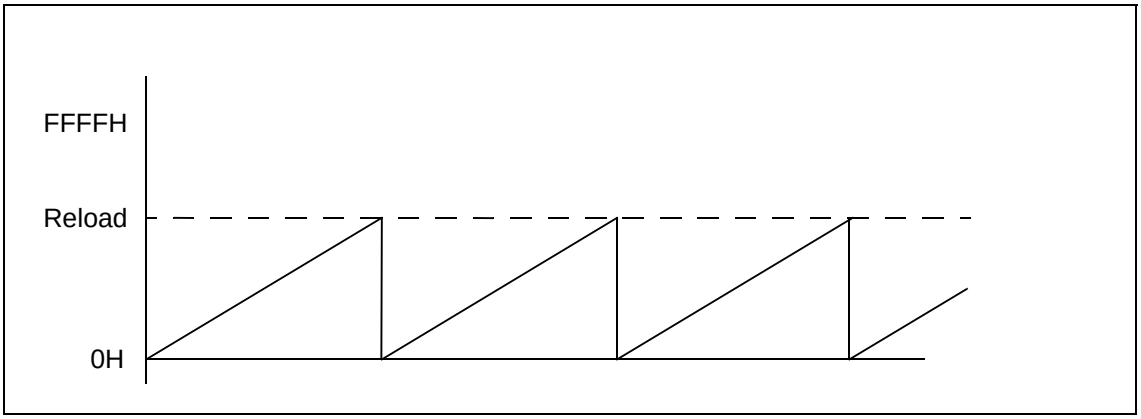



**Figure 15. Count Modulo Mode**

## 10.2.7. 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 shown in Figures 16, 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.

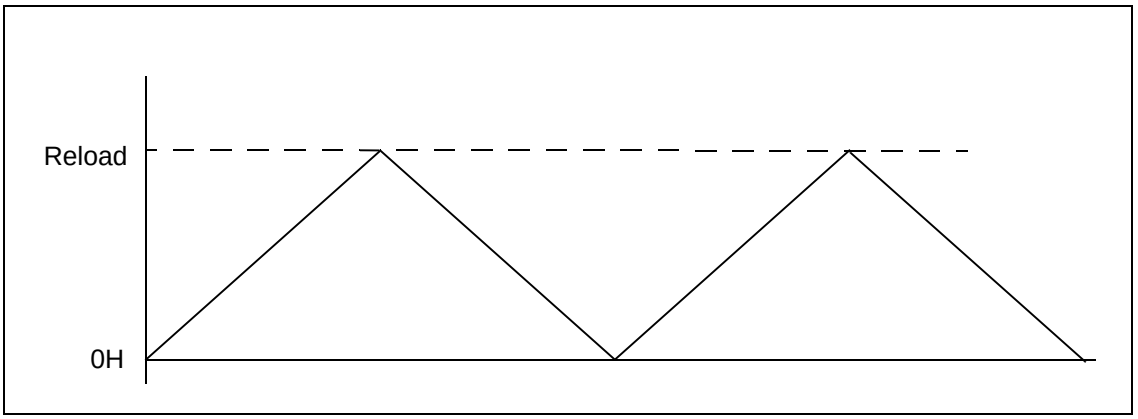



**Figure 16. Count Up/Down Mode**

# 10.3. 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 (T4CHA, T4CHB, T4CHC, or T4CHD) 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 ONE-SHOT COMPARE, CONTINUOUS COMPARE, PWM OUTPUT, or INPUT CAPTURE mode.

## 10.3.1. One-Shot Compare Operation

In a 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 Status 1 Register (MCTCHS1) to identify the responsible channel. The channel is then automatically disabled. The timer continues counting according to the programmed mode. If the timer channel output alternate function is enabled, the channel output pin (T4CHA, T4CHB, T4CHC, or T4CHD) changes state for one system clock cycle upon match (i.e., from Low to High, then back to Low or High to Low, then back to High as determined by the CHPOL bit).

## 10.3.2. Continuous Compare Operation

In a 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 Status 1 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 (T4CHA, T4CHB, T4CHC, or T4CHD) changes state upon match (i.e., from Low to High then back to Low; or High to Low then back to High, as determined by the CHPOL bit).

### 10.3.3. PWM Output Operation

In a PWM OUTPUT operation, the timer generates a PWM output signal on the channel output pin (T4CHA, T4CHB, T4CHC, or T4CHD). 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).

### 10.3.4. Capture Operation

In a CAPTURE operation, the current timer count is recorded when the selected transition occurs on T4CHA, T4CHB, T4CHC or T4CHD. 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.

## 10.4. 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 (T4CHA, T4CHB, T4CHC, T4CHD).

### 10.4.1. Timer Interrupt

If enabled by the 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.

### 10.4.2. 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.

# 10.5. Low-Power Modes

The Z8 Encore! XP F1680 Series of MCUs contains power-saving features. The highest level of power reduction is provided by STOP Mode. The next level of power reduction is provided by HALT Mode.

## 10.5.1. 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.

## 10.5.2. Operation in STOP Mode

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

## 10.5.3. 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 and write to the registers when the enable bit(s) are deasserted.

# 10.6. Multi-Channel Timer Applications Examples

This section provides two brief examples that describe how the the F1680 Series multi-channel timer can be used in your application.

## 10.6.1. PWM Programmable Deadband Generation

The count up/down mode supports motor control applications that require dead time between output signals. Figure 17 displays dead time generation between two channels operating in count up/down mode.

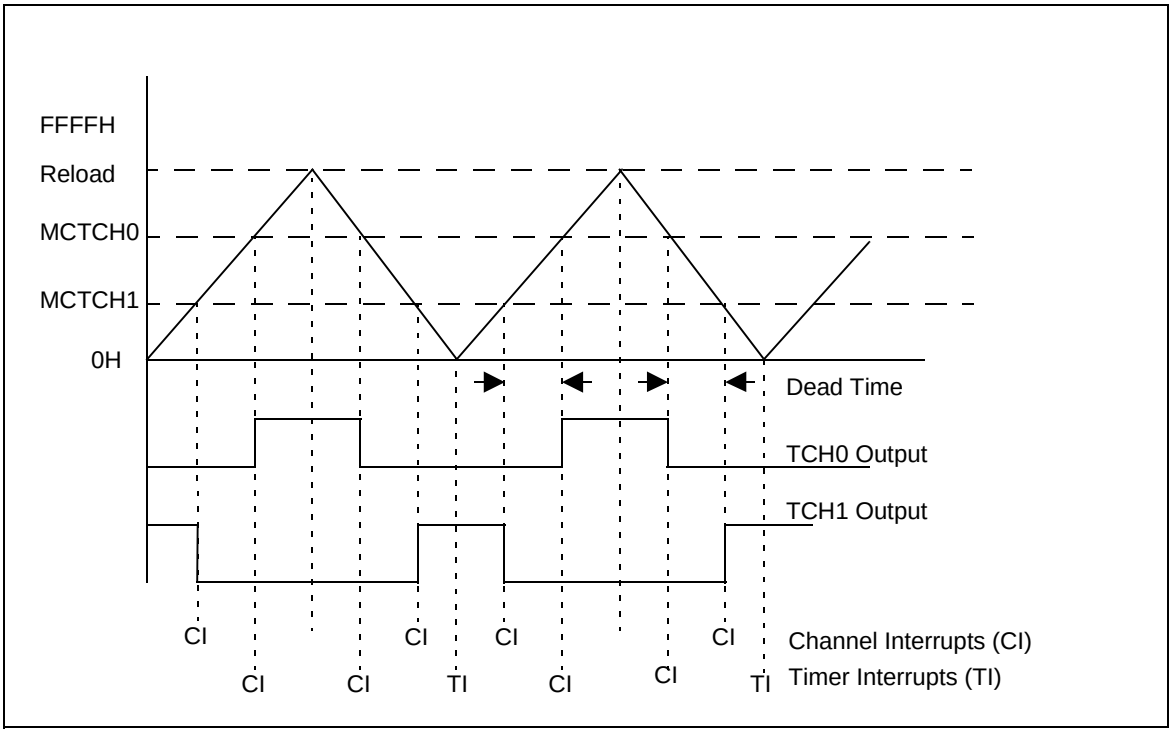



**Figure 17. Count Up/Down Mode with PWM Channel Outputs and Deadband**

## 10.6.2. Multiple Timer Intervals Generation

Figure 18 shows a timing diagram featuring two constant time intervals, T0 and T1. 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 Channel Update Enable (CHUE) bit must be set in channels 0 and 1 so that writes to the Capture/ Compare registers take affect immediately.

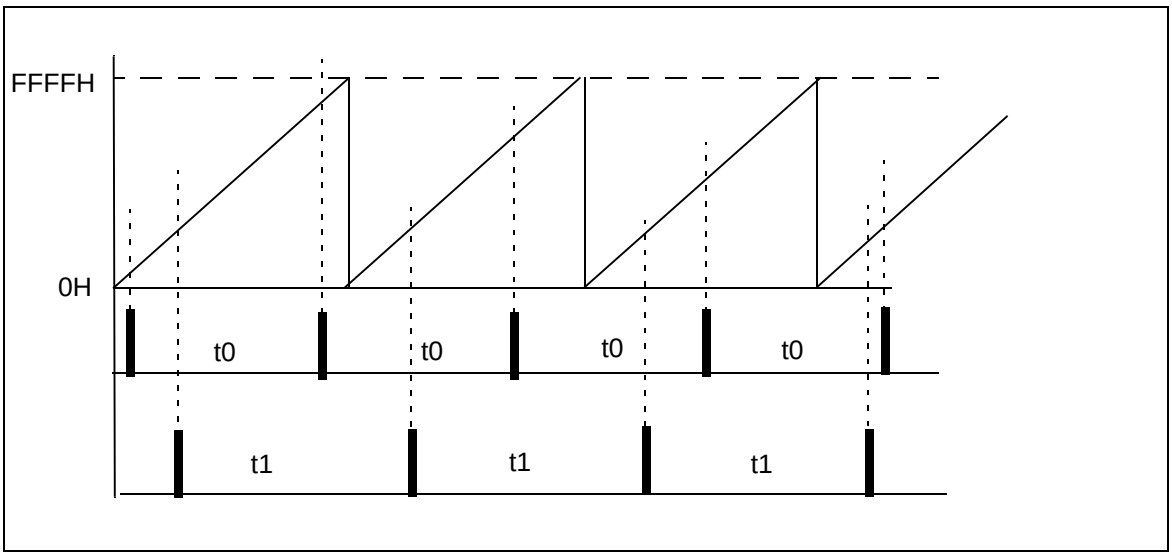



**Figure 18. Count Max Mode with Channel Compare**

## 10.7. Multi-Channel Timer Control Register Definitions

This section defines the features of the following Multi-Channel Timer Control registers.

Multi-Channel Timer High and Low Byte Registers: see page 130

Multi-Channel Timer Reload High and Low Byte Registers: see page 130

Multi-Channel Timer Subaddress Register: see page 131

Multi-Channel Timer Subregister x (0, 1, or 2): see page 132

Multi-Channel Timer Control 0, Control 1 Registers: see page 132

Multi-Channel Timer Channel Status 0 and Status 1 Registers: see page 135

Multi-Channel Timer Channel-y Control Registers: see page 137

Multi-Channel Timer Channel-y High and Low Byte Registers: see page 139

### 10.7.1. Multi-Channel Timer Address Map

Table 69 defines the byte address offsets for the Multi-channel Timer registers. For saving address space, a subaddress is used for the Timer Control 0, Timer Control 1, Channel Status 0, Channel Status 1, Channel-y Control, and Channel-y High and Low byte registers. Only the Timer High and Low Byte registers and the Reload High and Low Byte registers can be directly accessed.

While writing a subregister, first write the subaddress to Timer Subaddress Register, then write data to subregister0, subregister1, or subregister2. A read is the same as a write.

**Table 69. Multi-Channel Timer Address Map**

| Address/Subaddress | Register/Subregister Name |
|---|---|
| **Direct Access Register** | |
| FA0 | Timer (Counter) High |
| FA1 | Timer (Counter) Low |
| FA2 | Timer Reload High |
| FA3 | Timer Reload Low |
| FA4 | Timer Subaddress |
| FA5 | Subregister 0 |
| FA6 | Subregister 1 |
| FA7 | Subregister 2 |
| **Subregister 0** | |
| 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 |
| **Subregister 1** | |
| 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 Low |
| **Subregister 2** | |
| 0 | Reserved |
| 1 | Reserved |
| 2 | Channel A Control |
| 3 | Channel B Control |
| 4 | Channel C Control |
| 5 | Channel D Control |

## 10.7.2. Multi-Channel Timer High and Low Byte Registers

The High and Low Byte (MCTH and MCTL) registers, shown in Table 70, contain the current 16-bit Multi-Channel Timer count value.

Zilog does not recommend writing to the Multi-Channel Timer High and Low Byte registers while the Multi-Channel Timer is enabled. If either or both of the Multi-Channel Timer 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 70. Multi-Channel Timer High and Low Byte Registers (MCTH, MCTL)**

| Bit | 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 |
| Address | FA0H | | | | | | | |

| Bit | 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 |
| Address | FA1H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] MCTH, MCTL | **Multi-Channel Timer High and Low Byte**<br>These bytes contain the current 16-bit Multi-Channel Timer count value, {MCTH[7:0], MCTL[7:0]}. |

When the Multi-Channel Timer 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 the Multi-Channel Timer is enabled. When the Multi-Channel Timer is disabled, reads from MCTL read the register directory. The Multi-Channel Timer High and Low Byte registers are not reset when TEN = 0.

## 10.7.3. Multi-Channel Timer Reload High and Low Byte Registers

The Multi-Channel Timer Reload High and Low Byte (MCTRH and MCTRL) registerss, shown in Table 71, 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.

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

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

**Table 71. Multi-Channel Timer Reload High and Low Byte Registers (MCTRH, MCTRL)**

| Bit | 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 |
| Address | FA2H | | | | | | | |

| Bit | 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 |
| Address | FA3H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] MCTRH, MCTRL | **Multi-Channel Timer Reload Register High and Low**<br>These two bytes form the 16-bit reload value, {MCTRH[7:0], MCTRL[7:0]}. This value sets the Multi-Channel Timer period in Modulo and Up/Down Count modes. |

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 Multi-Channel Timer reload value.

## 10.7.4. Multi-Channel Timer Subaddress Register

The Multi-Channel Timer Subaddress Register stores 3-bit subaddresses for subregisters. These three bits are from MCTSAR[2:0], all other bits are reserved. When accessing sub-register (writing or reading), set MCTSA right value first, then access subregister by writing or reading Subregisters 0, 1, or 2.

**Table 72. Multi-Channel Timer Subaddress Register (MCTSA)**

| Bit | 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 |
| Address | FA4H | | | | | | | |

## 10.7.5. Multi-Channel Timer Subregister x (0, 1, or 2)

The Multi-Channel Timer subregisters 0, 1 or 2 store the 8-bit data write to subregister or 8-bit data read from subregister. The Multi-Channel Timer Subaddress Register selects the subregister to be written to or read from.

**Table 73. Multi-Channel Timer Subregister x (MCTSRx)**

| Bit | 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 |
| Address | FA5H, FA6H, FA7H | | | | | | | |

## 10.7.6. 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 the MCTCTL1 Register are buffered when TEN = 1 and will not take effect until the next end of the cycle count occurs.

**Table 74. Multi-Channel Timer Control 0 Register (MCTCTL0)**

| Bit | 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/W | R | R/W | R | R | R/W | R/W | R/W |
| Address | See note. | | | | | | | |
| Note: If a 00H is in the Subaddress Register, it is accessible through Subregister 0. | | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TCTST | **Timer Count Status**<br>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.<br>0 = Timer count cycle is not complete.<br>1 = Timer count cycle is complete. |
| [6]<br>CHST | **Channel Status**<br>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.<br>0 = No channel capture/compare event has occurred.<br>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. |
| [5]<br>TCIEN | **Timer Count Interrupt Enable**<br>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.<br>0 = Timer Count Interrupt is disabled.<br>1 = Timer Count Interrupt is enabled. |
| [4:3] | Reserved; must be 0. |
| 2:0<br>TCLKS | **Timer Clock Source**<br>000 = System Clock (Prescaling enabled)<br>001 = Reserved<br>010 = System Clock gated by active High Timer Input signal (Prescaling enabled).<br>011 = System Clock gated by active Low Timer Input signal (Prescaling enabled).<br>100 = Timer I/O pin input rising edge (Prescaler disabled).<br>101 = Timer I/O pin input falling edge (Prescaler disabled).<br>110 = Reserved.<br>111 = Reserved. |

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

**Table 75. Multi-Channel Timer Control 1 Register (MCTCTL1)**

| Bit | 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 |
| Address | See note. | | | | | | | |
| Note: If a 00H is in the Subaddress Register, it is accessible through Subregister 1. | | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TEN | **Timer Enable**<br>0 = Timer is disabled and the counter is reset.<br>1 = Timer is enabled to count. |
| [6] | Reserved; must be 0. |
| [5:3]<br>PRES | **Prescale Value**<br>The system clock is divided by 2PRES, 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.<br>000 = Divide by 1<br>001 = Divide by 2<br>010 = Divide by 4<br>011 = Divide by 8<br>100 = Divide by 16<br>101 = Divide by 32<br>110 = Divide by 64<br>111 = Divide by 128 |
| [2] | Reserved; must be 0. |
| [1:0]<br>TMODE | **Timer Mode**<br>00 = Count Modulo: Timer Counts up to Reload Register value. Then it is reset to 0000H and counting up resumes.<br>01 = Reserved.<br>10 = Count up/down: Timer Counts up to Reload and then counts down to 0000H. The count up and count down cycle continues.<br>11 = Reserved. |

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

The Multi-Channel Timer Channel Status 0 and Status 1 registers (MCTCHS0, MCTCHS1) indicate channel overruns and channel capture/compare events.

**Table 76. Multi-Channel Timer Channel Status 0 Register (MCTCHS0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | CHDEO | CHCEO | CHBEO | CHAEO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Address | See note. | | | | | | | |
| Note: If a 01H is in the Subaddress Register, it is accessible through Subregister 0. | | | | | | | | |

| Bit | Description |
|---|---|
| [7:4] | Reserved; must be 0. |
| [3:0] CHyEO | **Channel y Event Flag Overrun**<br>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).<br>0 = No Overrun.<br>1 = Capture/Compare Event Flag Overrun |

**Table 77. Multi-Channel Timer Channel Status 1 Register (MCTCHS1)**

| Bit | 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/W | R/W | R/W | R/W |
| Address | See note. | | | | | | | |

Note: If a 01H is in the Subaddress Register, it is accessible through Subregister 1.

| Bit | Description |
|---|---|
| [7:4] | Reserved; must be 0. |
| [3:0] CHyEF | **Channel y Event Flag**<br>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 Multi-Channel Timer 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).<br>0 = No Capture/Compare Event occurred for this channel.<br>1 = A Capture/Compare Event occurred for this channel. |

## 10.7.8. 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 78. Multi-Channel Timer Channel Control Register (MCTCHyCTL)[1]**

| Bit | 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 |
| Address | See note 2. | | | | | | | |

Notes:
1. y = A, B, C, D.
2. If 02H, 03H, 04H and 05H are in the Subaddress Register, they are accessible through Subregister 2.

| Bit | Description |
|---|---|
| [7]<br>CHEN | **Channel Enable**<br>0 = Channel is disabled.<br>1 = Channel is enabled. |
| [6]<br>CHPOL | **Channel Input/Output Polarity**<br>Operation of this bit is a function of the current operating method of the channel.<br>**ONE-SHOT Operation**<br>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.<br>**CONTINUOUS COMPARE Operation**<br>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.<br>**PWM OUTPUT Operation**<br>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).<br>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).<br>**CAPTURE Operation**<br>0 = Count is captured on the rising edge of the Channel Input signal.<br>1 = Count is captured on the falling edge of the Channel Input signal. |

| Bit | Description (Continued) |
|---|---|
| [5]<br>CHIEN | **Channel Interrupt Enable**<br>This bit enables generation of channel interrupt. A channel interrupt is generated whenever there is a capture/compare event on the Timer Channel.<br>0 = Channel interrupt is disabled.<br>1 = Channel interrupt is enabled. |
| [4]<br>CHUE | **Channel Update Enable**<br>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.<br>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.<br>1 = Writes to the Channel High and Low Byte registers are not buffered when TEN = 1. |
| [3] | Reserved; must be 0. |
| [2:0]<br>CHOP | **Channel Operation Method**<br>This field determines the operating mode of the channel. For a detailed description of the operating modes, see Count Up/Down Mode on page 123.<br>000 = One-Shot Compare operation.<br>001 = Continuous Compare operation.<br>010 = PWM Output operation.<br>011 = Capture operation.<br>100 – 111 = Reserved. |

## 10.7.9. 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 until the next timer end count, unless CHUE = 1.

**Table 79. Multi-Channel Timer Channel-y High Byte Registers (MCTCHyH)***

| Bit | 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 |
| Address | See note. | | | | | | | |
| Note: If 02H, 03H, 04H and 05H are in the Subaddress Register, they are accessible through Subregister 0. | | | | | | | | |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | CHyL | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | | | | | | | | |
| Note: If 02H, 03H, 04H and 05H are in the Subaddress Register, they are accessible through Subregister 1. | | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] CHyH, CHyL | **Multi-Channel Timer Channel-y High and Low Bytes**<br>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 appropriate Channel Input transition occurs. |
| Note: *y = A, B, C, D. | |

# Chapter 11. Watchdog Timer

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

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

## 11.1. 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. After it is enabled, the WDT always counts and must be refreshed to prevent a time-out. An enable can be performed by executing the WDT instruction or by writing the WDT_AO option bit. When cleared to 0, the WDT_AO bit enables the WDT to operate continuously, 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 Oscillator Control 0 Register (OSCCTL0)[1]. 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 time-out period is calculated using the following equation:

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

In the above equation, the WDT reload value is computed using {WDTH[7:0], WDTL[7:0]} and the typical Watchdog Timer RC Oscillator frequency is 10 kHz. Users must consider system requirements when selecting the time-out delay. Table 80 indicates the approximate time-out delays for the default and maximum WDT reload values.

1. For details about this register, see Table 170 on page 319.

Table 80. Watchdog Timer Approximate Time-Out Delays

| WDT Reload Value (Hex) | WDT Reload Value (Decimal) | Approximate Time-Out Delay (with 10 kHz Typical WDT Oscillator Frequency) | |
|---|---|---|---|
| | | Typical | Description |
| 0400 | 1024 | 102 ms | Reset default value time-out delay. |
| FFFF | 65,536 | 6.55 s | Maximum time-out delay. |

## 11.1.1. 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 the eZ8 CPU is operating in DEBUG Mode (through the OCD), the WDT is continuously refreshed to prevent unnecessary WDT time-outs.

## 11.1.2. Watchdog Timer Time-Out Response

The WDT times out when the counter reaches 0000H. A time-out of the WDT generates either a system exception or a Reset. The WDT_RES option bit determines the time-out response of the WDT. For information about programming the WDT_RES option bit, see the Flash Option Bits section on page 276.

### 11.1.2.1. WDT System Exception in Normal Operation

If it is configured to generate a system exception when a time-out 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 time-out and system exception generation, the WDT is reloaded automatically and continues counting.

### 11.1.2.2. 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 time-out occurs and the CPU is in STOP Mode. Both the WDT status bit and the stop bit in the Reset Status Register are set to 1 following a WDT time-out in STOP Mode.

Upon 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.

#### 11.1.2.3. WDT Reset in Normal Operation

The WDT forces the device into the Reset state if it is configured to generate a Reset when a time-out occurs; the WDT status bit is set to 1 (for details, see the Reset Status Register section on page 40). For more information about Reset and the WDT status bit, see the Reset, Stop Mode Recovery and Low-Voltage Detection section on page 31. Following a Reset sequence, the WDT Counter is initialized with its reset value.

#### 11.1.2.4. WDT Reset in STOP Mode

If enabled in STOP Mode and configured to generate a Reset when a time-out occurs and the device is in STOP Mode, the WDT initiates a Stop Mode Recovery. Both the WDT status bit and the stop bit in the Reset Status Register (RSTSTAT) are set to 1 following a WDT time-out in STOP Mode. For more information, see the Reset, Stop Mode Recovery and Low-Voltage Detection section on page 31.

### 11.1.3. 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 time-out 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 appropriate value to the Watchdog Timer Reload High Register (WDTH).
4. Write the appropriate value to the Watchdog Timer Reload Low Register (WDTL). After this write occurs, the Watchdog Timer Reload registers are again locked.

All steps of the WDT Reload Unlock sequence must be written in the sequence defined above. The values in these WDT Reload registers are loaded into the counter every time a WDT instruction is executed.

## 11.2. Watchdog Timer Register Definitions

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

### 11.2.1. Watchdog Timer Reload High and Low Byte Registers

The Watchdog Timer Reload High and Low Byte (WDTH, WDTL) registers, shown in Tables 81 and 82, form the 16-bit reload value that is loaded into the Watchdog Timer when a WDT instruction executes; this 16-bit reload value is {WDTH[7:0], WDTL[7:0]}. Writing to these registers following the unlock sequence sets the appropriate reload value. Reading from these registers returns the current WDT count value.

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

| Bit | 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 |
| Address | FF2H | | | | | | | |

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

| Bit | 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 |
| Address | FF3H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] WDTH, WDTL | **Watchdog Timer Reload High and Low Bytes**<br>WDTH: The WDT Reload High Byte is the most significant byte, or bits [15:8] of the 16-bit WDT reload value.<br>WDTL: The WDT Reload Low Byte is the least significant byte, or bits [7:0] of the 16-bit WDT reload value. |

# Chapter 12. 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!® 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 can 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

## 12.1. LIN-UART 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 displays the LIN-UART architecture.

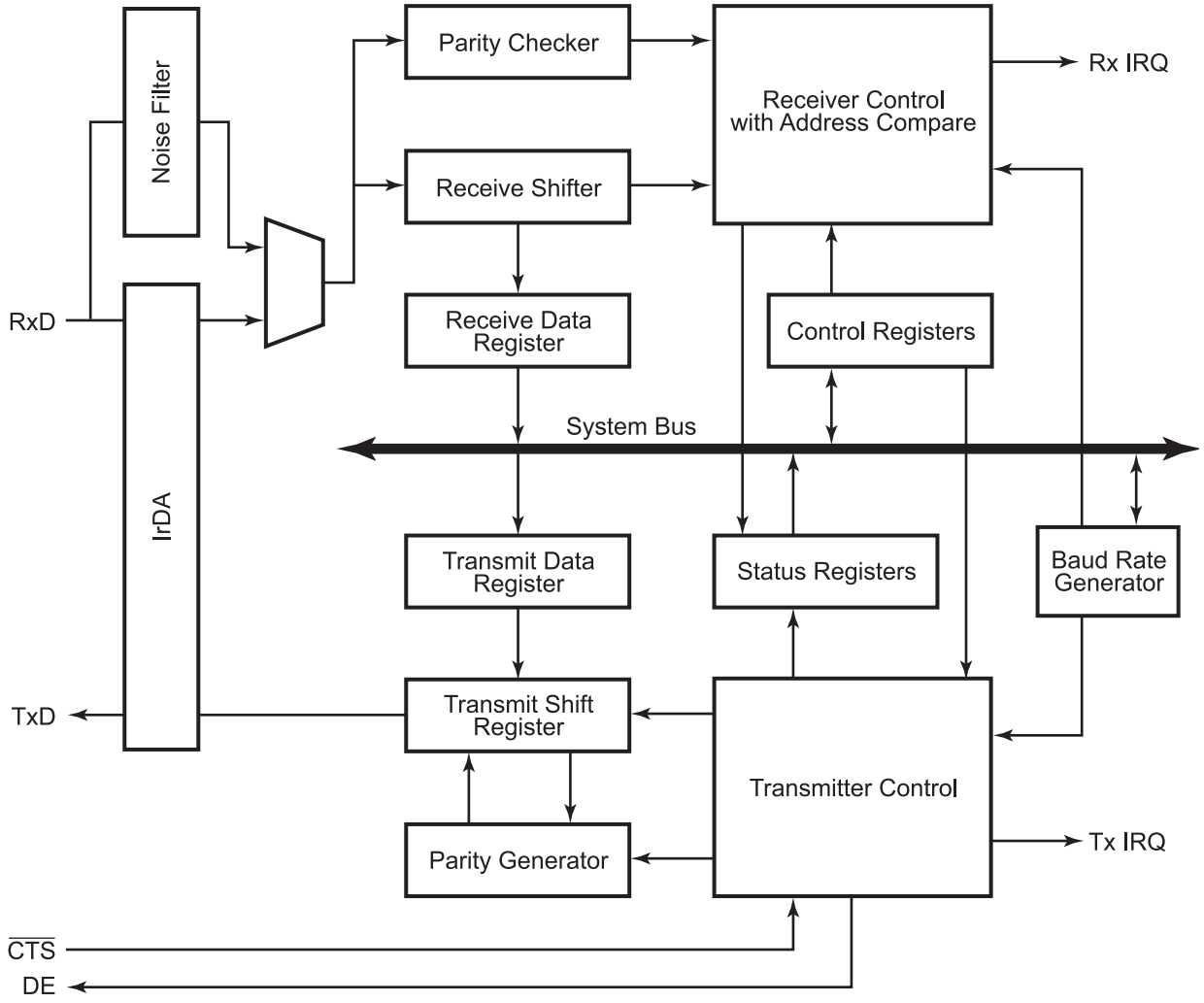



**Figure 19. LIN-UART Block Diagram**

## 12.1.1. 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. Figures 20 and 21 display the asynchronous data format employed by the LIN-UART without parity and with parity, respectively.

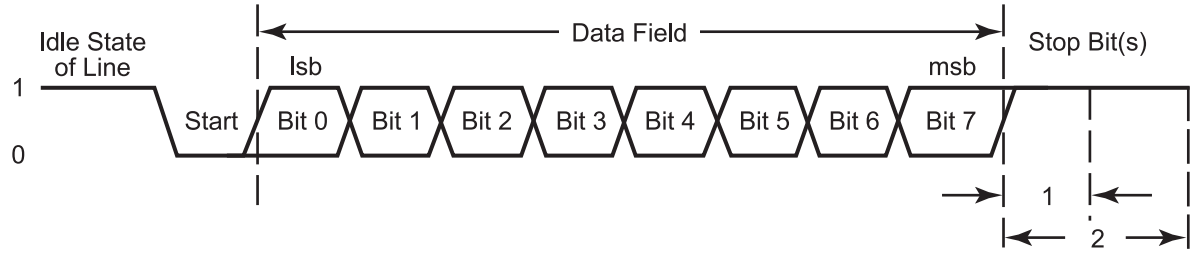



**Figure 20. LIN-UART Asynchronous Data Format without Parity**

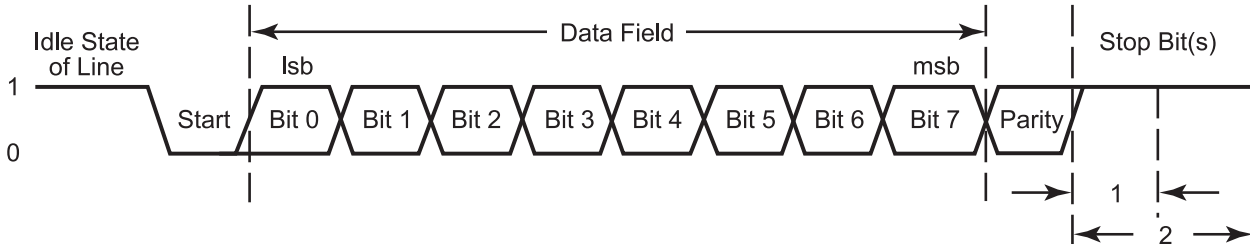



**Figure 21. LIN-UART Asynchronous Data Format with Parity**

## 12.1.2. Transmitting Data using the Polled Method

Observe the following steps 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 bit (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 $\overline{\text{CTS}}$ 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 operating in MULTIPROCESSOR Mode, write to 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 – changes can be made to the Multiprocessor Bit Transmitter (MPBT) value.
10. To transmit additional bytes, return to Step 5.

### 12.1.3. 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. Observe the following steps 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 bit (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 $\overline{\text{CTS}}$ 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, writes to the LIN-UART Control 1 Register to select the outgoing address bit:
   - Sets the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clears it if sending a data byte.
2. Writes 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. Executes 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 executes 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.

## 12.1.4. Receiving Data Using Polled Method

Observe the following steps 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 that is 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.

## 12.1.5. Receiving Data Using the Interrupt-Driven Method

The LIN-UART Receiver interrupt indicates the availability of new data (as well as error conditions). Observe the following steps 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 bit (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.

## 12.1.6. Clear To Send Operation

The Clear To Send ($\overline{\text{CTS}}$) pin, if enabled by the 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.

## 12.1.7. 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 DEPOL bit in the LIN-UART Control Register 1 sets the polarity of the Driver Enable signal.

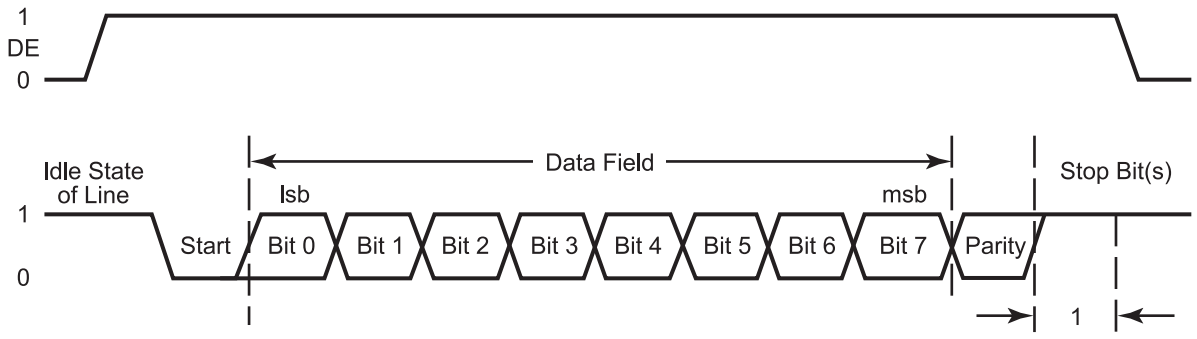



**Figure 22. LIN-UART Driver Enable Signal Timing with One Stop Bit and Parity**

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

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

## 12.1.8. 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.

## 12.1.9. MULTIPROCESSOR Mode

The LIN-UART features a MULTIPROCESSOR (9-bit) Mode that uses an extra ($9^{th}$) 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 displayed in Figure 23.

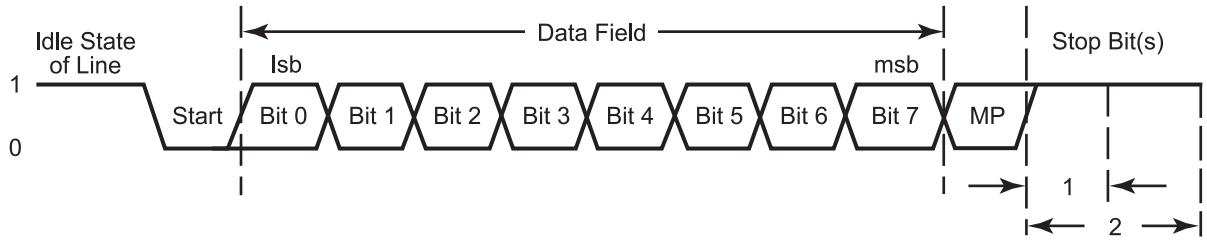



**Figure 23. LIN-UART Asynchronous MULTIPROCESSOR Mode Data Format**

In MULTIPROCESSOR (9-bit) Mode, the Parity bit location (9th 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.

### 12.1.9.1. 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 is not required 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:

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

These modes are selected with MPMD[1:0] in the 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.

## 12.1.10. 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 Status 0 Register). The *message frame* time-out 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:

- A Parity Error (the PE bit in the Status 0 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 definition applies to both Master and Slave operating modes.
- The Break Detect interrupt (BRKD bit in Status 0 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 time-out. 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 Status 0 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 (from the start bit to the beginning of bit 7 of the autobaud character), an Overrun Error is indicated (OE bit in the Status 0 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.

### 12.1.10.1. 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 ±0.5%. A slave with autobaud capability is required to have a baud clock matching the master oscillator within ±14%. The slave nodes autobaud to lock onto the master timing reference with an accuracy of ±2%. If a slave does not contain autobaud capability, it must include a baud clock which deviates from the masters by not more than ±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.

### 12.1.10.2. LIN Mode Initialization and Operation

The LIN protocol mode is selected by setting either the LIN Master (LMST) or LIN SLAVE (LSLV) and optionally (for the LIN SLAVE) the Autobaud Enable (ABEN) bits in the LIN Control Register. To access the LIN Control Register, the Mode Select (MSEL) field of the LIN-UART Mode Select/Status Register must be equal to `010B`. The LIN-

UART Control 0 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.

### 12.1.10.3. 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 after the SBRK bit is set and any character transmission currently underway has completed. The SBRK bit is deasserted by hardware until 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 until 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 Status 0 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.

### 12.1.10.4. 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 can 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 µs 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 can 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 ≥ 3 bit times. The total duration of the Wake-up message in bit times can 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.

### 12.1.10.5. 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 Status 0 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, in which the identifier character is received and the characters of the response section of the message are sent or received. The slave remains in this Active state until a break is received or software forces a state change. After it is in an Active state (i.e., autobaud has completed), a break of 10 or more bit times is recognized and causes 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.

## 12.1.11. 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.

### 12.1.11.1. 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 can be written with the next character to send. This provides 7 bit periods of latency to load the Transmit Data Register before the Transmit Shift Register completes shifting the current character. Writing to the LIN-UART Transmit Data Register clears the TDRE bit to 0.

### 12.1.11.2. 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 after 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)

### 12.1.11.3. 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 cannot 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.

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

Figures 24 displays the recommended procedure for use in LIN-UART receiver interrupt service routines.

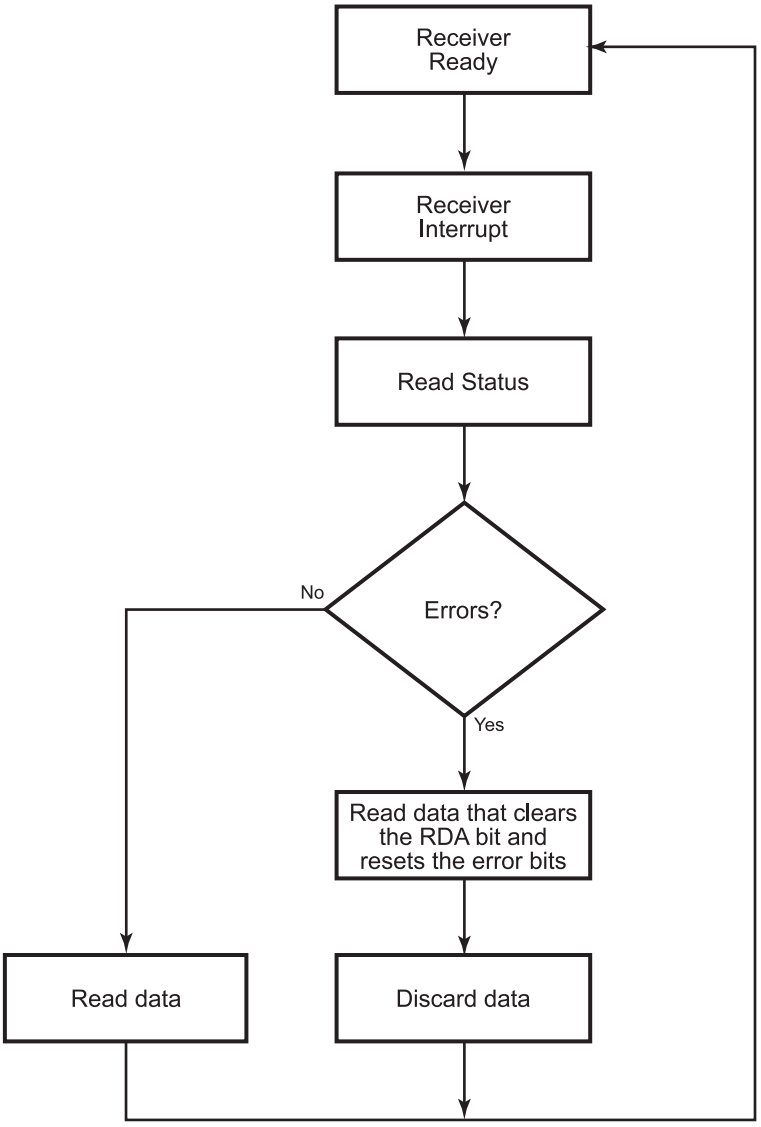



**Figure 24. LIN-UART Receiver Interrupt Service Routine Flow**

### 12.1.11.5. Baud Rate Generator Interrupts

If the BRGCTL bit of the Multiprocessor Control Register (LIN-UART Control 1 Register with MSEL = 000b) 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.

### 12.1.12. 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 time-out. To configure the BRG as a timer with interrupt on time-out, follow the procedure below:

1. Disable the LIN-UART receiver by clearing the REN bit in the LIN-UART Control 0 Register to 0 (i.e., the TEN bit can be asserted; transmit activity can 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.

## 12.2. 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 noise filter 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

## 12.2.1. Architecture

Figure 25 displays how the noise filter is integrated with the LIN-UART on a LIN network.

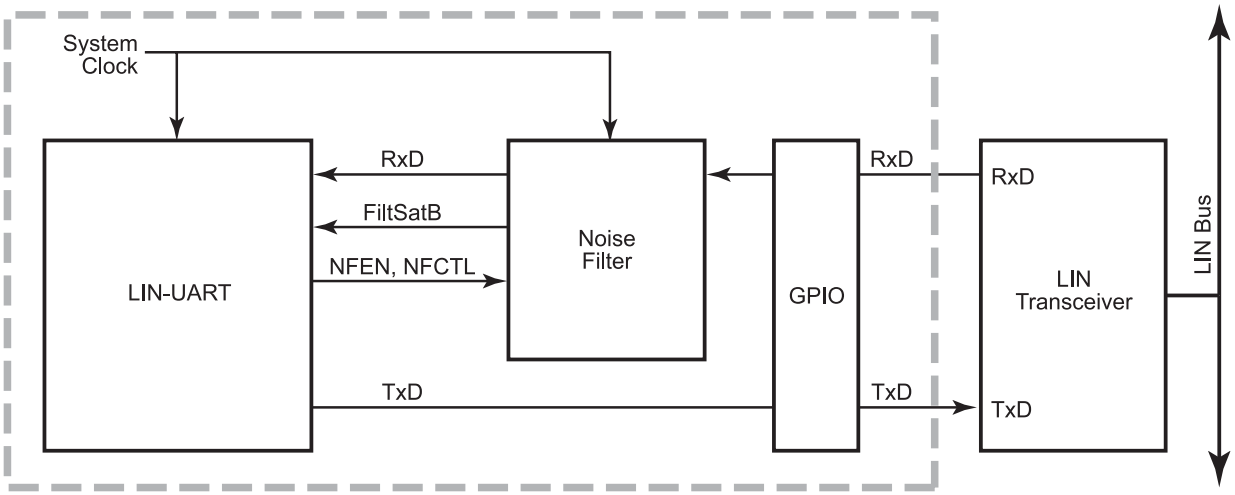



**Figure 25. Noise Filter System Block Diagram**

## 12.2.2. Operation

Figure 26 displays 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.

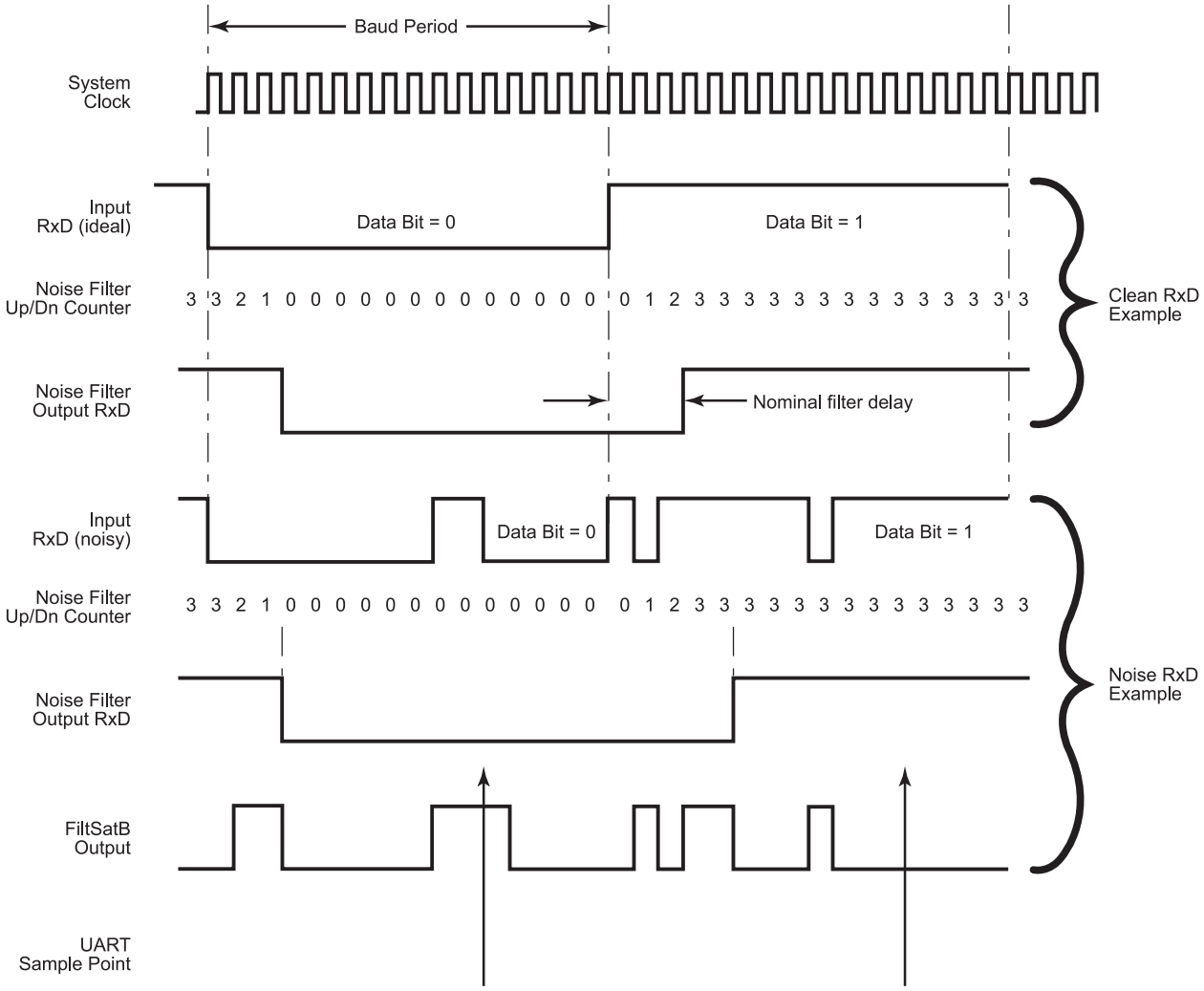



**Figure 26. Noise Filter Operation**

## 12.3. 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 about the infrared operation, see the Infrared Encoder/Decoder section on page 182.

### 12.3.1. LIN-UART Transmit Data Register

Data bytes written to the LIN-UART Transmit Data Register, shown in 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)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TxD | | | | | | | |
| Reset | X | X | X | X | X | X | X | X |
| R/W | W | W | W | W | W | W | W | W |
| Address | F40H, F48H | | | | | | | |

Note: W = Write; X = undefined.

| Bit | Description |
|---|---|
| [7:0]<br>TxD | **Transmit Data**<br>LIN–UART transmitter data byte to be shifted out through the TxD pin. |

## 12.3.2. LIN-UART Receive Data Register

Data bytes received through the RxD pin are stored in the LIN-UART Receive Data Register as shown 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)**

| Bit | 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 |
| Address | F40H, F48H | | | | | | | |

Note: R = Read.

| Bit | Description |
|---|---|
| [7:0] RxD | **Receive Data**<br>LIN–UART receiver data byte from the RxD pin. |

## 12.3.3. 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 describes the Status 0 Register for LIN mode.

**Table 85. LIN-UART Status 0 Register—Standard UART Mode (U0STAT0 = F41H)**

| Bit | 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 |
| Address | F41H, F49H | | | | | | | |

Note: R = Read; X = undefined.

| Bit | Description |
|---|---|
| [7]<br>RDA | **Receive Data Available**<br>This bit indicates that the LIN-UART Receive Data Register has received data. Reading the LIN-UART Receive Data Register clears this bit.<br>0 = The LIN-UART Receive Data Register is empty.<br>1 = There is a byte in the LIN-UART Receive Data Register. |
| [6]<br>PE | **Parity Error**<br>This bit indicates that a parity error has occurred. Reading the Receive Data Register clears this bit.<br>0 = No parity error occurred.<br>1 = A parity error occurred. |
| [5]<br>OE | **Overrun Error**<br>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.<br>0 = No overrun error occurred.<br>1 = An overrun error occurred. |
| [4]<br>FE | **Framing Error**<br>This bit indicates that a framing error (no stop bit following data reception) was detected. Reading the Receive Data Register clears this bit.<br>0 = No framing error occurred.<br>1 = A framing error occurred. |
| [3]<br>BRKD | **Break Detect**<br>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.<br>0 = No break occurred.<br>1 = A break occurred. |

| Bit | Description (Continued) |
|---|---|
| [2]<br>TDRE | **Transmitter Data Register Empty**<br>This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit.<br>0 = Do not write to the Transmit Data Register.<br>1 = The Transmit Data Register is ready to receive an additional byte for transmission. |
| [1]<br>TXE | **Transmitter Empty**<br>This bit indicates that the Transmit Shift Register is empty and character transmission is finished.<br>0 = Data is currently transmitting.<br>1 = Transmission is complete. |
| [0]<br>CTS | **Clear to Send Signal**<br>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)**

| Bit | 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 |
| Address | F41H, F49H | | | | | | | |

Note: R = Read.

| Bit | Description |
|---|---|
| [7]<br>RDA | **Receive Data Available**<br>This bit indicates that the Receive Data Register has received data. Reading the Receive Data Register clears this bit.<br>0 = The Receive Data Register is empty.<br>1 = There is a byte in the Receive Data Register. |
| [6]<br>PLE | **Physical Layer Error**<br>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.<br>0 = Transmit and Receive data match.<br>1 = Transmit and Receive data do not match. |

| Bit | Description (Continued) |
|---|---|
| [5]<br>OE | **Receive Data and Autobaud Overrun Error**<br>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.<br>0 = No autobaud or data overrun error occurred.<br>1 = An autobaud or data overrun error occurred. |
| [4]<br>FE | **Framing Error**<br>This bit indicates that a framing error (no stop bit following data reception) is detected. Reading the Receive Data Register clears this bit.<br>0 = No framing error occurred.<br>1 = A framing error occurred. |
| [3]<br>BRKD | **Break Detect**<br>This bit is set in LIN mode if:<br>• It is in Lin Sleep state and a break of at least 4 bit times occurred (Wake-up event) or<br>• It is in Slave Wait Break state and a break of at least 11 bit times occurred (Break event) or<br>• 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.<br><br>0 = No LIN break occurred.<br>1 = LIN break occurred. |
| [2]<br>TDRE | **Transmitter Data Register Empty**<br>This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit.<br>0 = Do not write to the Transmit Data Register.<br>1 = The Transmit Data Register is ready to receive an additional byte for transmission. |
| [1]<br>TXE | **Transmitter Empty**<br>This bit indicates that the Transmit Shift Register is empty and character transmission is completed.<br>0 = Data is currently transmitting.<br>1 = Transmission is complete. |
| [0]<br>ATB | **LIN Slave Autobaud Complete**<br>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. |

## 12.3.4. LIN-UART Mode Select and Status Register

The LIN-UART Mode Select and Status Register, shown in Table 87, contains mode select and status bits.

**Table 87. LIN-UART Mode Select and Status Register (U0MDSTAT = F44H)**

| Bit | 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 |
| Address | F44H, F4CH | | | | | | | |

Note: R = Read; R/W = Read/Write.

| Bit | Description |
|---|---|
| [7:5] MSEL | **Mode Select**<br>This read/write 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.<br>000 = Multiprocessor and normal UART control/status<br>001 = Noise filter control/status<br>010 = LIN protocol control/status<br>011 = Reserved<br>100 = Reserved<br>101 = Reserved<br>110 = Reserved<br>111 = LIN-UART hardware revision (allows hardware revision to be read in the Mode Status field. |
| [4:0] | **Mode Status**<br>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 169.<br>MSEL[2:0]=000, MULTIPROCESSOR Mode status = {0,0,0,NEWFRM, MPRX}<br>MSEL[2:0]=001, Noise filter status = {NE,0,0,0,0}<br>MSEL[2:0]=010, LIN mode status = {NE, RxBreakLength}<br>MSEL[2:0]=011, Reserved; must be 00000<br>MSEL[2:0]=100, Reserved; must be 00000<br>MSEL[2:0]=101, Reserved; must be 00000<br>MSEL[2:0]=110, Reserved; must be 00000<br>MSEL[2:0]=111, LIN-UART hardware revision |

**Table 88. Mode Status Fields**

| | |
|---|---|
| MULTIPROCESSOR Mode Status Field | **NEWFRM**<br>Status bit denoting the start of a new frame. Reading the LIN-UART Receive Data Register resets this bit to 0.<br>0 = The current byte is not the first data byte of a new frame.<br>1 = The current byte is the first data byte of a new frame. |
| | **Multiprocessor Receive (MPRX)**<br>Returns the value of the last multiprocessor bit received. Reading from the LIN-UART Receive Data Register resets this bit to 0. |
| Digital Noise Filter Mode Status Field | **Noise Event (NE); MSEL = 001b**<br>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 can 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. |
| LIN Mode Status Field | **Noise Event (NE); MSEL = 010b**<br>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 can 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. |
| | **RxBreakLength**<br>LIN mode received break length. This field can 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. |
| Hardware Revision Mode Status Field | **Noise Event (NE); MSEL = 111b**<br>This field indicates the hardware revision of the LIN-UART block.<br>00_xxx LIN UART hardware revision.<br>01_xxx Reserved.<br>10_xxx Reserved.<br>11_xxx Reserved. |

## 12.3.5. LIN-UART Control 0 Register

The LIN-UART Control 0 Register, shown in 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)**

| Bit | 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 |
| Address | F42H, F4AH | | | | | | | |

Note: R/W = Read/Write.

| Bit | Description |
|---|---|
| [7]<br>TEN | **Transmit Enable**<br>This bit enables or disables the transmitter. The enable is also controlled by the $\overline{CTS}$ signal and the CTSE bit. If the $\overline{CTS}$ signal is Low and the CTSE bit is 1, the transmitter is enabled.<br>0 = Transmitter disabled.<br>1 = Transmitter enabled. |
| [6]<br>REN | **Receive Enable**<br>This bit enables or disables the receiver.<br>0 = Receiver disabled.<br>1 = Receiver enabled. |
| [5]<br>CTSE | **Clear To Send Enable**<br>0 = The $\overline{CTS}$ signal has no effect on the transmitter.<br>1 = The LIN-UART recognizes the $\overline{CTS}$ signal as an enable control for the transmitter. |
| [4]<br>PEN | **Parity Enable**<br>This bit enables or disables parity. Even or odd is determined by the PSEL bit.<br>0 = Parity is disabled. This bit is overridden by the MPEN bit.<br>1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit. |
| [3]<br>PSEL | **Parity Select**<br>0 = Even parity is sent as an additional parity bit for the transmitter/receiver.<br>1 = Odd parity is sent as an additional parity bit for the transmitter/receiver. |

| Bit | Description (Continued) |
|---|---|
| [2]<br>SBRK | **Send Break**<br>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.<br>0 = No break is sent.<br>1 = The output of the transmitter is 0. |
| [1]<br>STOP | **Stop Bit Select**<br>0 = The transmitter sends one stop bit.<br>1 = The transmitter sends two stop bits. |
| [0]<br>LBEN | **Loop Back Enable**<br>0 = Normal operation.<br>1 = All transmitted data is looped back to the receiver within the IrDA module. |

## 12.3.6. LIN-UART Control 1 Registers

Multiple registers, shown in Tables 90 and 91, 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.

### 12.3.6.1. Multiprocessor Control Register

When MSEL = 000b, the Multiprocessor Control Register, shown in Table 90, provides control for UART MULTIPROCESSOR Mode, IRDA Mode and Baud Rate Timer Mode as well as other features that can apply to multiple modes.

**Table 90. Multiprocessor Control Register (U0CTL1 = F43H with MSEL = 000b)**

| Bit | 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 |
| Address | F43H, F4BH | | | | | | | |

Note: R/W = Read/Write.

| Bit Position | Value | Description |
|---|---|---|
| [7,5] MPMD[1:0] | **MULTIPROCESSOR (9-bit) Mode** | |
| | 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 | **Multiprocessor Enable** This bit is used to enable MULTIPROCESSOR (9-bit) Mode. | |
| | 0 | Disable MULTIPROCESSOR (9-bit) Mode. |
| | 1 | Enable MULTIPROCESSOR (9-bit) Mode. |
| [4] MPBT | **Multiprocessor Bit Transmit** This bit is applicable only when MULTIPROCESSOR (9-bit) Mode is enabled. | |
| | 0 | Send a 0 in the multiprocessor bit location of the data stream (9th bit). |
| | 1 | Send a 1 in the multiprocessor bit location of the data stream (9th bit). |
| [3] DEPOL | **Driver Enable Polarity** | |
| | 0 | DE signal is active High. |
| | 1 | DE signal is active Low. |

| Bit Position | Value | Description (Continued) |
|---|---|---|
| [2]<br>BRGCTL | **Baud Rate Generator Control**<br>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. | |
| | **Baud Rate Generator Control when the LIN-UART receiver is not enabled:** | |
| | 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. |
| | **Baud Rate Generator Control when the LIN-UART receiver is enabled:** | |
| | 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]<br>RDAIRQ | **Receive Data Interrupt** | |
| | 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]<br>IREN | **Loop Back Enable** | |
| | 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. |

## 12.3.7. Noise Filter Control Register

When MSEL = 001b, the Noise Filter Control Register, shown in Table 91, provides control for the digital noise filter.

**Table 91. Noise Filter Control Register (U0CTL1 = F43H with MSEL = 001b)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | NFEN | NFCTL | | | — | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Address | F43H, F4BH | | | | | | | |

Note: R = Read; R/W = Read/Write.

| Bit Position | Value | Description |
|---|---|---|
| [7]<br>NFEN | **Noise Filter Enable** | |
| | 0 | Noise filter is disabled. |
| | 1 | Noise filter is enabled. Receive data is preprocessed by the noise filter. |
| [6:4]<br>NFCTL | **Noise Filter Control**<br>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. | |
| | 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]<br>Reserved | — | Reserved; must be 0000. |

## 12.3.8. LIN Control Register

When MSEL = 010b, the LIN Control Register provides control for the LIN mode of operation.

**Table 92. LIN Control Register (U0CTL1 = F43H with MSEL = 010b)**

| Bit | 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 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | F43H, F4BH | | | | | | | |

Note: R/W = Read/Write.

| Bit Position | Value | Description |
|---|---|---|
| [7] LMST | **LIN MASTER Mode** | |
| | 0 | LIN MASTER Mode not selected. |
| | 1 | LIN MASTER Mode selected (if MPEN, PEN, LSLV = 0). |
| [6] LSLV | **LIN SLAVE Mode** | |
| | 0 | LIN SLAVE Mode not selected. |
| | 1 | LIN SLAVE Mode selected (if MPEN, PEN, LMST = 0). |
| [5] ABEN | **Autobaud Enable** | |
| | 0 | Autobaud not enabled. |
| | 1 | Autobaud enabled, if in LIN SLAVE Mode. |
| [4] ABIEN | **Autobaud Interrupt Enable** | |
| | 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. |

| Bit Position | Value | Description (Continued) |
|---|---|---|
| [3:2]<br>LINSTATE[1:0] | **LIN State Machine**<br>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 until the software sets it back to the Sleep state. After configuration, software does not alter the LinState field during operation. | |
| | 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). |
| [1:0]<br>TxBreakLength | **TxBreakLength**<br>Used in LIN mode by the master to control the duration of the transmitted break. | |
| | 00 | 13 bit times. |
| | 01 | 14 bit times. |
| | 10 | 15 bit times. |
| | 11 | 16 bit times. |

## 12.3.9. LIN-UART Address Compare Register

The LIN-UART Address Compare Register, shown in Table 93, 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.

**Table 93. LIN-UART Address Compare Register (U0ADDR = F45H)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | COMP_ADDR | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | F45H, F4DH | | | | | | | |

Note: R/W = Read/Write.

| Bit Position | Value | Description |
|---|---|---|
| [7:0]<br>COMP_ADDR | — | **Compare Address**<br>This 8-bit value is compared to the incoming address bytes. |

## 12.3.10. LIN-UART Baud Rate High and Low Byte Registers

The LIN-UART Baud Rate High and Low Byte registers, shown in Tables 94 and 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)**

| Bit | 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 |
| Address | F46H, F4EH | | | | | | | |

Note: R/W = Read/Write.

| Bit Position | Value | Description |
|---|---|---|
| [7:0]<br>BRH | — | **Baud Rate High**<br>These bits set the High byte of the baud rate divisor value. |

**Table 95. LIN-UART Baud Rate Low Byte Register (U0BRL = F47H)**

| Bit | 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 |
| Address | F47H, F4FH | | | | | | | |

Note: R/W = Read/Write.

| Bit Position | Value | Description |
|---|---|---|
| [7:0]<br>BRL | — | **Baud Rate Low**<br>These bits set the Low Byte of the baud rate divisor value. |

The LIN-UART data rate is calculated using the following equation for standard UART modes. For the LIN protocol, the Baud Rate registers must be programmed with the baud period rather than 1/16th of the 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. Tables 96 through 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 |

**Table 97. LIN-UART Baud Rates, 10.0 MHz System Clock (Continued)**

| Applicable Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) | Applicable Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
|---|---|---|---|---|---|---|---|
| 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 | | | | |

# Chapter 13. Infrared Encoder/Decoder

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.

## 13.1. Architecture

Figure 27 displays the architecture of the infrared endec.

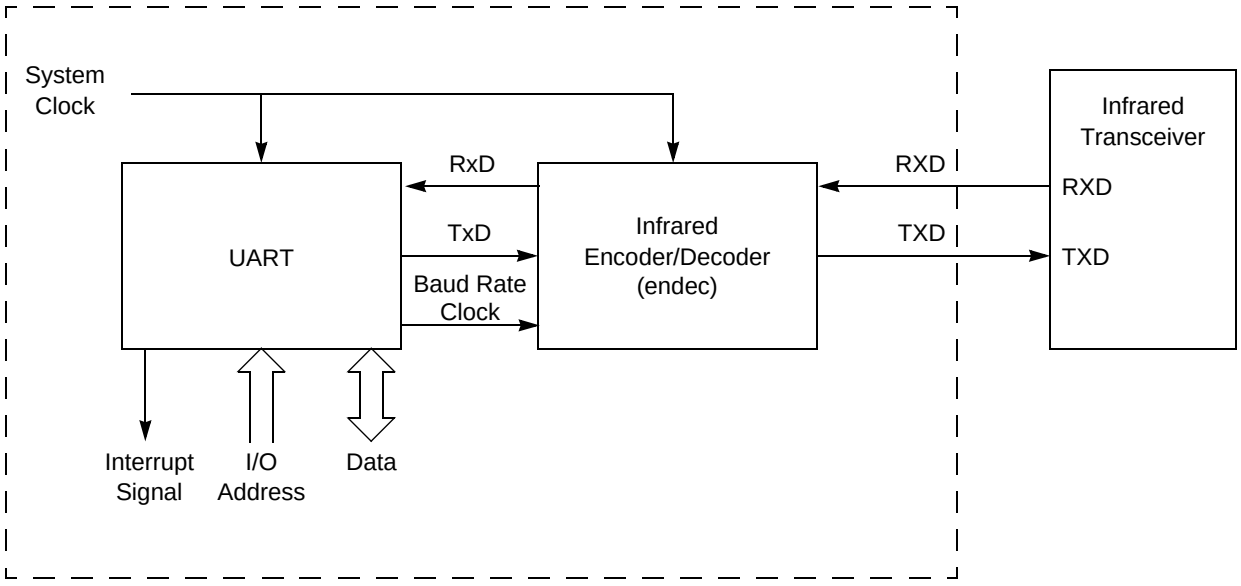



**Figure 27. Infrared Data Communication System Block Diagram**

## 13.2. 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 115.2KBaud. 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:

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

## 13.2.1. 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 displays 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.

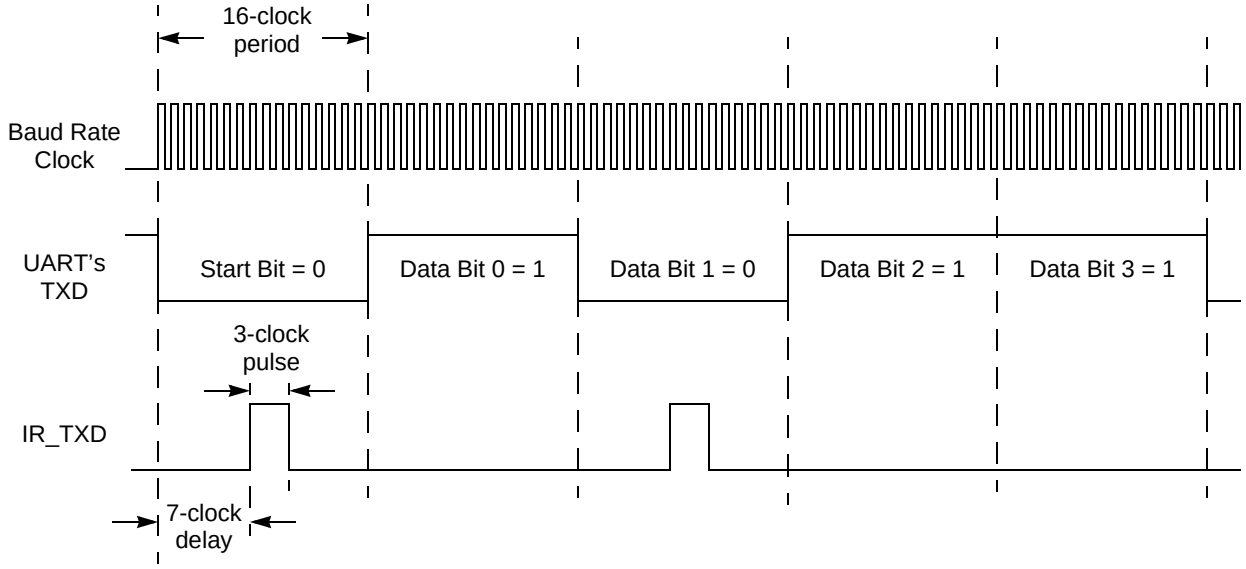



**Figure 28. Infrared Data Transmission**

## 13.2.2. 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 displays 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.

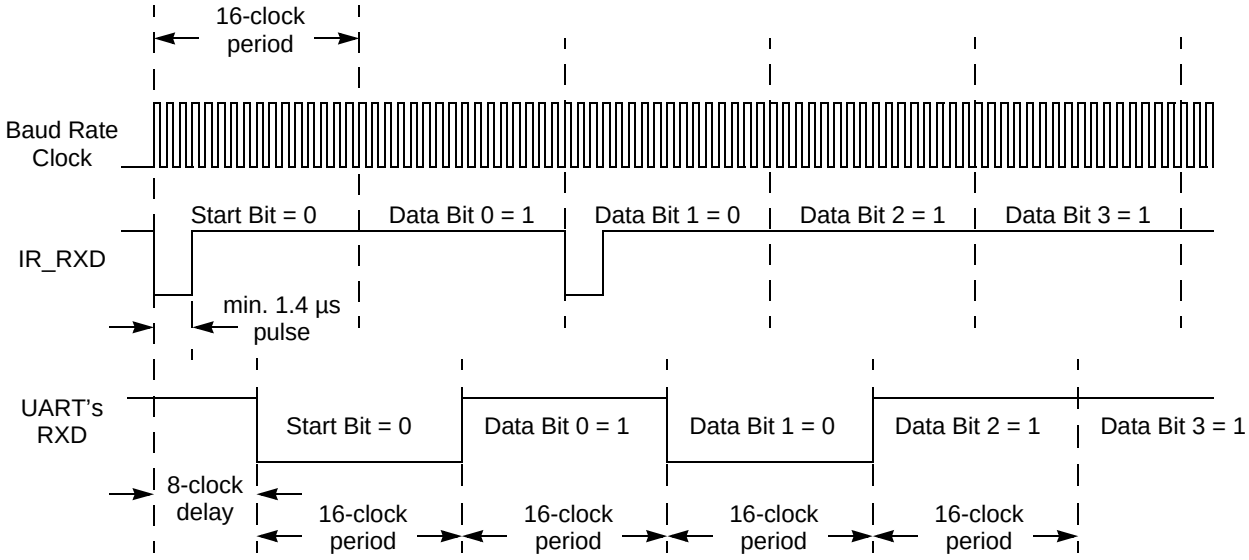



**Figure 29. IrDA Data Reception**

**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.

### 13.2.2.1. 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.

## 13.3. 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 163.

**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 Tables 17 through 19 on pages 49–54 for details.

# Chapter 14. 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 µs
- 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

## 14.1. Architecture

The ADC architecture, shown in Figure 30, 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.

## 14.2. Operation

The ADC converts the analog input, $ANA_X$, to a 10-bit digital representation. The equation for calculating the digital value is calculated by:

$$\text{ADC Output} = 1024 \times (ANA_x \div V_{REF})$$

Assuming zero gain and offset errors, any voltage outside the ADC input limits of $AV_{SS}$ and $V_{REF}$ returns all 0s or 1s, 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, this start bit can be read to indicate ADC operation status (busy or available).

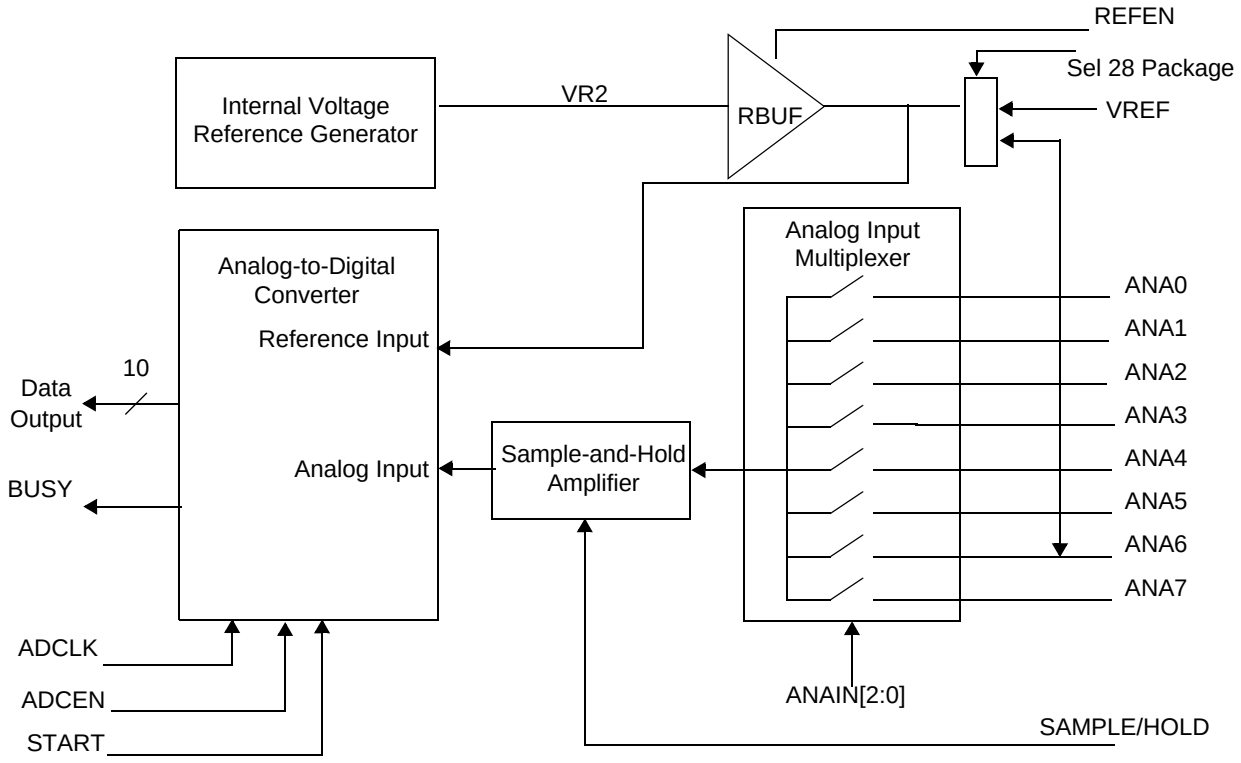



**Figure 30. Analog-to-Digital Converter Block Diagram**

## 14.2.1. ADC Timing

Each ADC measurement consists of 3 phases:

1. Input sampling (programmable, minimum of 1.8µs).
2. Sample-and-hold amplifier settling (programmable, minimum of 0.5µs).
3. Conversion is 13 ADCLK cycles.

Figure 31 displays the timing of an ADC conversion.

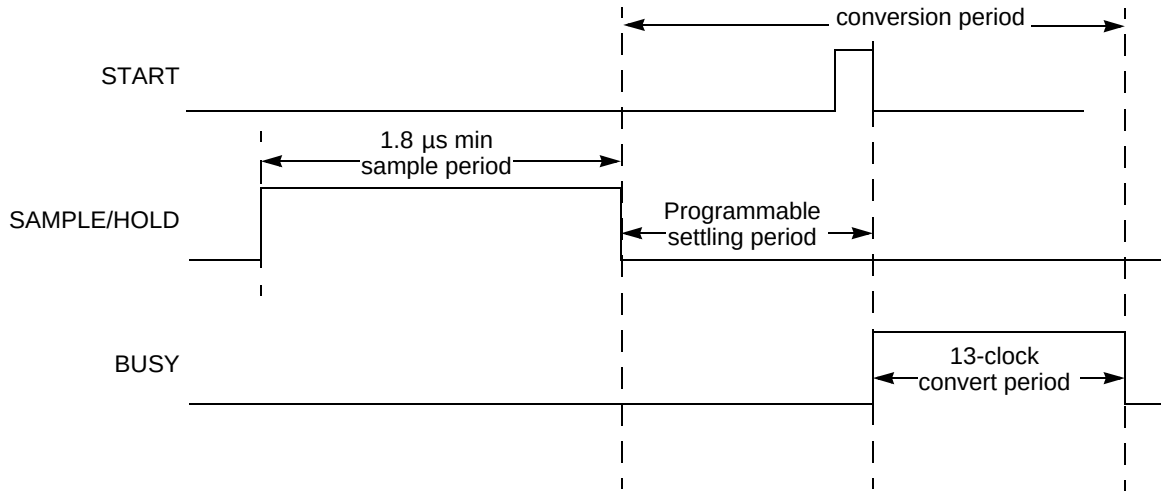



**Figure 31. ADC Timing Diagram**

## 14.2.2. 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. See Figure 32.

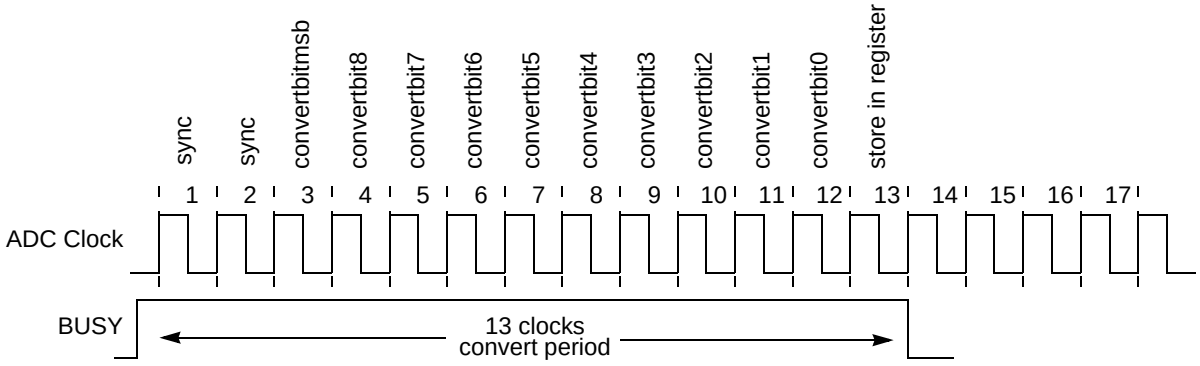



**Figure 32. ADC Convert Timing**

## 14.2.3. 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.

### 14.2.4. Internal Voltage Reference Generator

The Internal Voltage Reference Generator provides the voltage, VR2, for the RBUF. VR2 is 1.6V.

### 14.2.5. 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.

## 14.3. ADC Control Register Definitions

The registers that control analog-to-digital conversion functions are defined in this section.

### 14.3.1. ADC Control Register 0

The ADC Control Register 0, shown in Table 101, initiates the A/D conversion and provides ADC status information.

**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 |
| Address | F70h | | | | | | | |

| Bit Position | Value (H) | Description |
|---|---|---|
| [7] START | 0 | **ADC Start/Busy**<br>Writing a 0 has no effect.<br>Reading a 0 indicates the ADC is available to begin a conversion. |
| | 1 | Writing a 1 starts a conversion.<br>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. |

| Bit Position | Value (H) | Description (Continued) |
|---|---|---|
| [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. |
| [3:0] ANAIN | | **Analog Input Select** |
| | 0000 | 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. |

## 14.3.2. ADC Raw Data High Byte Register

The ADC Raw Data High Byte Register, shown in Table 102, 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 | | | | | | | |
| Address | F71H | | | | | | | |

| Bit Position | Value (H) | Description |
|---|---|---|
| [7:0] | 00–FF | **ADC Raw Data High Byte**<br>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. |

## 14.3.3. ADC Data High Byte Register

The ADC Data High Byte Register, shown in 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 | | | | | | | |
| Address | F72H | | | | | | | |

| Bit Position | Value (H) | Description |
|---|---|---|
| [7:0] | 00–FF | **ADC High Byte**<br>The last conversion output is held in the data registers until the next ADC conversion has completed. |

## 14.3.4. ADC Data Low Bits Register

The ADC Data Low Bits Register, shown in 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 | | | | | |
| Address | F73H | | | | | | | |

| Bit Position | Value (H) | Description |
|---|---|---|
| [7:6] ADCDL | 00–11b | **ADC Low Bit**<br>These bits are the 2 least significant bits of the 10-bit ADC output; they are undefined after a Reset. The Low bits are latched into this register whenever the ADC Data High Byte register is read. |
| [5:0] | 0 | Reserved; must be 0. |

## 14.3.5. Sample Settling Time Register

The Sample Settling Time Register, shown in 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 0.5µs 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 | | | |
| Address | F74H | | | | | | | |

| Bit Position | Value (H) | Description |
|---|---|---|
| [7:4] | 0 | Reserved; must be 0. |
| [3:0] SST | 0–F | Sample settling time in number of system clock periods to meet 0.5µs minimum. |

## 14.3.6. Sample Time Register

The Sample Time Register, shown in 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 µs 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 | | | | | |
| Address | 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. |

## 14.3.7. ADC Clock Prescale Register

The ADC Clock Prescale Register, shown in 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 maintains 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 | | | | | | | |
| Address | F76H | | | | | | | |

| Bit | Description |
|---|---|
| [7:4] | Reserved; must be 0. |
| [3]<br>DIV16 | **Divide by 16**<br>0 = Clock is not divided.<br>1 = System Clock is divided by 16 for ADC Clock. |
| [2]<br>DIV8 | **Divide by 8**<br>0 = Clock is not divided.<br>1 = System Clock is divided by 8 for ADC Clock. |
| [1]<br>DIV4 | **Divide by 4**<br>0 = Clock is not divided.<br>1 = System Clock is divided by 4 for ADC Clock. |
| [0]<br>DIV2 | **Divide by 2**<br>0 = Clock is not divided.<br>1 = System Clock is divided by 2 for ADC Clock. |

**Caution:** The maximum ADC clock at 2.7V–3.6V is 5MHz. The maximum ADC clock at 1.8V–2.7V is 2.5MHz. Set the Prescale Register correctly according to the different system clocks. See the ADC Clock Prescale Register for details.

# Chapter 15. 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 nonzero bias point. In a standard single-supply system, this bias point must be substantially above ground to measure positive input currents. The noninverting input can also be used for offset correction.

**Note:** This amplifier 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 noninverting input.

To use the low-power operational amplifier, it must be enabled in the Power Control Register Definitions, discussed on page 44. 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 (see Power Control Register 0 on page 44). When making normal ADC (i.e., not transimpedance) measurements on ANA0, 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 the Port A–E Alternate Function Subregisters on page 61).

Standard transimpedance measurements are made on ANA0 as selected by the ANAIN[3:0] bits of the ADC Control Register 0, discussed on page 189. 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.

# Chapter 16. Enhanced Serial Peripheral Interface

The Enhanced Serial Peripheral Interface (ESPI) supports the Serial Peripheral Interface (SPI) and other synchronous serial interface modes, such as Inter-IC Sound ($I^2S$) 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

## 16.1. 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 displays a diagram of the ESPI block.

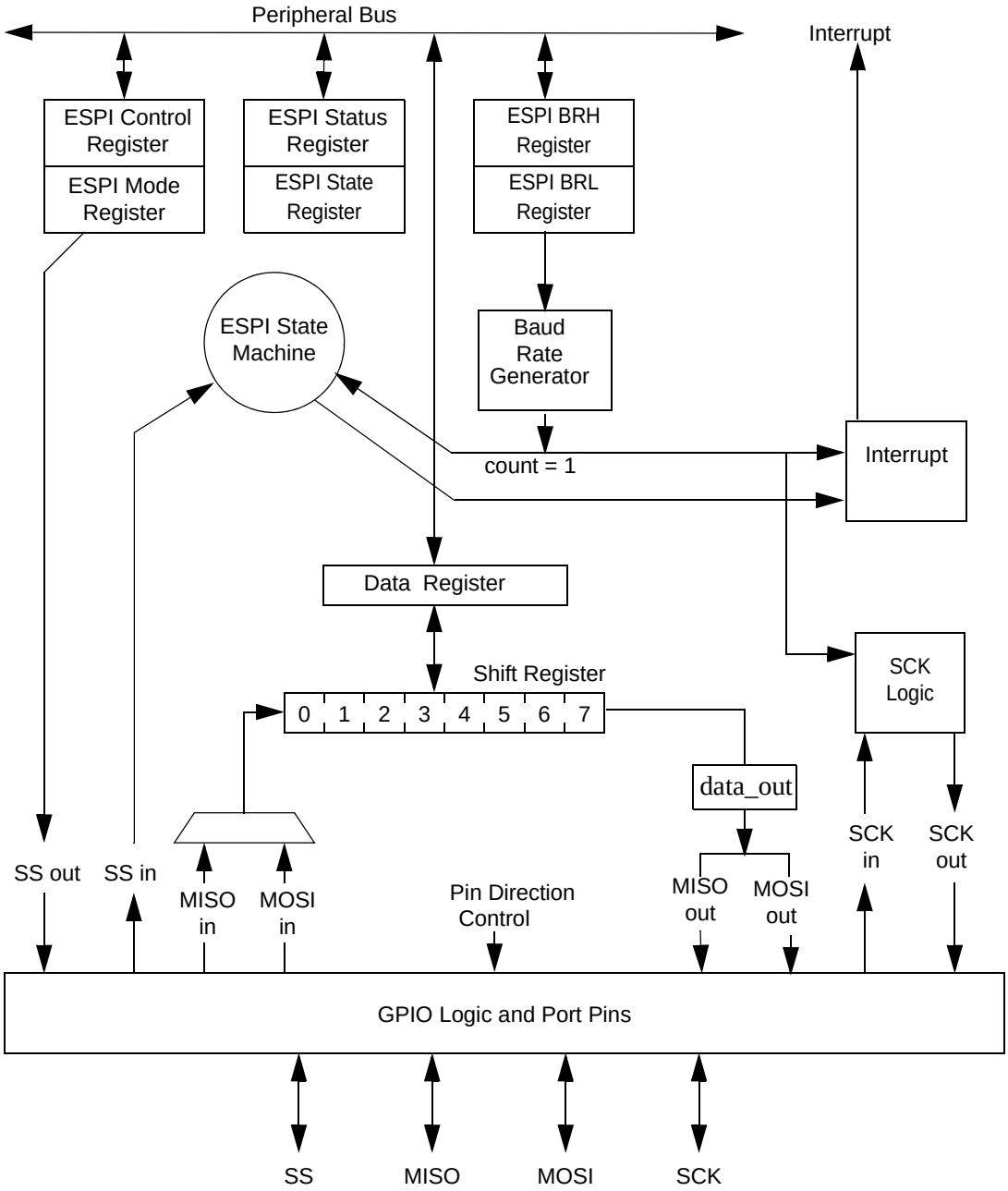



**Figure 33. ESPI Block Diagram**

# 16.2. 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 as they operate in both MASTER and SLAVE modes.

## 16.2.1. 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 most significant bit first. 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.

## 16.2.2. 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 most significant bit first. 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.

## 16.2.3. 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 (see Table 112 on page 217). 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.

### 16.2.4. 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.

## 16.3. 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 SPI 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). After 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 can 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.

## 16.3.1. Throughput

In MASTER Mode, the maximum SCK rate supported is one-half the system clock frequency. This rate 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.

## 16.3.2. 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 108 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 108. 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 |

### 16.3.2.1. Transfer Format when Phase Equals Zero

Figure 34 displays 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 can be interpreted as either a Master or Slave timing diagram because the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the Master and the Slave.

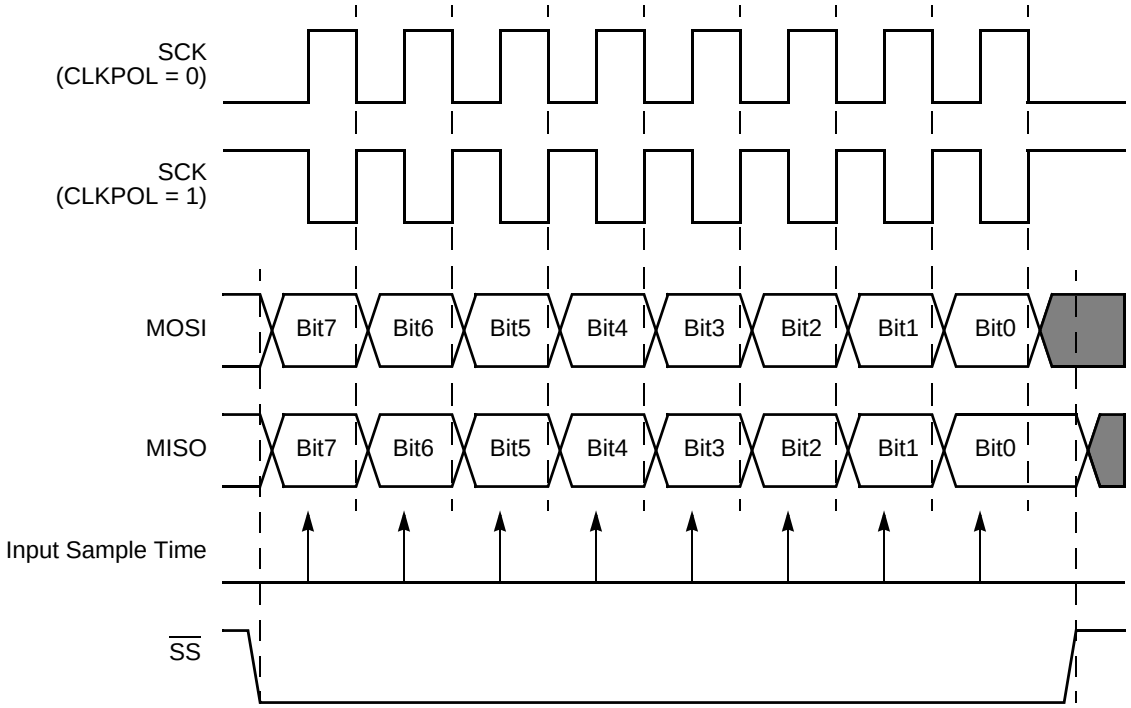



**Figure 34. ESPI Timing when PHASE=0**

### 16.3.2.2. Transfer Format When Phase Equals One

Figure 35 displays a 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.

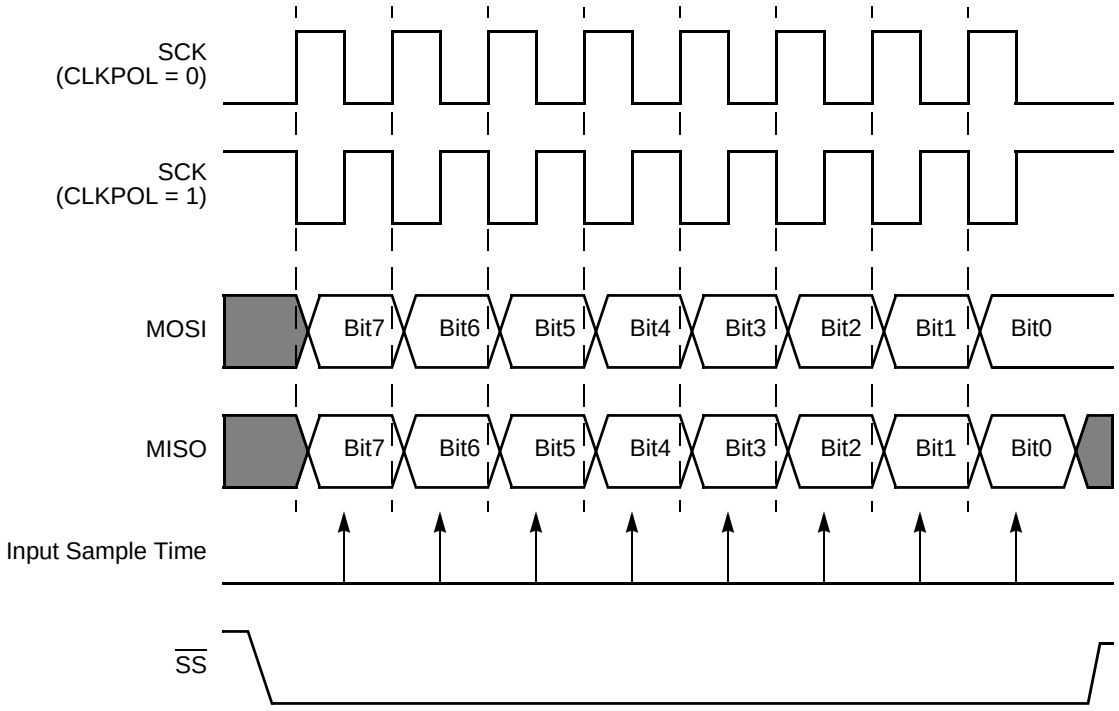



**Figure 35. ESPI Timing when PHASE = 1**

## 16.3.3. 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.

### 16.3.3.1. 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). After 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 displays 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.

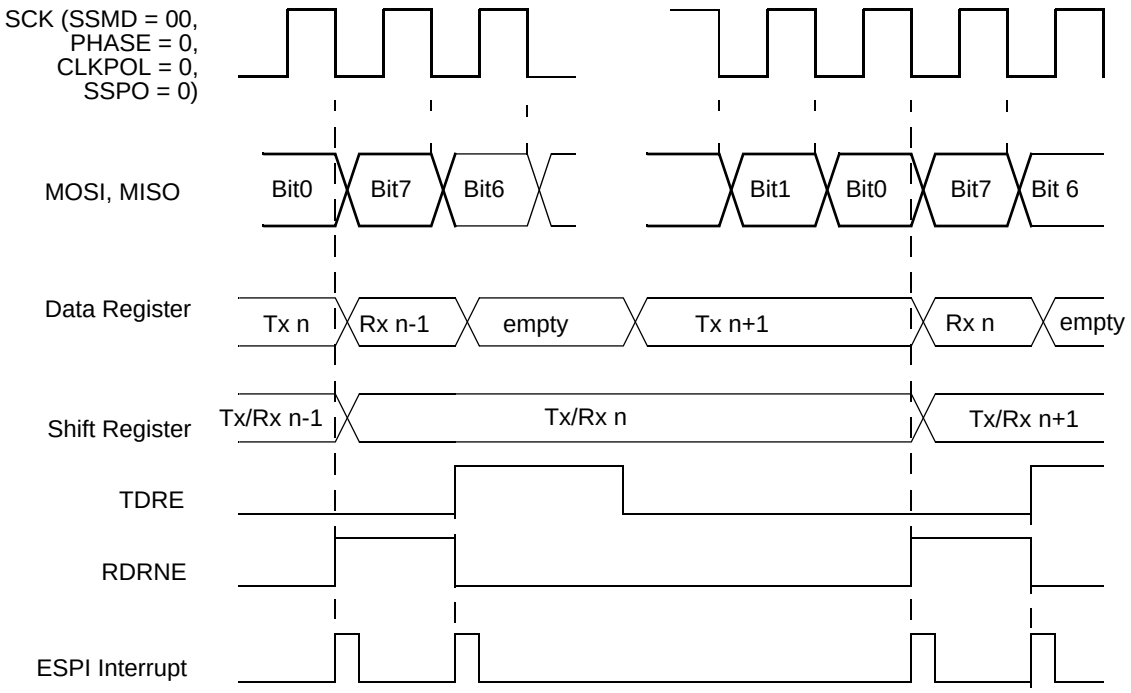



**Figure 36. SPI Mode (SSMD = 00)**

### 16.3.3.2. 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. After the transaction has completed, hardware will clear the SSV bit. Figure 37 displays a frame with synchronous frame sync pulse mode.

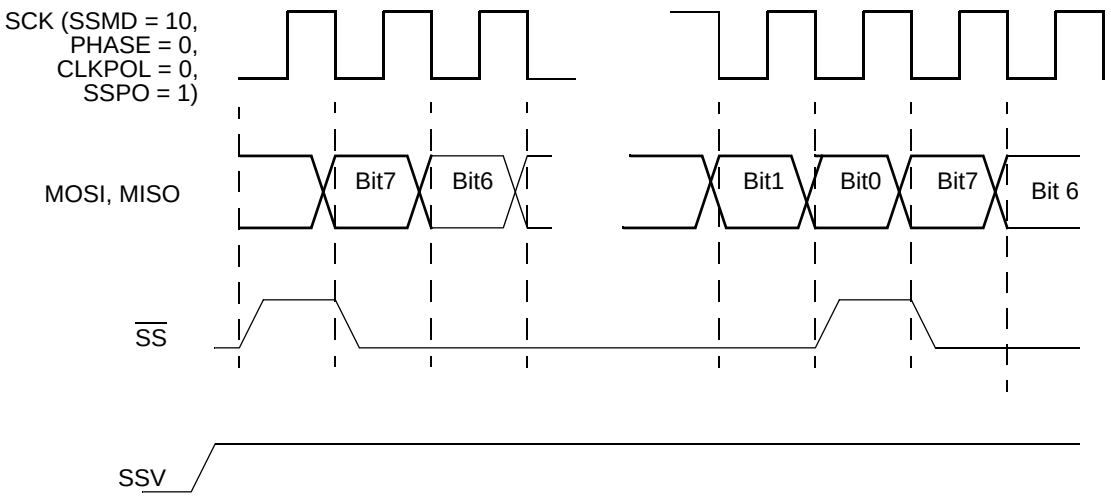



**Figure 37. Synchronous Frame Sync Pulse mode (SSMD = 10)**

### 16.3.3.3. Synchronous Framing with $\overline{SS}$ Mode

This mode is selected by setting the SSMD field of the Mode Register to 11. Figure 38 displays 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 ($I^2S$) 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.

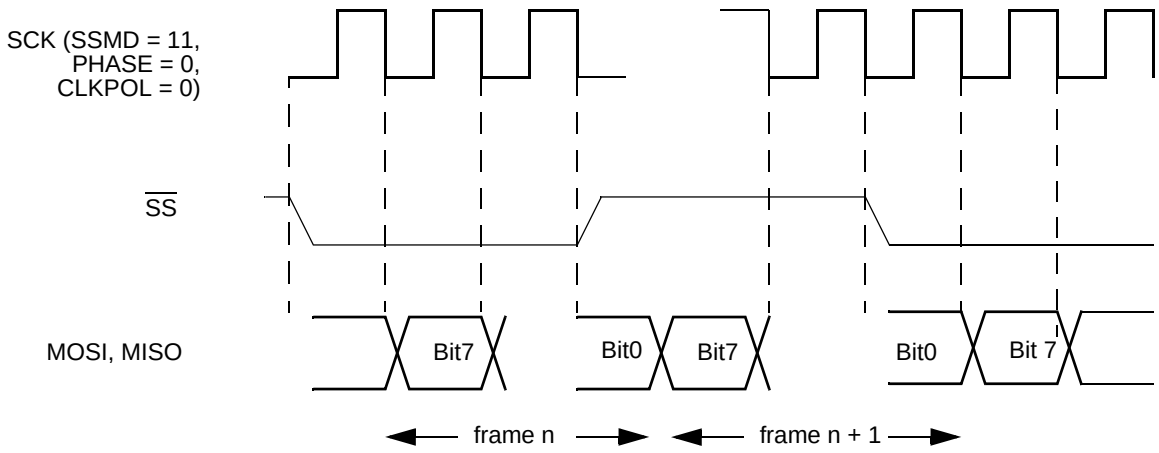



**Figure 38. Synchronous Message Framing Mode (SSMD = 11), Multiple Frames**

## 16.3.4. 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.

### 16.3.4.1. 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, 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. Figures 39 and 40 display a block diagram of the the ESPI configured as an SPI Master.

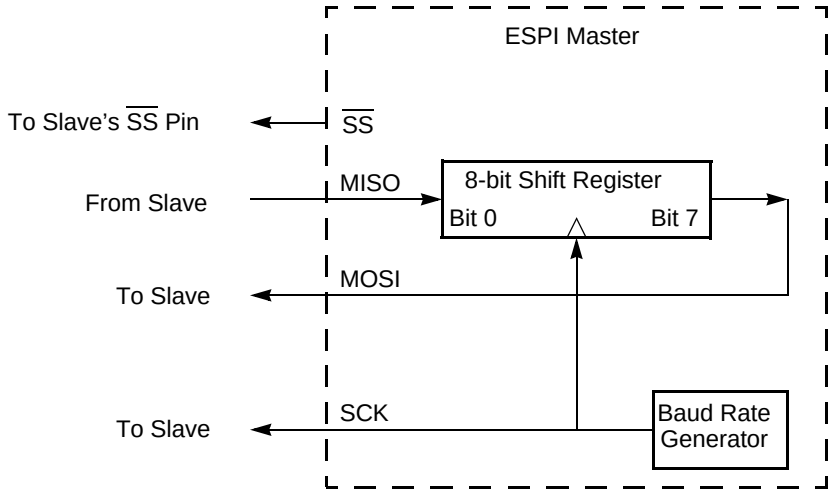



**Figure 39. ESPI Configured as an SPI Master in a Single Master, Single Slave System**

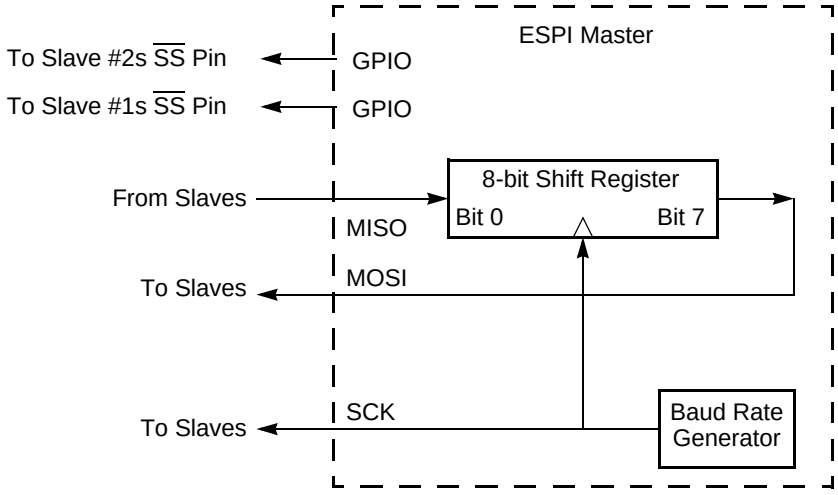



**Figure 40. ESPI Configured as an SPI Master in a Single Master, Multiple Slave System**

### 16.3.4.2. 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.

### 16.3.4.3. 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 1s.

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 displays the ESPI configuration in SPI SLAVE Mode.

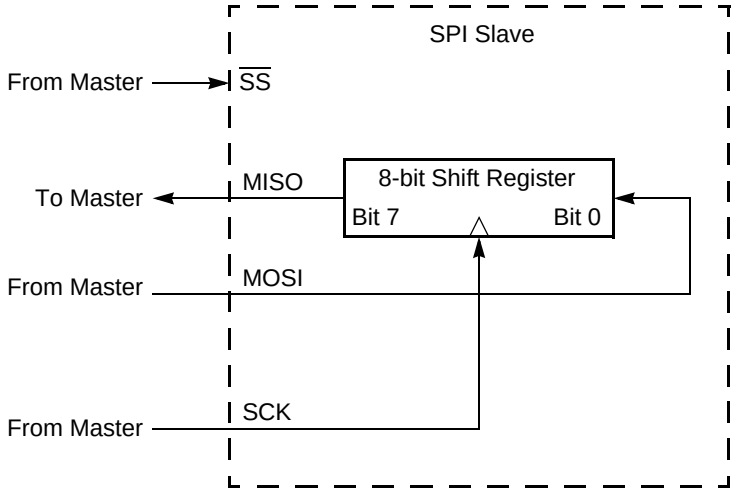



**Figure 41. ESPI Configured as an SPI Slave**

## 16.3.5. 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.

### 16.3.5.1. 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.

### 16.3.5.2. 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.

#### 16.3.5.3. 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.

#### 16.3.5.4. SLAVE Mode Abort

In SLAVE Mode, 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 time-out 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 time-out indicates the Master is stalled or disabled. Writing a 1 to ABT clears this error flag.

### 16.3.6. 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 can 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. After 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 can 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 time-out. This timer function must be enabled by setting the BRGCTL bit in the ESPICTL register. This timer interrupt does not set any of the bits of the ESPI Status Register.

## 16.3.7. 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:

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

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

When the ESPI is disabled, the Baud Rate Generator can function as a basic 16-bit timer with interrupt on time-out. Observe the following steps to configure the Baud Rate Generator as a timer with interrupt on time-out:

1. Disable the ESPI by clearing the ESPIEN1,0 bits in the ESPI Control Register.
2. Load the appropriate 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.

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

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

## 16.4. ESPI Control Register Definitions

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

ESPI Data Register: see page 213

ESPI Transmit Data Command and Receive Data Buffer Control Register: see page 214

ESPI Control Register: see page 215

ESPI Mode Register: see page 217

ESPI Status Register: see page 219

ESPI State Register: see page 220

ESPI Baud Rate High and Low Byte Registers: see page 221

### 16.4.1. ESPI Data Register

The ESPI Data Register, shown in Table 109, 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 109. 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 |
| Address | F60H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] DATA | **Data**<br>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. |

## 16.4.2. ESPI Transmit Data Command and Receive Data Buffer Control Register

The ESPI Transmit Data Command and Receive Data Buffer Control Register, shown in Table 110, provides control of the $\overline{SS}$ 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 110. ESPI Transmit Data Command and Receive Data Buffer Control Register (ESPITDCR)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | CRDR | RDFLAG | | Reserved | | | TEOF | SSV |
| Reset | 0 | 00 | | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | | R | R | R | R/W | R/W |
| Address | F61H | | | | | | | |

| Bit | Description |
|---|---|
| [7] CRDR | **Clear Receive Data Register**<br>Writing 1 to this bit is used to clear all data in receive data buffer. |
| [6:5] RDFLAG | **Receive Data Buffer Flag**<br>This bit is used to indicate how many bytes stored in receive buffer.<br>00 = 0 or 4 bytes (see RDRNE in the ESPI Status Register).<br>01 = 1 byte.<br>02 = 2 bytes.<br>03 = 3 bytes. |
| [4:2] | **Reserved**<br>These bits are reserved and must be programmed to 000. |

| Bit | Description |
|---|---|
| [1]<br>TEOF | **Transmit End of Frame**<br>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.<br>0 = The data in the Transmit Data Register is not the last character in the message.<br>1 = The data in the Transmit Data Register is the last character in the message. |
| [0]<br>SSV | **Slave Select Value**<br>When SSIO = 1, writes to this register will control the value output on the $\overline{SS}$ pin. For more details, see the SSMD field of the ESPI Mode Register on page 217. |

## 16.4.3. ESPI Control Register

The ESPI Control Register, shown in Table 111, configures the ESPI for transmit and receive operations.

**Table 111. ESPI Control Register**

| 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 |
| Address | F62H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>DIRQE | **Data Interrupt Request Enable**<br>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.<br>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.<br>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. |

| Bit | Description (Continued) |
|---|---|
| [6,0]<br>ESPIEN1,<br>ESPIEN0 | **ESPI Enable and Direction Control**<br>00 = The ESPI block is disabled. BRG can be used as a general-purpose timer by setting BRGCTL = 1.<br>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 1s. Not valid in MASTER Mode since Master must source data to drive the transfer.<br>10 = Transmit Only Mode<br>Use this setting in MASTER or SLAVE Mode when the software application is sending data but not receiving. RDRNE will not assert.<br>11 = Transmit/Receive Mode<br>Use this setting if the software application is both sending and receiving information. Both TDRE and RDRNE will be active. |
| [5]<br>BRGCTL | **Baud Rate Generator Control**<br>The function of this bit depends upon ESPIEN1,0. When ESPIEN1,0 = 00, this bit allows enabling the BRG to provide periodic interrupts.<br>**If the ESPI is disabled**<br>0 = The Baud Rate Generator timer function is disabled. Reading the Baud Rate High and Low registers returns the BRG reload value.<br>1 = The Baud Rate Generator timer function and time-out interrupt is enabled. Reading the Baud Rate High and Low registers returns the BRG Counter value.<br>**If the ESPI is enabled**<br>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.<br>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 time-out. See the SLAVE Mode Abort error description on page 211.<br>**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. Zilog recommends reading the counter using (2-byte) word reads. |
| [4]<br>PHASE | **Phase Select**<br>Sets the phase relationship of the data to the clock. For more information about operation of the PHASE bit, see the ESPI Clock Phase and Polarity Control section on page 201. |
| [3]<br>CLKPOL | **Clock Polarity**<br>0 = SCK idles Low (0).<br>1 = SCK idles High (1). |
| [2]<br>WOR | **Wire-OR (Open-Drain) Mode Enabled**<br>0 = ESPI signal pins not configured for open-drain.<br>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. |
| [1]<br>MMEN | **ESPI MASTER Mode Enable**<br>This bit controls the data I/O pin selection and SCK direction.<br>0 = Data out on MISO, data in on MOSI (used in SPI SLAVE Mode), SCK is an input.<br>1 = Data out on MOSI, data in on MISO (used in SPI MASTER Mode), SCK is an output. |

## 16.4.4. ESPI Mode Register

The ESPI Mode Register, shown in Table 112, configures the character bit width and mode of the ESPI I/O pins.

**Table 112. 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 |
| Address | F63H | | | | | | | |

| Bit | Description |
|---|---|
| [7:5]<br>SSMD | **Slave Select Mode**<br>This field selects the behavior of $\overline{SS}$ as a framing signal. For a detailed description of these modes, see Slave Select on page 200.<br>**000 = SPI Mode**<br>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 appropriate 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).<br>**001 = Loopback Mode**<br>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 $\overline{SS}$ in asserts, MISO (Slave output) is tied to MOSI (Slave input) to provide an asynchronous remote loop back (echo) function.<br>**010 = I2S Mode (Synchronous Framing with SSV)**<br>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 $I^2S$ 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. |

| Bit | Description (Continued) |
|---|---|
| [4:2]<br>NUMBITS[2:0] | **Number of Data Bits Per Character to Transfer**<br>This field contains the number of bits to shift for each character transfer. For information about valid bit positions when the character length is less than 8 bits, see the description of the ESPI Data Register on page 213.<br>000 = 8 bits<br>001 = 1 bit<br>010 = 2 bits<br>011 = 3 bits<br>100 = 4 bits<br>101 = 5 bits<br>110 = 6 bits<br>111 = 7 bits |
| [1]<br>SSIO | **Slave Select I/O**<br>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.<br>0 = $\overline{SS}$ pin configured as an input (SPI SLAVE and MULTI-MASTER modes).<br>1 = $\overline{SS}$ pin configured as an output (SPI SINGLE MASTER Mode). |
| [0]<br>SSPO | **Slave Select Polarity**<br>This bit controls the polarity of the $\overline{SS}$ pin.<br>0 = $\overline{SS}$ is active Low. (SSV = 1 corresponds to $\overline{SS}$ = 0).<br>1 = $\overline{SS}$ is active High. (SSV = 1 corresponds to $\overline{SS}$ = 1). |

## 16.4.5. ESPI Status Register

The ESPI Status Register, shown in Table 113, indicates the current state of the ESPI. All bits revert to their Reset state if the ESPI is disabled.

**Table 113. 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 |
| Address | F64H | | | | | | | |

**Note**: R/W* = Read access. Write a 1 to clear the bit to 0.

| Bit | Description |
|---|---|
| [7]<br>TDRE | **Transmit Data Register Empty**<br>0 = Transmit Data Register is full or ESPI is disabled.<br>1 = Transmit Data Register is empty. A write to the ESPI (Transmit) Data Register clears this bit. |
| [6]<br>TUND | **Transmit Underrun**<br>0 = A Transmit Underrun error has not occurred.<br>1 = A Transmit Underrun error has occurred. |
| [5]<br>COL | **Collision**<br>0 = A multi-Master collision (mode fault) has not occurred.<br>1 = A multi-Master collision (mode fault) has occurred. |
| [4]<br>ABT | **SLAVE Mode Transaction Abort**<br>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 time-out (MMEN = 0, BRGCTL = 1).<br>0 = A SLAVE Mode transaction abort has not occurred.<br>1 = A SLAVE Mode transaction abort has occurred. |
| [3]<br>ROVR | **Receive Overrun**<br>0 = A Receive Overrun error has not occurred.<br>1 = A Receive Overrun error has occurred. |
| [2]<br>RDRNE | **Receive Data Register Not Empty**<br>0 = Receive Data Register is empty.<br>1 = Receive Data Register is not empty. |

| Bit | Description (Continued) |
|---|---|
| [1]<br>TFST | **Transfer Status**<br>0 = No data transfer is currently in progress.<br>1 = Data transfer is currently in progress. |
| [0]<br>SLAS | **Slave Select**<br>Reading this bit returns the current value of the $\overline{SS}$ pin.<br>0 = The $\overline{SS}$ pin input is Low.<br>1 = The $\overline{SS}$ pin input is High. |

## 16.4.6. ESPI State Register

The ESPI State Register, shown in Table 114, lets you observe the ESPI clock, data and internal state.

**Table 114. 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 | | | | | |
| Address | F65H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>SCKI | **Serial Clock Input**<br>This bit reflects the state of the serial clock pin.<br>0 = The SCK input pin is Low.<br>1 = The SCK input pin is High. |
| [6]<br>SDI | **Serial Data Input**<br>This bit reflects the state of the serial data input (MOSI or MISO depending on the MMEN bit).<br>0 = The serial data input pin is Low.<br>1 = The serial data input pin is High. |
| [5:0]<br>ESPISTATE | **ESPI State Machine**<br>Indicates the current state of the internal ESPI State Machine. This information is intended for manufacturing test purposes. The state values may change in future hardware revisions and are not intended to be used by a software driver. |

Table 115 defines the valid ESPI states.

**Table 115. ESPISTATE Values**

| ESPISTATE Value | Description |
|---|---|
| 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 |

## 16.4.7. ESPI Baud Rate High and Low Byte Registers

The ESPI Baud Rate High and Low Byte registers, shown in Tables 116 and 117, 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:

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

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

**Table 116. 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 |
| Address | F66H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>BRH | **ESPI Baud Rate High Byte**<br>The most significant byte, BRG[15:8], of the ESPI Baud Rate Generator's reload value. |

**Table 117. 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 |
| Address | F67H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>BRL | **ESPI Baud Rate Low Byte**<br>The least significant byte, BRG[7:0], of the ESPI Baud Rate Generator's reload value. |

# Chapter 17. I²C Master/Slave Controller

The I²C Master/Slave Controller ensures that the Z8 Encore! XP F1680 Series devices are bus-compatible with the I²C protocol. The I²C bus consists of the serial data signal (SDA) and a serial clock signal (SCL) bidirectional lines. The features of I²C controller include:

- 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 I²C controller is disabled

## 17.1. Architecture

Figure 42 displays the architecture of the I²C controller.

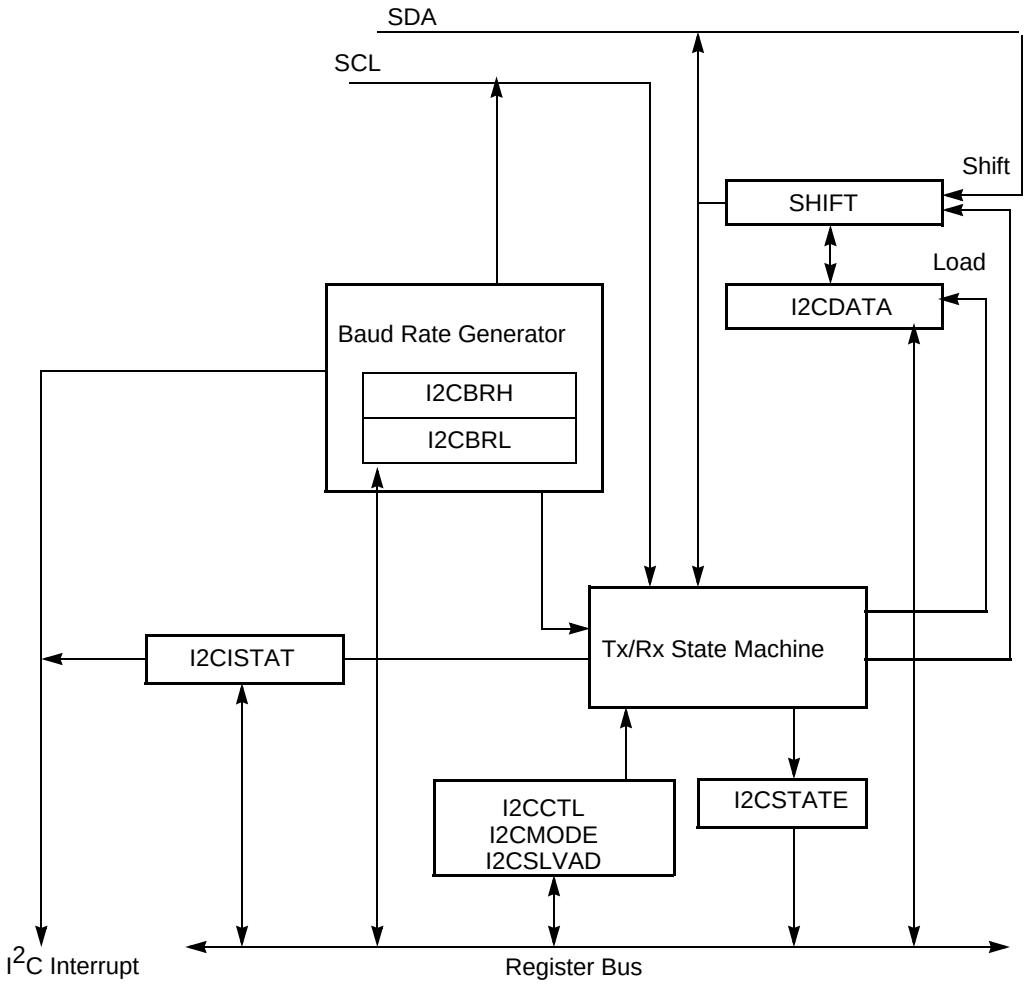



**Figure 42. I²C Controller Block Diagram**

## 17.1.1. I²C Master/Slave Controller Registers

Table 118 summarizes the I²C Master/Slave controller's software-accessible registers.

**Table 118. I²C Master/Slave Controller Registers**

| Name | Abbreviation | Description |
|---|---|---|
| I²C Data | I2CDATA | Transmit/Receive Data Register. |
| I²C Interrupt Status | I2CISTAT | Interrupt status register. |
| I²C Control | I2CCTL | Control Register—basic control functions. |

**Table 118. I²C Master/Slave Controller Registers (Continued)**

| Name | Abbreviation | Description |
|---|---|---|
| I²C Baud Rate High | I2CBRH | High byte of baud rate generator initialization value. |
| I²C Baud Rate Low | I2CBRL | Low byte of baud rate generator initialization value. |
| I²C State | I2CSTATE | State register. |
| I²C Mode | I2CMODE | Selects MASTER or SLAVE modes, 7-bit or 10-bit addressing; configure address recognition, define slave address bits [9:8]. |
| I²C Slave Address | I2CSLVAD | Defines slave address bits [7:0]. |

# 17.2. Operation

The I²C 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.

## 17.2.1. SDA and SCL Signals

The I²C circuit sends all addresses, Data and Acknowledge signals over the SDA line, with most-significant bit first. SCL is the clock for the I²C 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 I²C 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 I²C 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 I2C FAST Mode (400 Kbps) and can also be used at lower data rates.

## 17.2.2. I2C Interrupts

The I2C controller contains multiple interrupt sources that are combined into one interrupt request signal to the interrupt controller. If the I2C controller is enabled, the source of the interrupt is determined by which bits are set in the I2CISTAT Register. If the I2C 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.

### 17.2.2.1. 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 I2C Control Register.
- The I2C controller is enabled with one of the following elements:
  - 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 I2C Data Register always clears the TRDE bit to 0.

### 17.2.2.2. Receive Interrupts

Receive interrupts (RDRF bit = 1 in I2CISTAT) occur when a byte of data has been received by the I2C controller. The RDRF bit is cleared by reading from the I2C Data Register. If the RDRF interrupt is not serviced prior to the completion of the next Receive byte, the I2C 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.

### 17.2.2.3. Slave Address Match Interrupts

Slave address match interrupts (SAM bit = 1 in I2CISTAT) occur when the I2C 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 $I^2C$ 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/$\overline{W}$} and the second byte is compared against SLA[7:0].

### 17.2.2.4. Arbitration Lost Interrupts

Arbitration Lost interrupts (ARBLST bit = 1 in I2CISTAT) occur when the $I^2C$ controller is in MASTER Mode and loses arbitration (outputs 1 on SDA and receives 0 on SDA). The $I^2C$ controller switches to SLAVE Mode when this instance occurs. This bit clears automatically when the I2CISTAT Register is read.

### 17.2.2.5. Stop/Restart Interrupts

A Stop/Restart event interrupt (SPRS bit = 1 in I2CISTAT) occurs when the $I^2C$ controller is operating in SLAVE Mode and a stop or restart condition is received, indicating the end of the transaction. The RSTR bit in the $I^2C$ 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.

### 17.2.2.6. Not Acknowledge Interrupts

Not Acknowledge interrupts (NCKI bit = 1 in I2CISTAT) occur in MASTER Mode when Not Acknowledge is received or sent by the $I^2C$ controller and the start or stop bit is not set in the $I^2C$ 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 $I^2C$ controller waits until 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.

### 17.2.2.7. General Purpose Timer Interrupt from Baud Rate Generator

If the $I^2C$ 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 $I^2C$ Controller to be used as a general-purpose timer when the $I^2C$ Controller is disabled.

## 17.2.3. Start and Stop Conditions

The Master generates the start and stop conditions to start or end a transaction. To start a transaction, the $I^2C$ controller generates a start condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the $I^2C$ controller generates a stop condition by creating a Low-to-High transition of the SDA signal while the SCL signal is High. These start and stop events occur when the start and stop bits in the $I^2C$ 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.

## 17.2.4. Software Control of $I^2C$ Transactions

The $I^2C$ controller is configured via the $I^2C$ Control and $I^2C$ Mode registers. The MODE[1:0] field of the $I^2C$ Mode Register allows the configuration of the $I^2C$ 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 $I^2C$ system
- MASTER/SLAVE in a Multimaster/multislave $I^2C$ system
- SLAVE ONLY operation in an $I^2C$ system

In SLAVE ONLY mode, the start bit of the $I^2C$ 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 $I^2C$ transactions by enabling the $I^2C$ controller interrupt in the interrupt controller or by polling the $I^2C$ Status Register.

To use interrupts, the $I^2C$ interrupt must be enabled in the interrupt controller and followed by executing an EI instruction. The TXI bit in the $I^2C$ Control Register must be set to enable transmit interrupts. An $I^2C$ interrupt service routine then checks the $I^2C$ Status Register to determine the cause of the interrupt.

To control transactions by polling, the TDRE, RDRF, SAM, ARBLST, SPRS and NCKI interrupt bits in the $I^2C$ Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

## 17.2.5. Master Transactions

The following sections describe Master Read and Write transactions to both 7-bit and 10-bit slaves.

### 17.2.5.1. 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 $I^2C$ controller will not arbitrate. If a Slave address match occurs and the $I^2C$ 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 $I^2C$ controller will initiate the master transaction after the $I^2C$ bus is no longer busy.

### 17.2.5.2. 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).

### 17.2.5.3. 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

### 17.2.5.4. Master Write Transaction with a 7-Bit Address

Figure 43 displays 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**

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

1. The software initializes the MODE field in the $I^2C$ 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 $I^2C$ Control Register.
2. The software asserts the TXI bit of the $I^2C$ Control Register to enable transmit interrupts.
3. The $I^2C$ interrupt asserts, because the $I^2C$ Data Register is empty.
4. 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 $I^2C$ Data Register.
5. The software sets the start bit of the $I^2C$ Control Register.
6. The $I^2C$ controller sends a start condition to the $I^2C$ slave.
7. The $I^2C$ controller loads the $I^2C$ Shift Register with the contents of the $I^2C$ Data Register.
8. After one bit of 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 $I^2C$ Data Register.
10. The $I^2C$ controller shifts the remainder of the address and the Write bit out via the SDA signal.

11. The I²C slave sends an Acknowledge (by pulling the SDA signal Low) during the next High period of SCL. The I²C controller sets the ACK bit in the I²C Status Register.

    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.
12. The I²C controller loads the contents of the I²C Shift Register with the contents of the I²C Data Register.
13. The I²C controller shifts the data out 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 I²C 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 I²C Control Register.
17. The I²C controller completes transmission of the data on the SDA signal.
18. The I²C controller sends a stop condition to the I²C bus.

**Note:** If the slave terminates the transaction early by responding with a Not Acknowledge during the transfer, the I²C 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 I²C controller hardware automatically flushes transmit data in the not acknowledge case.

### 17.2.5.5. Master Write Transaction with a 10-Bit Address

Figure 44 displays 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/$\overline{A}$ | 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.

Observe the following steps for a master transmit operation to a 10-bit addressed slave:

1. The software initializes the MODE field in the $I^2C$ Mode Register for MASTER/SLAVE Mode with 7- or 10-bit addressing (the $I^2C$ 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 $I^2C$ Control Register.
2. The software asserts the TXI bit of the $I^2C$ Control Register to enable transmit interrupts.
3. The $I^2C$ interrupt asserts because the $I^2C$ Data Register is empty.
4. 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 $I^2C$ Control Register.
6. The $I^2C$ controller sends a start condition to the $I^2C$ Slave.
7. The $I^2C$ controller loads the $I^2C$ Shift Register with the contents of the $I^2C$ Data Register.
8. After one bit of 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 $I^2C$ Data Register.
10. The $I^2C$ controller shifts the remainder of the first byte of the address and the Write bit out via the SDA signal.
11. The $I^2C$ slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL. The $I^2C$ controller sets the ACK bit in the $I^2C$ Status Register.

    If the slave does not acknowledge the first address byte, the $I^2C$ controller sets the NCKI bit in the $I^2C$ Status Register, sets the ACKV bit and clears the ACK bit in the $I^2C$ State Register. The software responds to the Not Acknowledge interrupt by setting the stop bit and clearing the TXI bit. The $I^2C$ 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 $I^2C$ controller loads the $I^2C$ Shift Register with the contents of the $I^2C$ Data Register (2nd address byte).
13. The $I^2C$ 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 I2C Control Register.
15. The I2C 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 I2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL. The I2C controller sets the ACK bit in the I2C Status Register. If the slave does not acknowledge, see the second paragraph of Step 11.
17. The I2C 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 I2C Control Register.
20. The I2C controller completes transmission of the data on the SDA signal.
21. The I2C controller sends a stop condition to the I2C bus.

**Note:** If the slave responds with a Not Acknowledge during the transfer, the I2C controller asserts the NCKI bit, sets the ACKV bit, clears the ACK bit in the I2C 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.

### 17.2.5.6. Master Read Transaction with a 7-Bit Address

Figure 45 displays 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**

Observe the following steps for a Master Read operation to a 7-bit addressed slave:

1. The software initializes the MODE field in the I2C Mode Register for MASTER/SLAVE Mode with 7- or 10-bit addressing (the I2C 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.

### 17.2.5.7. Master Read Transaction with a 10-Bit Address

Figure 46 displays the read transaction format for a 10-bit addressed Slave.

| S | Slave Address 1st Byte | W=0 | A | Slave Address 2nd Byte | A | S | Slave Address 1st Byte | R=1 | A | Data | A | Data | $\overline{A}$ | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**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.

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

1. The software initializes the MODE field in the I2C Mode Register for MASTER/SLAVE Mode with 7- or 10-bit addressing (the I2C 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 `11110b`, followed by the two most-significant address bits and a 0 (write) to the I2C Data Register.
3. The software asserts the start bit of the I2C Control Register.
4. The I2C controller sends a start condition.
5. The I2C controller loads the I2C Shift Register with the contents of the I2C 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 I2C Data Register.
8. The I2C controller completes shifting of the first address byte.
9. The I2C 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 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 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 I2C controller loads the I2C Shift Register with the contents of the I2C Data Register (the lower byte of the 10-bit address).
11. The I2C controller shifts out the next eight bits of the address. After the first bit shifts, the I2C controller generates a transmit interrupt.
12. The software responds by setting the start bit of the I2C 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 I2C Data Register.
14. If the user chooses to read only one byte, the software responds by setting the NAK bit of the I2C Control Register.
15. After the I2C controller shifts out the address bits listed in Step 9 (the second address transfer), the I2C 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 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 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 I2C controller sends a repeated start condition.
17. The I2C controller loads the I2C Shift Register with the contents of the I2C Data Register (the third address transfer).
18. The I2C controller sends `11110b`, followed by the two most-significant bits of the slave read address and a 1 (Read).
19. The I2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.
20. The I2C controller shifts in a byte of data from the slave.
21. The I2C controller asserts the Receive interrupt.
22. 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.
23. The I2C controller sends an Acknowledge or Not Acknowledge to the I2C Slave, based on the value of the NAK bit.
24. If there are more bytes to transfer, the I2C controller returns to Step 18.
25. The I2C controller generates a NAK interrupt (the NCKI bit in the I2CISTAT Register).
26. The software responds by setting the stop bit of the I2C Control Register.
27. A stop condition is sent to the I2C Slave.

## 17.2.6. Slave Transactions

The following sections describe Read and Write transactions to the I2C controller configured for 7-bit and 10-bit Slave modes.

### 17.2.6.1. Slave Address Recognition

The following two slave address recognition options are supported; a description of each follows.

- Slave 7-Bit Address Recognition Mode
- Slave 10-Bit Address Recognition Mode

**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 $I^2C$ 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 $I^2C$ controller automatically responds during the Acknowledge phase with the value in the NAK bit of the I2CCTL Register.

### 17.2.6.2. 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 $I^2C$ 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 $I^2C$ specification.

### 17.2.6.3. 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 $I^2C$ 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.

### 17.2.6.4. 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 |
| $\overline{A}$ | Not Acknowledge |
| P | Stop |

### 17.2.6.5. 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 displayed in Figure 47. The procedure that follows describes the $I^2C$ 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/$\overline{A}$ | P/S |
|---|---|---|---|---|---|---|---|---|---|---|

**Figure 47. Data Transfer Format—Slave Receive 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 $I^2C$ 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 $I^2C$ Slave Address Register.
   d. Set IEN = 1 in the $I^2C$ Control Register. Set NAK = 0 in the $I^2C$ Control Register.
2. The bus master initiates a transfer, sending the address byte. In SLAVE Mode, the $I^2C$ controller recognizes its own address and detects that R/$\overline{W}$ bit = 0 (written from the master to the slave). The $I^2C$ 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 $I^2C$ controller holds the SCL signal Low waiting for the software to load the first data byte.
3. 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 I2C 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 I2C 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 I2C 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.

### 17.2.6.6. 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 displayed in Figure 48. The procedure that follows describes the I2C 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/$\overline{A}$ | 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 I2C Control Register.
2. The Master initiates a transfer, sending the first address byte. The I2C controller recognizes the start of a 10-bit address with a match to SLA[9:8] and detects R/$\overline{W}$ bit = 0 (a Write from the master to the slave). The I2C controller acknowledges, indicating it is available to accept the transaction.
3. The Master sends the second address byte. The SLAVE Mode I2C 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 $I^2C$ 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 $I^2C$ 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 $I^2C$ 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 $I^2C$ 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.

### 17.2.6.7. 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 displayed in Figure 49. The procedure that follows describes the $I^2C$ 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 $I^2C$ 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 $I^2C$ Slave Address Register.

d. Set IEN = 1 in the I²C Control Register. Set NAK = 0 in the I²C Control Register.

2. The Master initiates a transfer by sending the address byte. The SLAVE Mode I²C controller finds an address match and detects that the R/$\overline{W}$ bit = 1 (read by the master from the slave). The I²C 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 I²C 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 I²C 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 I²C controller holds SCL Low until the Data Register has been written. When a Not Acknowledge instruction is received by the slave, the I²C 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 I²C 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.

### 17.2.6.8. Slave Transmit Transaction With 10-Bit Address

The data transfer format for a master reading data from a slave with 10-bit addressing is displayed in Figure 50. The following procedure describes the I²C Master/Slave Controller operating as a slave in 10-bit addressing mode, transmitting data to the bus master.

| S | Slave Address 1st Byte | W = 0 | A | Slave Address 2nd Byte | A | S | Slave Address 1st Byte | R = 1 | A | Data | A | Data | A | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**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 $I^2C$ 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 $I^2C$ MODE Register.
   d. Set IEN = 1 and NAK = 0 in the $I^2C$ Control Register.
2. The Master initiates a transfer by sending the first address byte. The SLAVE Mode $I^2C$ 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 $I^2C$ controller acknowledges indicating it is available to accept the transaction.
3. The Master sends the second address byte. The SLAVE Mode $I^2C$ 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 $I^2C$ controller acknowledges on the $I^2C$ 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 $I^2C$ 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 $I^2C$ 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 $I^2C$ 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 $I^2C$ 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 I2C 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 I2C 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 I2C controller holds SCL Low until the Data Register is written.

    When a Not Acknowledge is received by the slave, the I2C 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 I2C 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.

## 17.3. I2C Control Register Definitions

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

I2C Data Register: see page 243

I2C Interrupt Status Register: see page 245

I2C Interrupt Status Register: see page 245

I2C Baud Rate High and Low Byte Registers: see page 248

I2C State Register: see page 250

I2C Mode Register: see page 253

I2C Slave Address Register: see page 255

### 17.3.1. I2C Data Register

The I2C Data Register listed in Table 119 contains the data that is to be loaded into the Shift Register to transmit onto the I2C bus. This register also contains data that is loaded

from the Shift Register after it is received from the I2C bus. The I2C 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 I2C controller is in SLAVE Mode and data is being received).

**Table 119. I2C 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 |
| Address | F50H | | | | | | | |

| Bit Position | Value | Description |
|---|---|---|
| [7:0]<br>DATA | — | I2C Data Byte |

## 17.3.2. I²C Interrupt Status Register

The read-only I²C Interrupt Status Register, shown in Table 120, indicates the cause of any current I²C interrupt and provides status of the I²C 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 120. I²C 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 |
| Address | F51H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TDRE | **Transmit Data Register Empty**<br>When the I²C controller is enabled, this bit is 1 when the I²C Data Register is empty. When set, this bit causes the I²C controller to generate an interrupt, except when the I²C controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit clears by writing to the I2CDATA Register. |
| [6]<br>RDRF | **Receive Data Register Full**<br>This bit is set = 1 when the I²C controller is enabled and the I²C controller has received a byte of data. When asserted, this bit causes the I²C controller to generate an interrupt. This bit clears by reading the I2CDATA Register. |
| [5]<br>SAM | **Slave Address Match**<br>This bit is set = 1 if the I²C 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 I²C 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. |
| [4]<br>GCA | **General Call Address**<br>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 I²C 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). |
| [3]<br>RD | **Read**<br>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. |

| Bit | Description (Continued) |
|---|---|
| [2]<br>ARBLST | **Arbitration Lost**<br>This bit is set when the $I^2C$ 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. |
| [1]<br>SPRS | **Stop/Restart Condition Interrupt**<br>This bit is set when the $I^2C$ 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. |
| [0]<br>NCKI | **NAK Interrupt**<br>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. |

## 17.3.3. I²C Control Register

The I²C Control Register, shown in Table 121, enables and configures I²C operation.

**Note:** The R/W1 bit can be set (written to 1) when IEN = 1, but cannot be cleared (written to 0).

**Table 121. I²C 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 |
| Address | F52H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>IEN | **I²C Enable**<br>This bit enables the I²C controller. |
| [6]<br>START | **Send Start Condition**<br>When set, this bit causes the I²C controller (when configured as the master) to send a start condition. After it is asserted, this bit is cleared by the I²C 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 I²C Shift Register. If there is no data in one of these registers, the I²C controller waits until data is loaded. If this bit is set while the I²C 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 until 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. |
| [5]<br>STOP | **Send Stop Condition**<br>When set, this bit causes the I²C controller (when configured as the master) to send the stop condition after the byte in the I²C Shift Register has completed transmission or after a byte is received in a receive operation. When set, this bit is reset by the I²C 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. |
| [4]<br>BIRQ | **Baud Rate Generator Interrupt Request**<br>This bit is ignored when the I²C controller is enabled. If this bit is set = 1 when the I²C 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. |

| Bit | Description (Continued) |
|---|---|
| [3]<br>TXI | **Enable TDRE Interrupts**<br>This bit enables interrupts when the I2C Data Register is empty. |
| [2]<br>NAK | **Send NAK**<br>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 IEN bit is cleared. If this bit is 1, it cannot be cleared to 0 by writing to the register. |
| [1]<br>FLUSH | **Flush Data**<br>Setting this bit clears the I2C Data Register and sets the TDRE bit to 1. This bit allows flushing of the I2C Data Register when an NAK condition is received after the next data byte is written to the I2C Data Register. Reading this bit always returns 0. |
| [0]<br>FILTEN | **I2C Signal Filter Enable**<br>Setting this bit enables low-pass digital filters on the SDA and SCL input signals. This function provides the spike suppression filter required in I2C 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. |

## 17.3.4. I2C Baud Rate High and Low Byte Registers

The I2C Baud Rate High and Low Byte registers, shown in Tables 122 and 123, combine to form a 16-bit reload value, BRG[15:0], for the I2C Baud Rate Generator. The I2C 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, use 10000H in the equation.

**Table 122. I2C Baud Rate High Byte Register (I2CBRH = 53H)**

| 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 |
| Address | F53H | | | | | | | |

| Bit Position | Value | Description |
|---|---|---|
| [7:0]<br>BRH | | **I2C Baud Rate High Byte**<br>The most significant byte, BRG[15:8], of the I2C Baud Rate Generator's reload value. |

**Note:** If the DIAG bit in the I²C Mode Register is set to 1, a read of the I2CBRH Register returns the current value of the I²C Baud Rate Counter[15:8].

**Table 123. I²C Baud Rate Low Byte Register (I2CBRL = F54H)**

| 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 |
| Address | F54H | | | | | | | |

| Bit Position | Value | Description |
|---|---|---|
| [7:0] | **I²C Baud Rate Low Byte** | |
| BRL | The least significant byte, BRG[7:0], of the I²C Baud Rate Generator's reload value. | |

**Note:** If the DIAG bit in the I²C Mode Register is set to 1, a read of the I2CBRL Register returns the current value of the I²C Baud Rate Counter[7:0].

## 17.3.5. I²C State Register

The read-only I²C State Register, shown in Table 124, provides information about the state of the I²C bus and the I²C bus controller. When the DIAG bit of the I²C Mode Register is cleared, this register provides information about the internal state of the I²C controller and I²C bus; see Table 126.

When the DIAG bit of the I²C Mode Register is set, this register returns the value of the I²C controller state machine.

**Table 124. I²C 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 |
| Address | F55H | | | | | | | |

| Bit | Description |
|---|---|
| [7:4]<br>I2CSTATE_H | **I²C State**<br>This field defines the current state of the I²C controller. It is the most significant nibble of the internal state machine. Table 126 defines the states for this field. |
| [3:0]<br>I2CSTATE_L | **Least Significant Nibble of the I²C State Machine**<br>This field defines the substates for the states defined by I2CSTATE_H. Table 127 defines the values for this field. |

**Table 125. I²C 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 |
| Address | F55H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>ACKV | **ACK Valid**<br>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. |
| [6]<br>ACK | **Acknowledge**<br>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. |
| [5]<br>AS | **Address State**<br>This bit is active High while the address is being transferred on the I²C bus. |
| [4]<br>DS | **Data State**<br>This bit is active High while the data is being transferred on the I²C bus. |
| [3]<br>10B | **10B**<br>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 after the address has been sent. |
| [2]<br>RSTR | **RESTART**<br>This bit is updated each time a stop or restart interrupt occurs (SPRS bit set in I2CISTAT Register).<br>0 = Stop condition.<br>1 = Restart condition. |
| [1]<br>SCLOUT | **Serial Clock Output**<br>Current value of Serial Clock being output onto the bus. The actual values of the SCL and SDA signals on the I²C bus can be observed via the GPIO Input Register. |
| [0]<br>BUSY | **I²C Bus Busy**<br>0 = No activity on the I²C Bus.<br>1 = A transaction is underway on the I²C bus. |

**Table 126. I2CSTATE_H**

| State Encoding | State Name | State Description |
|---|---|---|
| 0000 | Idle | $I^2C$ bus is idle or $I^2C$ controller is disabled. |
| 0001 | Slave Start | $I^2C$ 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 127. 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. |

Table 127. I2CSTATE_L (Continued)

| State I2CSTATE_H | Substate I2CSTATE_L | Substate Name | State Description |
|---|---|---|---|
| 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. |

## 17.3.6. I²C Mode Register

The I²C Mode Register, shown in Table 128, provides control over master versus slave operating mode, slave address and diagnostic modes.

Table 128. I²C 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 |
| Address | F56H | | | | | | | |

| Bit | Description |
|---|---|
| [7] | Reserved; must be 0. |
| [6:5] MODE[1:0] | **Selects the I²C Controller Operational Mode**<br>00 = MASTER/SLAVE capable (supports multi-master arbitration) with 7-bit Slave address.<br>01 = MASTER/SLAVE capable (supports multi-master arbitration) with 10-bit slave address.<br>10 = Slave Only capable with 7-bit address.<br>11 = Slave Only capable with 10-bit address. |

| Bit | Description (Continued) |
|---|---|
| [4]<br>IRM | **Interactive Receive Mode**<br>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.<br>0 = Acknowledge occurs automatically and is determined by the value of the NAK bit of the I2CCTL Register.<br>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. |
| [3]<br>GCE | **General Call Address Enable**<br>Enables reception of messages beginning with the General Call Address or start byte.<br>0 = Do not accept a message with the General Call Address or start byte.<br>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 $I^2C$ Status Register indicates whether the address matched the General Call Address/start byte or not. Following the General Call Address byte, the software can set the IRM bit that allows software to examine the following data byte(s) before acknowledging. |
| [2:1]<br>SLA[9:8] | **Slave Address Bits 9 and 8**<br>Initialize with the appropriate slave address value when using 10-bit slave addressing. These bits are ignored when using 7-bit slave addressing. |
| [0]<br>DIAG | **Diagnostic Mode**<br>Selects read back value of the Baud Rate Reload and State registers.<br>0 = Reading the Baud Rate registers returns the Baud Rate register values. Reading the State register returns $I^2C$ controller state information.<br>1 = Reading the Baud Rate registers returns the current value of the baud rate counter. Reading the State register returns additional state information. |

## 17.3.7. I²C Slave Address Register

The I²C Slave Address Register, shown in Table 129, provides control over the lower order address bits used in 7 and 10 bit slave address recognition.

**Table 129. I²C Slave Address Register (I2CSLVAD = 57H)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SLA[7:0] | | | | | | | |
| Reset | 00H | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F57H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>SLA[7:0] | **Slave Address Bits**<br>Initialize with the appropriate Slave address value. When using 7-bit Slave addressing, SLA[9:7] are ignored. |

# Chapter 18. Comparator

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 trigger timer counting
- Output can be either an interrupt source or an output to an external pin
- Operation in STOP Mode

## 18.1. Operation

One of the comparator inputs can be connected to an internal reference which is a user-selectable reference that is user-programmable with 200mV resolution.

The comparator can be powered down to save supply current or to continue to operate in STOP Mode. For details, see the Power Control Register 0 section on page 44. In STOP Mode, the comparator interrupt (if enabled) automatically initiates a Stop Mode Recovery and generates an interrupt request. In the Reset Status Register (see page 40), the stop bit is set to 1. Also, the Comparator request bit in the Interrupt Request 1 Register (see page 74) is set. Following completion of the Stop Mode Recovery, and 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, spurious interrupts can result after enabling the comparator. Zilog recommends not enabling the comparator without first disabling interrupts, then waiting for the comparator output to settle.

The following code 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
```

## 18.2. Comparator Control Register Definitions

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

Comparator 0 Control Register: see page 257

Comparator 1 Control Register: see page 258

### 18.2.1. Comparator 0 Control Register

The Comparator 0 Control Register (CMP0), shown in Table 130, configures the Comparator 0 inputs and sets the value of the internal voltage reference.

**Table 130. 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 |
| Address | F90H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>INPSEL | **Signal Select for Positive Input**<br>0 = GPIO pin used as positive comparator 0 input.<br>1 = Temperature sensor used as positive comparator 0 input. |
| [6]<br>INNSEL | **Signal Select for Negative Input**<br>0 = Internal reference disabled, GPIO pin used as negative comparator 0 input.<br>1 = Internal reference enabled as negative comparator 0 input. |

| Bit | Description (Continued) |
|---|---|
| [5:2]<br>REFLVL | **Comparator 0 Internal Reference Voltage Level**<br>This reference is independent of the ADC voltage reference.<br>0000 = 0.0 V<br>0001 = 0.2 V<br>0010 = 0.4 V<br>0011 = 0.6 V<br>0100 = 0.8 V<br>0101 = 1.0 V (Default)<br>0110 = 1.2 V<br>0111 = 1.4 V<br>1000 = 1.6 V<br>1001 = 1.8 V<br>1010–1111 = Reserved |
| [1:0]<br>TIMTRG | **Timer Trigger (COMPARATOR COUNTER MODE)**<br>00 = Disable Timer Trigger.<br>01 = Comparator 0 output works as Timer 0 Trigger.<br>10 = Comparator 0 output works as Timer 1 Trigger.<br>11 = Comparator 0 output works as Timer 2 Trigger. |

## 18.2.2. Comparator 1 Control Register

The Comparator 1 Control Register (CMP1), shown in Table 131, configures the comparator 1 inputs and sets the value of the internal voltage reference.

**Table 131. 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 |
| Address | F91H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>INPSEL | **Signal Select for Positive Input**<br>0 = GPIO pin used as positive comparator 1 input.<br>1 = Temperature sensor used as positive comparator 1 input. |
| [6]<br>INNSEL | **Signal Select for Negative Input**<br>0 = Internal reference disabled, GPIO pin used as negative comparator 1 input.<br>1 = Internal reference enabled as negative comparator 1 input. |

| Bit | Description (Continued) |
|---|---|
| [5:2]<br>REFLVL | **Comparator 1 Internal Reference Voltage Level**<br>This reference is independent of the ADC voltage reference.<br>0000 = 0.0 V<br>0001 = 0.2 V<br>0010 = 0.4 V<br>0011 = 0.6 V<br>0100 = 0.8 V<br>0101 = 1.0 V (Default)<br>0110 = 1.2 V<br>0111 = 1.4 V<br>1000 = 1.6 V<br>1001 = 1.8 V<br>1010–1111 = Reserved |
| [1:0]<br>TIMTRG | **Timer Trigger (Comparator Counter Mode)**<br>Enable/disable timer operation.<br>00 = Disable Timer Trigger.<br>01 = Comparator 1 output works as Timer 0 Trigger.<br>10 = Comparator 1 output works as Timer 1 Trigger.<br>11 = Comparator 1 output works as Timer 2 Trigger. |

# Chapter 19. Temperature Sensor

The on-chip Temperature Sensor allows you to measure temperature on the die to an accuracy of roughly ±7°C over a range of –40°C to +105°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
- ±7°C full-range accuracy for calibrated version
- ±1.5°C accuracy over the range of 20°C to 30°C
- Flash recalibration capability

## 19.1. Operation

The on-chip temperature sensor is a Proportional To Absolute Temperature (PTAT) topology which provides 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 (see page 44) 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 retrimming 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 two equations define the relationship between the ADC reading and the die temperature. In each equation, T is the temperature in degrees Celsius, and ADC is the 10-bit compensated ADC value.

**Equation #1.** If bit 2 of TEMPCALH calibration option byte is 0, then:

T = (25/128) * (ADC + {TEMPCALH_bit1, TEMPCALH_bit0, TEMPCALL}) – 77

**Equation #2.** If bit 2 of TEMPCALH calibration option byte is 1, then:

T = (25/128) * (ADC – {TEMPCALH_bit1, TEMPCALH_bit0, TEMPCALL}) –77

In these two equations, TEMPCALH and TEMPCALL are a pair of Flash option bits containing the calibration data. For more details, see the discussion of TEMPCALH and TEMPCALL in the Flash Option Bits chapter on page 276.

**Note:** The equations above are temporary test results of the Z8F1680 MCU, version A. The coefficient in the formula may change according to results from tests of version B.

### 19.1.1. 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, Zilog recommends that the user accounts for the difference between ambient and die temperatures when inferring ambient temperature conditions.

# Chapter 20. Flash Memory

The products in the Z8 Encore! XP F1680 Series feature either 24KB (24576 bytes), 16KB (16384 bytes) and 8KB (8192 bytes) of nonvolatile 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, Flash memory is also divided into sectors. In the Z8 Encore! XP F1680 Series, 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 about their operation, see the Flash Option Bits chapter on page 276.

Table 132 lists the Flash memory configuration for each device in the Z8 Encore! XP F1680 Series.

**Table 132. 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 |

## 20.1. 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 in the Flash Information Area, and so are factory calibration data for the Temperature Sensor. Figures 51 through 53 display the Flash memory arrangement.

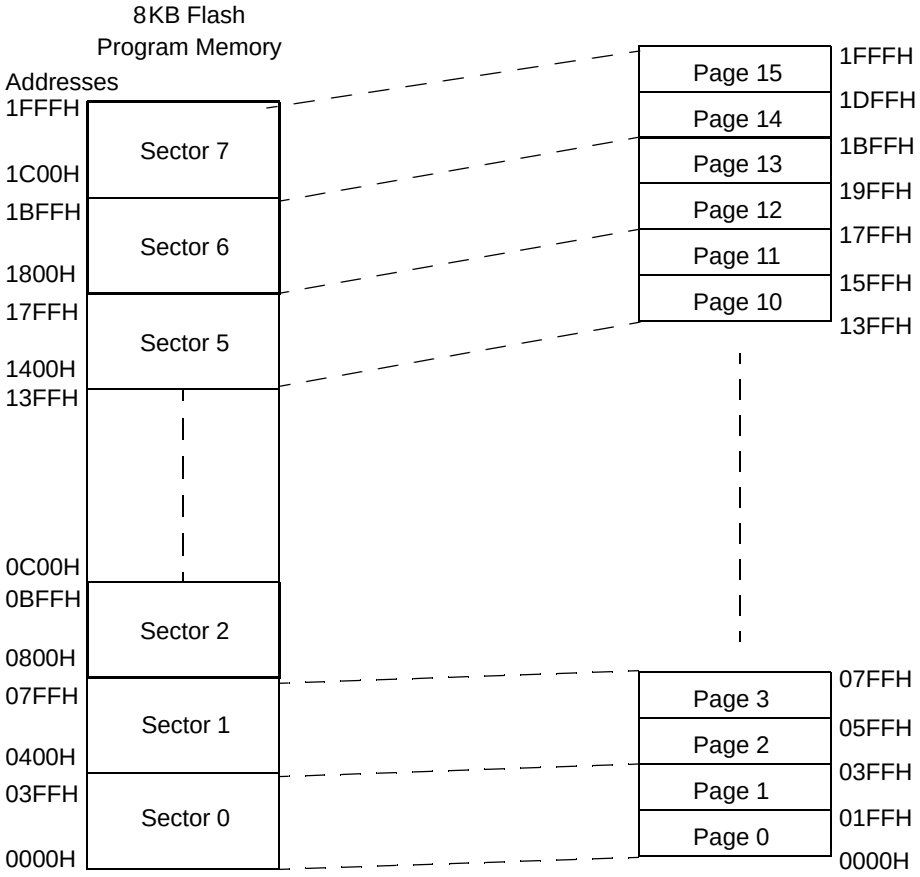



**Figure 51. 8KB Flash Memory Arrangement**

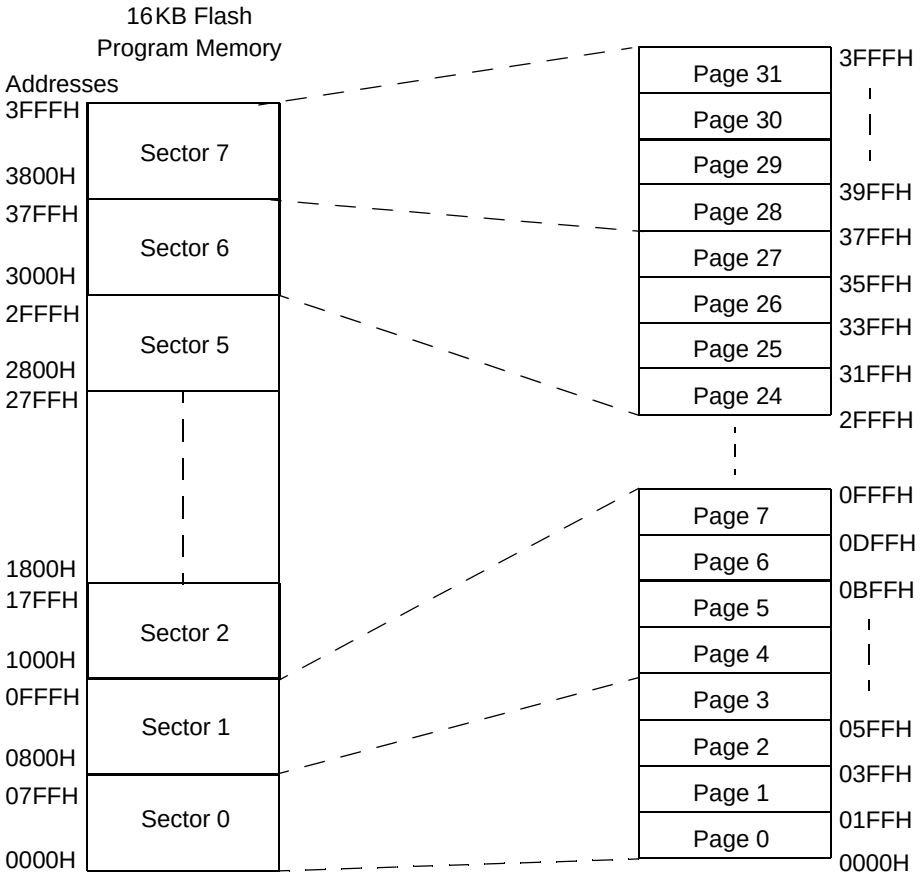



**Figure 52. 16KB Flash Memory Arrangement**

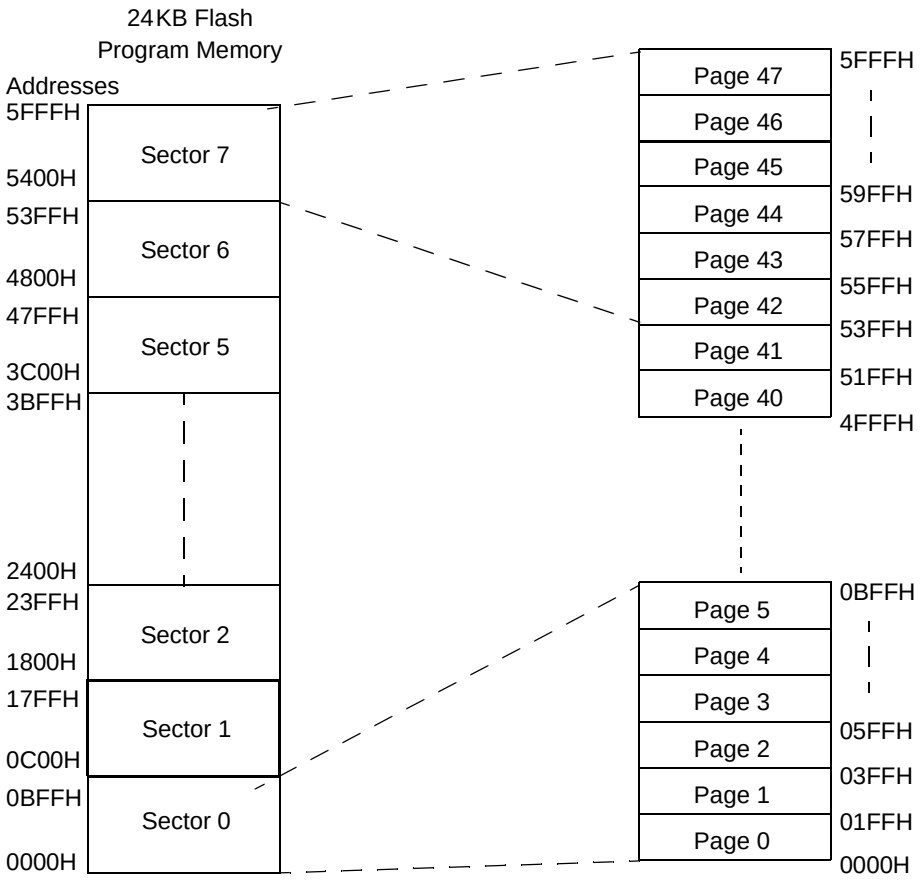



**Figure 53. 24KB Flash Memory Arrangement**

## 20.2. 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 displays basic Flash Controller operation. The sections that follow provide details about the various operations (Lock, Unlock, Byte Programming, Page Protect, Page Unprotect, Page Select Page Erase and Mass Erase) shown in Figure 54.

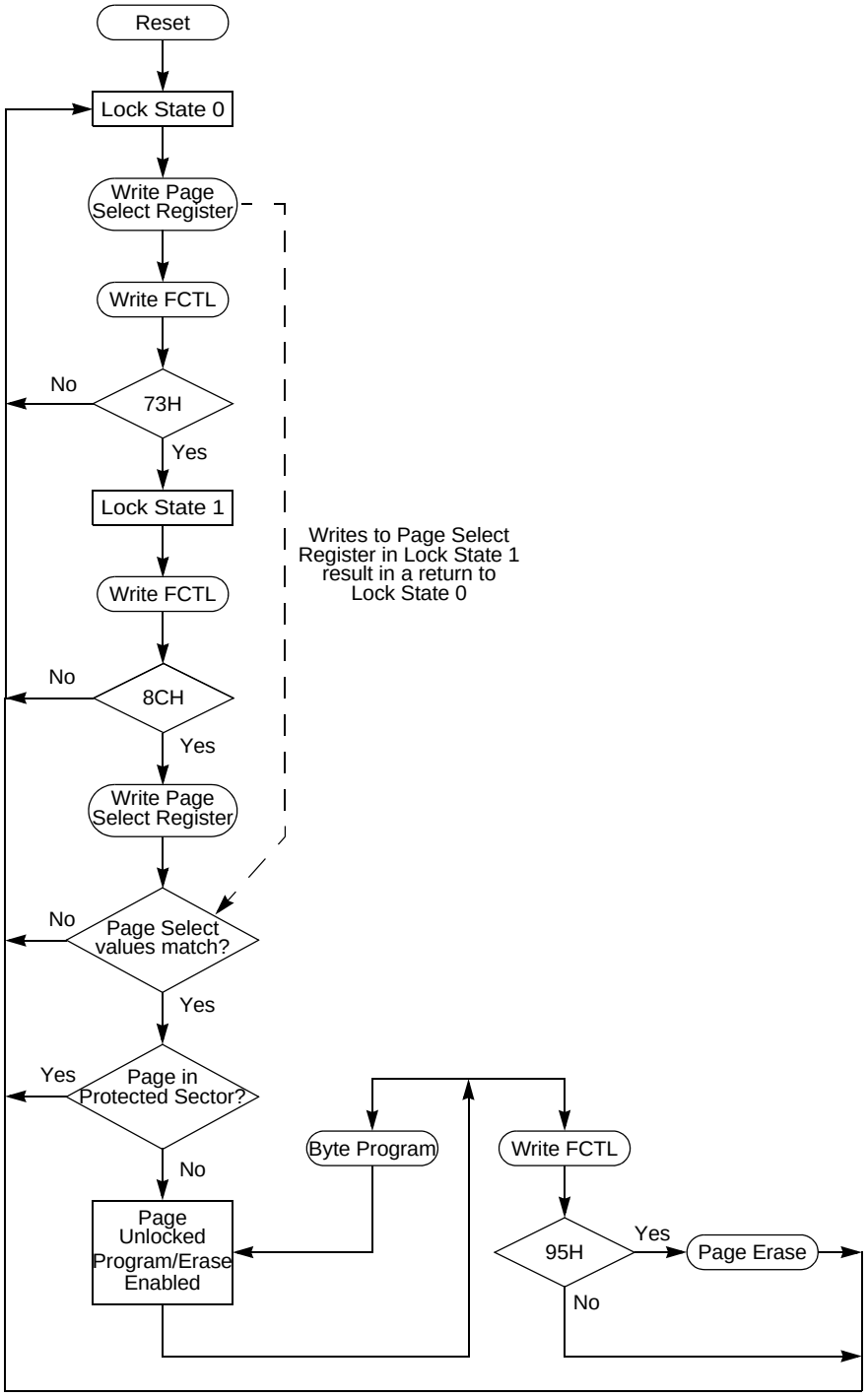



**Figure 54. Flowchart: Flash Controller Operation**

### 20.2.1. 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.

### 20.2.2. Flash Code Protection Against External Access

The user code contained within 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 the Flash Option Bits chapter on page 276 and the On-Chip Debugger chapter on page 294.

### 20.2.3. 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 contents of Flash memory. 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.

#### 20.2.3.1. Flash Code Protection Using the Flash Option Bits

The FWP Flash option bit provides Flash Program Memory protection as listed in Table 133. For more details, see the Flash Option Bits chapter on page 276.

**Table 133. 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. |

### 20.2.3.2. Flash Code Protection Using the Flash Controller

At Reset, the Flash Controller locks to prevent accidental program or erasure of the contents of Flash memory. Follow the steps below to unlock the Flash Controller from user code:

1. Write the Page Select Register with the target page.
2. Write the first unlock command 73H to the Flash Control Register.
3. Write the second unlock command 8CH to the Flash Control Register.
4. Rewrite the Page Select Register 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 the flowchart in Figure 54 on page 266.

**Note:** Byte Programming, Page Erase and Mass Erase will not be allowed if the FWP bit is cleared or if the page resides in a protected block.

After unlocking a specific page, Byte Programing or Page Erase can be performed. At the conclusion of a Page Erase, the Flash Controller is automatically locked. To lock the Flash Controller after Byte Programming, write to the Flash Control Register with any value other than the Page Erase or Mass Erase comands.

### 20.2.3.3. 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 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 3KB, 2KB or 1KB depending on available on-chip Flash size of 24KB, 16KB and 8KB.

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

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

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

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 a System Reset.

## 20.2.4. Byte Programming

Flash memory is enabled for byte programming on the active page after unlocking the Flash Controller. Erase the address(es) to be programmed using either the Page Erase or Mass Erase command prior to performing byte programming. An erased Flash byte contains all 1s (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 command.

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 the eZ8 CPU Core User Manual (UM0128), available for download at www.zilog.com. While the Flash Controller programs the contents of Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate.

After a byte is written, the page remains unlocked, allowing for subsequent writes to other bytes on the same page. To exit programming mode and lock Flash memory, write any value to the Flash Control Register except the Mass Erase or Page Erase commands.

**Caution:** The byte at each Flash memory address cannot be programmed (any bits written to 0) more than twice before an erase cycle occurs.

## 20.2.5. Page Erase

The Flash memory can be erased one page (512 bytes) at a time. Page-erasing 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, writing the value 95h to the Flash Control Register initiates the Page Erase operation on the active page. 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.

## 20.2.6. Mass Erase

Flash memory can also be mass-erased using the Flash Controller, but only by using the On-Chip Debugger. Mass-erasing Flash memory sets all bytes to the value FFH. With the Flash Controller unlocked, 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.

## 20.2.7. Flash Controller Bypass

The Flash Controller can be bypassed and the control signals for Flash memory are brought out to the GPIO pins. Bypassing the Flash Controller allows faster row programming algorithms by controlling these 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 Flash memory. Mass Erase and Page Erase operations are also supported when the Flash Controller is bypassed.

For more information about bypassing the Flash Controller, please contact Zilog Technical Support.

### 20.2.8. 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, the 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.

## 20.3. Flash Control Register Definitions

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

Flash Control Register: see page 271

Flash Status Register: see page 272

Flash Page Select Register: see page 273

Flash Sector Protect Register: see page 274

Flash Frequency High and Low Byte Registers: see page 274

### 20.3.1. Flash Control Register

The Flash Controller must be unlocked using the Flash Control Register (see Table 134) before programming or erasing Flash memory. The Flash Controller is unlocked by writing to the Flash Page Select Register, then 73H 8CH, sequentially, to the Flash Control Register, and finally again to the Flash Page Select Register with the same value as the previous write. When the Flash Controller is unlocked, Mass Erase or Page Erase can be initiated by writing the appropriate command to the FCTL. Erase applies only to the active page selected in the 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 134. 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 |
| Address | FF8H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] FCMD | **Flash Command**<br>73H = First unlock command.<br>8CH = Second unlock command.<br>95H = Page Erase command (must be third command in sequence to initiate Page Erase).<br>63H = Mass Erase command (must be third command in sequence to initiate Mass Erase).<br>5EH = Enable Flash Sector Protect Register Access |

## 20.3.2. Flash Status Register

The Flash Status register (Table 135) 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 135. 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 |
| Address | FF8H | | | | | | | |

| Bit | Description |
|---|---|
| [7:6] Program_ status | **Indicate the fail or success after Flash Write/Erase**<br>00 = Success.<br>10 = Success.<br>11 = Fail due to low power.<br>01 = Reserved. |

| Bit | Description (Continued) |
|---|---|
| [5:0] FSTAT | **Flash Controller Status**<br>000000 = Flash Controller locked.<br>000001 = First unlock command received (73H written).<br>000010 = Second unlock command received (8CH written).<br>000011 = Flash Controller unlocked.<br>000100 = Sector protect register selected.<br>001xxx = Program operation in progress.<br>010xxx = Page erase operation in progress.<br>100xxx = Mass erase operation in progress. |

## 20.3.3. Flash Page Select Register

The Flash Page Select Register, shown in Table 136, shares address space with the Flash Sector Protect Register. Unless the Flash controller was last 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 provided by FPS[6:0] are chosen for program/erase operation.

**Table 136. 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 |
| Address | FF9H | | | | | | | |

| Bit | Description |
|---|---|
| [7] INFO_EN | **Information Area Enable**<br>0 = Information Area is not selected.<br>1 = Information Area is selected. The Information Area is mapped into the Program Memory address space at addresses FE00H through FFFFH. |
| [6:0] PAGE | **Page Select**<br>This 7-bit field identifies the Flash memory page for Page Erase and page unlocking. Program Memory address[15:9] = PAGE[6:0].<br>• On Z8F2480 devices, the upper 1 bit must always be 0.<br>• On Z8F1680 devices, the upper 2 bits must always be 0.<br>• On Z8F0880 devices, the upper 3 bits must always be 0. |

## 20.3.4. Flash Sector Protect Register

The Flash Sector Protect Register is shared with the Flash Page Select Register. When the Flash Control Register (see page 271) is written with 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 can only be unprotected (the register bit can only be cleared) by a System Reset. Please refer to Table 132 on page 262 and to Figures 51 through 53 to review how Flash memory is arranged by sector.

**Table 137. 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 |
| Address | FF9H | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] SPROT*x* | **Sector Protection**<br>• On Z8F2480 devices, each bit corresponds to a 3KB Flash sector.<br>• On Z8F1680 devices, each bit corresponds to a 2KB Flash sector.<br>• On Z8F0880 devices, each bit corresponds to a 1KB Flash sector. |

## 20.3.5. Flash Frequency High and Low Byte Registers

The Flash Frequency High and Low Byte registers, shown in Tables 138 and 139, 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:

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

**Caution:** Flash programming and erasure is not supported for system clock frequencies below 32 kHz or above 20 MHz. The Flash Frequency High and Low Byte registers must be loaded with the correct values to ensure proper operation of the device.

**Table 138. 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 |
| Address | FFAH | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] FFREQH | **Flash Frequency High Byte**<br>High byte of the 16-bit Flash Frequency value. |

**Table 139. Flash Frequency Low Byte Register (FFREQL)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | FFREQL | | | | | | | |
| Reset | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FFBH | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] FFREQL | **Flash Frequency Low Byte**<br>Low byte of the 16-bit Flash Frequency value. |

# Chapter 21. Flash Option Bits

Programmable Flash option bits allow user configuration of certain aspects of Z8 Encore! XP F1680 Series MCU operation. The feature configuration data is stored in Flash program memory and are read during Reset. The features available for control through the Flash option bits include:

- Watchdog Timer time-out 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
- LVD voltage threshold selection
- Oscillator mode selection for high, medium and low-power crystal oscillators or an external RC oscillator
- Factory trimming information for the IPO and Temperature Sensor

## 21.1. Operation

This section describes the types of option bits and their configuration in the Option Configuration registers.

### 21.1.1. 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 Flash Program Memory and written to the Option Configuration registers. These Option Configuration registers control operation of the devices within the Z8 Encore! XP F1680 Series MCU. 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.

## 21.1.2. Option Bit Types

This section describes the User, Trim and Calibration option bit types.

### 21.1.2.1. 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.

### 21.1.2.2. Trim Option Bits

The trim option bits are contained in the Flash memory information page. 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 via the trim bit address and data registers.

### 21.1.2.3. 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 the Flash Information Area section on page 21.

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
```

## 21.2. Flash Option Bit Control Register Definitions

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

User Option Bits: see page 278

Trim Bit Data Option Bits: see page 281

Trim Bit Address Option Bits: see page 281

Trim Bit Address Space: see page 282

Zilog Calibration Option Bits: see page 289

### 21.2.1. User Option Bits

The first two bytes of Flash program memory, at addresses 0000H and 0001H, are reserved for the user-programmable Flash option bits, as shown in Tables 140 and 141.

**Table 140. 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 |
| Address | Program Memory 0000H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7]<br>WDT_RES | **Watchdog Timer Reset**<br>0 = Watchdog Timer time-out generates an interrupt request. Interrupts must be globally enabled for the eZ8 CPU to acknowledge the interrupt request.<br>1 = Watchdog Timer time-out causes a System Reset. This setting is the default for unprogrammed (erased) Flash. |
| [6]<br>WDT_AO | **Watchdog Timer Always ON**<br>0 = Watchdog Timer is automatically enabled upon application of system power. Watchdog Timer cannot be disabled.<br>1 = Watchdog Timer is enabled upon execution of the WDT instruction. After it is enabled, the Watchdog Timer can only be disabled by a Reset or Stop Mode Recovery. This setting is the default for unprogrammed (erased) Flash. |

| Bit | Description (Continued) |
|---|---|
| [5:4]<br>OSC_SEL[1:0] | **Oscillator Mode Selection**<br>00 = On-chip oscillator configured for use with external RC networks or external clock input (<4MHz).<br>01 = Reserved.<br>10 = Medium power for use with medium frequency crystals or ceramic resonators (1.0MHz to 8.0MHz).<br>11 = Maximum power for use with high-frequency crystals (8.0MHz to 20.0MHz). This setting is default for unprogrammed (erased) Flash. |
| [3]<br>VBO_AO | **Voltage Brown-Out Protection Always ON**<br>0 = Voltage Brown-Out 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.<br>1 = Voltage Brown-Out 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. |
| [2]<br>FRP | **Flash Read Protect**<br>0 = User program code is inaccessible. Limited control features are available through the On-Chip Debugger.<br>1 = User program code is accessible. All On-Chip Debugger commands are enabled. This setting is default for unprogrammed (erased) Flash. |
| [1]<br>PRAM_M | **On-Chip Program RAM Mode Select**<br>0 = Program RAM is used as on-chip Register RAM and it begins at the first available Register File address space. See the Register Map chapter on page 23.<br>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. |
| [0]<br>FWP | **Flash Write Protect**<br>This option bit provides Flash program memory protection:<br>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.<br>1 = Programming, Page Erase and Mass Erase are enabled for all of Flash program memory. |

**Table 141. 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 |
| Address | Program Memory 0001H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7:6]<br>EXTLTMG | **External Crystal Reset Timing**<br>00 = 500 Internal Precision Oscillator Cycles.<br>01 = 1000 Internal Precision Oscillator Cycles.<br>10 = 5000 Internal Precision Oscillator Cycles.<br>11 = 10,000 Internal Precision Oscillator Cycles. |
| [5]<br>FLASH_WR_PRO_EN | **Flash Write Operation Protect**<br>0 = Flash write protect disable.<br>1 = Flash write protect with internal LVD. |
| [4]<br>EXTL_AO | **External Crystal Always ON**<br>0 = Crystal oscillator is enabled during Reset, resulting in longer reset timing.<br>1 = Crystal oscillator is disabled during Reset, resulting in shorter reset timing.<br>Note: This bit determines the state of the external crystal oscillator at Reset. Its selection as system clock must be performed in the Oscillator Control Register (OSCCTL0). |
| [3:2] | Reserved; must be 00. |
| [1]<br>X2_Mode | **Secondary Crystal Mode Select**<br>0 = External clock input.<br>1 = External 32kHz watch crystal. |
| [0]<br>X2TL_AO | **Secondary Crystal Always ON**<br>0 = Secondary Crystal Oscillator is enabled during reset.<br>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 performed in the Peripheral Control (for example, Timer Control2) register.

## 21.2.2. Trim Bit Data Option Bits

The Trim Bit Data Register, shown in Table 142, contains the read or write data for access to the trim option bits.

**Table 142. 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 |
| Address | FF7H | | | | | | | |

## 21.2.3. Trim Bit Address Option Bits

The Trim Bit Address Register, shown in Table 143, contains the target address for access to the trim option bits. Trim Bit addresses in the range 00H-1FH map to the Information Area address range 20H-3FH, as indicated in Table 142.

**Table 143. 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 |
| Address | FF6H | | | | | | | |

**Table 144. Trim Bit Address Map**

| Trim Bit Address | Information Area Address |
|---|---|
| 00H | 20H |
| 01H | 21H |
| 02H | 22H |
| 03H | 23H |
| : | : |
| 1FH | 3FH |

## 21.2.4. Trim Bit Address Space

All available Trim Bit addresses and their functions are summarized in Table 145. For details about each, see Tables 146 through 156.

**Table 145. Trim Bit Address Description**

| Address | Function |
|---|---|
| 0000H | Temperature Sensor Trim0 |
| 0001H | Temperature Sensor Trim1 |
| 0002H | Internal Precision Oscillator |
| 0003H | VBO and LVD |
| 0004H | ADC and Comparator 0/1 |
| 0005H | ADC Reference Voltage |
| 0006H | 20-M Oscillator and 32-K Oscillator |
| 0007H | Reserved |
| 0008H | Reserved |

### 21.2.4.1. Trim Bit Addresses 0000H and 0001H

The Trim Option Bits registers at addresses 0000H and 00001H, shown in Tables 146 and 147, govern control of the temperature sensor trim bits.

**Table 146. Trim Option Bits at Address 0000H (TTEMP0)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TS_FINE | | | | Reserved | TS_ULTRAFINE | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Address | Information Page Memory 0020H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7:4]<br>TS_FINE | **Temperature Sensor Fine Control Trim Bits**<br>Contains fine control offset trimming bits for the Temperature Sensor. |
| [3] | Reserved; must be 1. |
| [2:0]<br>TS_ULTRAFINE | **Temperature Sensor Ultra Fine Control Trim Bits**<br>Contains ultra-fine control offset trimming bits for the Temperature Sensor. |

**Table 147. Trim Option Bits at 0001H (TTEMP1)**

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

| Bit | Description |
|---|---|
| [7:4]<br>TS_NEG | **Temperature Sensor Negative Control Trim Bits**<br>Negative control offset trimming bits for the Temperature Sensor. |
| [3:0]<br>TS_COARSE | **Temperature Sensor Coarse Control Trim Bits**<br>Contains coarse control offset trimming bits for the Temperature Sensor. |

### 21.2.4.2. Trim Bit Address 0002H

The Trim Option Bits Register at address 0002H, shown in Table 148, governs control of the Internal Precision Oscillator trim bits.

**Table 148. Trim Option Bits at 0002H (TIPO)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | IPO_TRIM | | | | | | | |
| Reset | U | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | Information Page Memory 0022H | | | | | | | |
| Note: U = Unchanged by Reset. R/W = Read/Write. | | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>IPO_TRIM | **Internal Precision Oscillator Trim Byte**<br>Contains trimming bits for Internal Precision Oscillator. |

### 21.2.4.3. Trim Bit Address 0003H

The Trim Option Bits Register at address 0003H, shown in Table 149, governs control of the Voltage Brown-Out and Low Voltage Detect trim bits.

**Table 149. 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 |
| Address | Information Page Memory 0023H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7:4]<br>VBO_TRIM | **Reserved**<br>These bits are reserved and must be programmed to 0000. |
| [3:0]<br>LVD_TRIM | **Low Voltage Detect Trim**<br>This trimming affects the low-voltage detection threshold. Each LSB represents a 50mV change in the threshold level. Alternatively, the low voltage threshold can be computed from the options bit value by the following equation:<br>LVD_LVL = 3.2 V – LVD_TRIM * 0.05 V<br>Typical LVD_TRIM values are listed in Table 150. |

**Table 150. LVD_Trim Values**

| | LVD Threshold (V) | | | |
|---|---|---|---|---|
| LVD_TRIM | Minimum | Typical | Maximum | Description |
| 00000 | | 3.20 | | Maximum LVD threshold |
| 00001 | | 3.15 | | |
| 00010 | | 3.10 | | |
| 00011 | | 3.05 | | |
| 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 | | |

**Table 150. LVD_Trim Values (Continued)**

| | LVD Threshold (V) | | | |
|---|---|---|---|---|
| **LVD_TRIM** | **Minimum** | **Typical** | **Maximum** | **Description** |
| 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. |

### 21.2.4.4. Trim Bit Address 0004H

The Trim Option Bits Register at address 0004H, shown in Table 151, governs control of the Temperature Sensor Test, Comparator, and ADC trim bits.

**Table 151. Trim Option Bits at 0004H (TCOMP_ADC)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEMPTST | 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 |
| Address | Information Page Memory 0024H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7]<br>TEMPTST | **Temperature Sensor Test Control Bit**<br>The default is 1. |
| [6]<br>COMP1_OPT | **Comparator 1 Trim Bit**<br>This COMP1_OPT bit controls the comparator's *hys* input; see Table 152. |
| [5] | Reserved; must be 0. |
| [4]<br>COMP0_OPT | **Comparator 0 Trim Bit**<br>This COMP0_OPT bit controls the comparator's *hys* input; see Table 152. |
| [3:0]<br>ADC_OPT | **ADC Trim Values**<br>Contains factory trimmed values for the ADC block. |

**Table 152. Truth Table of HYS**

| HYS | Hysteresis Input |
|---|---|
| 1 | Enable |
| 0 | Disable |

### 21.2.4.5. Trim Bit Address 0005H

In the Trim Option Bits Register at address 0005H and shown in Table 153, all bits are reserved.

**Table 153. Trim Option Bits at 0005H (TVREF)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | 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 |
| Address | Information Page Memory 0025H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7:5] | Reserved; must be 1. |
| [4:0] | Reserved; must be 1. |

### 21.2.4.6. Trim Bit Address 0006H

The Trim Option Bits Register at address 0006H, shown in Table 154, governs crystal oscillator trim signals.

**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 |
| Address | Information Page Memory 0026H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7:4] X1_TRIM | 4-bit trimming signal for the 20M crystal oscillator. |
| [3:0] X0_TRIM | 4-bit trimming signal for the 32K second crystal oscillator. |

### 21.2.4.7. Trim Bit Address 0007H

In the Trim Option Bits Register at address 0007H and shown in Table 155, all bits are reserved.

**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 |
| Address | Information Page Memory 0027H | | | | | | | |

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

### 21.2.4.8. Trim Bit Address 0008H

In the Trim Option Bits Register at address 0008H and shown in Table 156, all bits are reserved.

**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 |
| Address | Information Page Memory 0028H | | | | | | | |

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

## 21.2.5. Zilog Calibration Option Bits

This section describes the calibration of the temperature sensor's Low and High bytes.

### 21.2.5.1. Temperature Sensor Calibration High and Low Byte Registers

Tables 157 and 158 present the Temperature Sensor Calibration High and Low Byte registers.

**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 |
| Address | Information Page Memory FE60H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7:0] TEMPCALH | **Temperature Sensor Calibration High Byte**<br>Bits [7:3] of this register are not used. Bit 2 indicates whether the calibration data is added to or subtracted from the measured ADC data. If bit 2 is 0, the calibration data is added; if bit 2 is 1, the calibration data is subtracted.<br>Bits 1 and 0 are the High two bits of the 10-bit calibration data. |

**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 |
| Address | Information Page Memory FE61H | | | | | | | |

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

| Bit | Description |
|---|---|
| [7:0] TEMPCALL | **Temperature Sensor Calibration Low Byte**<br>TEMPCALL is the low eight bits of 10-bit calibration data. The entire 10-bit calibration data field is {TEMPCALH[1:0], TEMPCALL}. |

# Chapter 22. Nonvolatile Data Storage

The Z8 Encore! XP F1680 Series devices contain a Nonvolatile Data Storage (NVDS) element of up to 256 bytes. This memory can perform over 100,000 write cycles.

## 22.1. Operation

The NVDS is implemented by special purpose Zilog® software stored in areas of Program Memory not accessible to you. These special-purpose routines use 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.

## 22.2. 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 predefined 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 the Flash Operation Timing Using Flash Frequency Registers section on page 267.

## 22.2.1. 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 subroutine, the write status byte resides in working register R0. The bit fields of this status byte are defined in Table 159. 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 nonuniform execution time. In general, a write takes 136µs (assuming a 20MHz system clock). For every 200 writes, however, a maintenance operation is necessary. In this rare occurrence, the write takes up to 58ms 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 7µs execution time.

**Table 159. 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 |

| Bit | Description |
|---|---|
| [7:3] | Reserved; must be 0. |
| [2]<br>FE | **Flash Error**<br>If Flash error is detected, this bit is set to 1. |
| [1]<br>IGADDR | **Illegal Address**<br>When NVDS byte writes to invalid addresses (those exceeding the NVDS array size) occur, this bit is set to 1.<br>Note: When the NVDS array size is 256 bytes, there is no address exceeding the size; therefore the IGADDR bit cannot be used. |
| [0]<br>WE | **Write Error**<br>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. |

## 22.2.2. 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 subroutine, 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 160. 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 nonuniform execution time. A read operation takes between 71 µs and 258 µs (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 µs execution time. The status byte returned by the NVDS read routine is zero for successful read. If the status byte is nonzero, 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 160. 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 |

| Bit | Description |
|---|---|
| [7:5] | Reserved; must be 0. |
| [4]<br>DE | **Data Error**<br>When reading a NVDS address, if an error is found in the latest data corresponding to the NVDS address, this bit is set to 1. NVDS source code steps forward until it finds valid data at this address. |
| [3] | Reserved; must be 0. |
| [2]<br>FE | **Flash Error**<br>If a Flash error is detected, this bit is set to 1. |
| [1]<br>IGADDR | **Illegal Address**<br>When NVDS byte reads occur from invalid addresses (those exceeding the NVDS array size), this bit is set to 1.<br>Note: When the NVDS array size is 256 bytes, there is no address exceeding the size: therefore the IGADDR bit cannot be used. |
| [0] | Reserved; must be 0. |

## 22.2.3. 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 the Low-Power

Modes section on page 42) and configured for a threshold voltage of 2.4V or greater (see the Trim Bit Address Space section on page 282).

A System Reset that occurs during a write operation (such as a pin reset or watchdog timer reset) also perturbs the byte currently being written. All other bytes in the array remain unperturbed.

### 22.2.4. Optimizing NVDS Memory Usage for Execution Speed

As listed in Table 161, 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 0.8µs up to a maximum of 258µs.

**Table 161. NVDS Read Time**

| Operation | Minimum Latency (µs) | Maximum Latency (µs) |
|---|---|---|
| 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.

- Periodically refresh all addresses that are used; the most useful method. The optimal use of NVDS, in terms of speed, is to rotate the writes evenly among all planned addresses, 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.
- Use as few unique addresses as possible to optimize the impact of refreshing.

# Chapter 23. On-Chip Debugger

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

## 23.1. Architecture

The OCD consists of four primary functional blocks:

- Transmitter
- Receiver
- Autobaud detector/generator
- Debug controller

Figure 55 displays the architecture of the On-Chip Debugger.

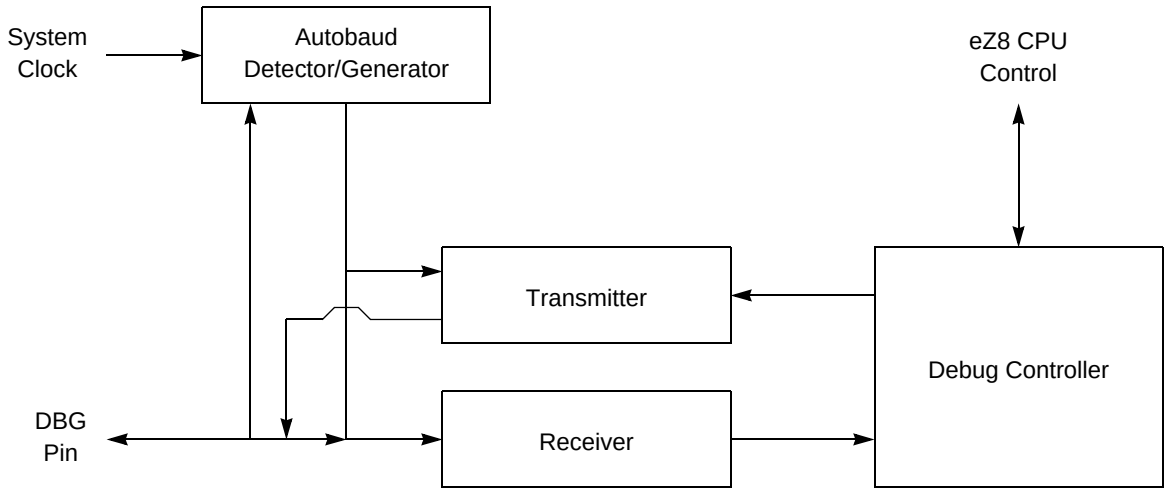



**Figure 55. On-Chip Debugger Block Diagram**

## 23.2. Operation of the 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 bidirectional open-drain interface that transmits and receives data. Data transmission is half-duplex, in that transmit and receive cannot occur simultaneously. The serial data on the DBG pin is sent using the standard asynchronous data format defined in RS-232. This pin interfaces the Z8 Encore! XP F1680 Series device to the serial port of a host PC using minimal external hardware. Figure 56 displays the connections between the debug connector and the Z8 Encore! microcontroller. Two different methods for connecting the DBG pin to an RS-232 interface are depicted in Figures 57 and  and 58.

**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.

The Serial Smart Cable (SSC) does not work with the F1680 device series because it does not fully support the silicon OCD. During external clock switching, the OCD sends a break command to the SSC. This causes the SSC to disconnect from the target and terminate the debug session. You must then reconnect to the target again. Use the Opto-Isolated USB, USB, or Ethernet Smart Cables when using in conjunction with ZDSII.

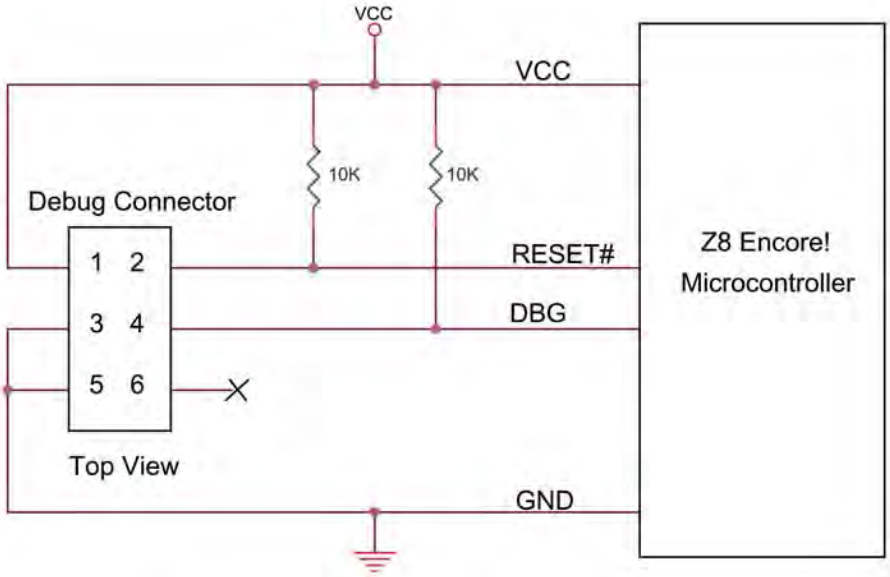



**Figure 56. Target OCD Connector Interface**

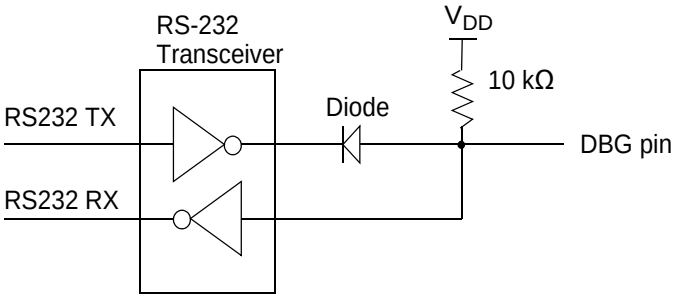



**Figure 57. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #1 of 2**

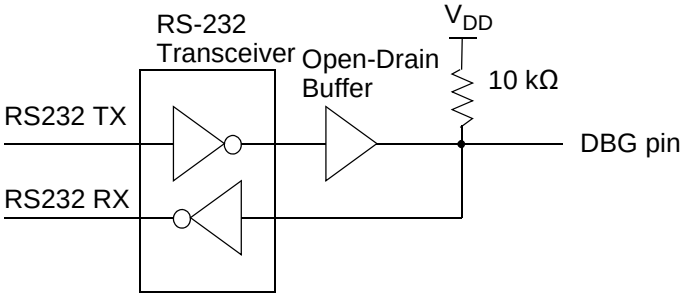



**Figure 58. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #2 of 2**

## 23.2.1. 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

### 23.2.1.1. 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 breakpoint (BRK) 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

### 23.2.1.2. 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 Brown-Out 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

## 23.2.2. 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 59).

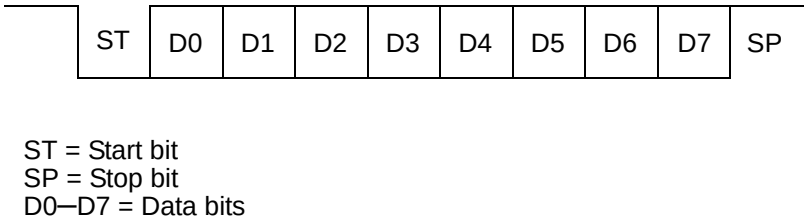



**Figure 59. OCD Data Format**

## 23.2.3. OCD Autobaud Detector/Generator

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

The Autobaud Detector/Generator is clocked by the system clock. The minimum baud rate is the system clock frequency divided by 512. 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 162 lists the minimum and recommended maximum baud rates for sample crystal frequencies.

**Table 162. 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 |
| 1.0MHz | 125 k | 1.96K |
| 32kHz | 4096 | 64 |

If the OCD receives a Serial Break (ten or more continuous bits Low), the Autobaud Detector/Generator resets. The Autobaud Detector/Generator can then be reconfigured by sending 80H. If the Autobaud Detector overflows while measuring the Autobaud character, the Autobaud Detector will remain reset.

## 23.2.4. 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 Autobaud to the appropriate speed. The Autobaud generator will automatically run at the appropriate 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 displayed in Figure 60.

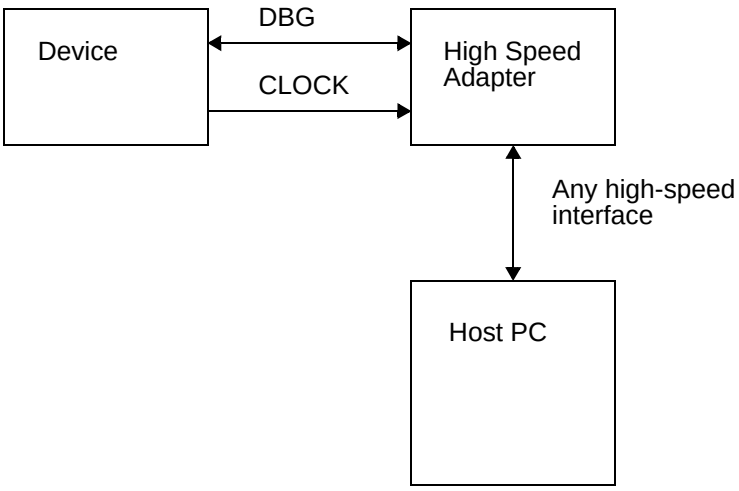



**Figure 60. Synchronous Operation**

## 23.2.5. 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 Autobaud Detector/Generator. A Framing Error or Transmit Collision can be caused by the host sending a Serial Break to the OCD. Because of the open-drain nature of the interface, returning a Serial Break back to the host only extends the length of the Serial Break if the host releases the Serial Break early.

The host transmits a Serial Break on the DBG pin when first connecting to theZ8 Encore! XP F1680 Series device or when recovering from an error. A Serial Break from the host resets the Autobaud Generator/Detector but does not reset the OCD Control Register. A Serial Break leaves the device in DEBUG mode if that is the current mode. The OCD is held in Reset until the end of the Serial Break when the DBG pin returns High. Because of the open-drain nature of the DBG pin, the host can send a Serial Break to the OCD even if the OCD is transmitting a character.

## 23.2.6. Automatic Reset

The Z8 Encore! XP F1680 Series devices have the capability to switch clock sources during operation. If the Autobaud is set and the clock source is switched, the Autobaud value becomes invalid. A new Autobaud 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 Autobaud is set, the device will automatically send a Serial Break for 4096 clocks. This will reset the Autobaud and indicate to the host that a new Autobaud character should be sent.

## 23.2.7. 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; see Figure 61.

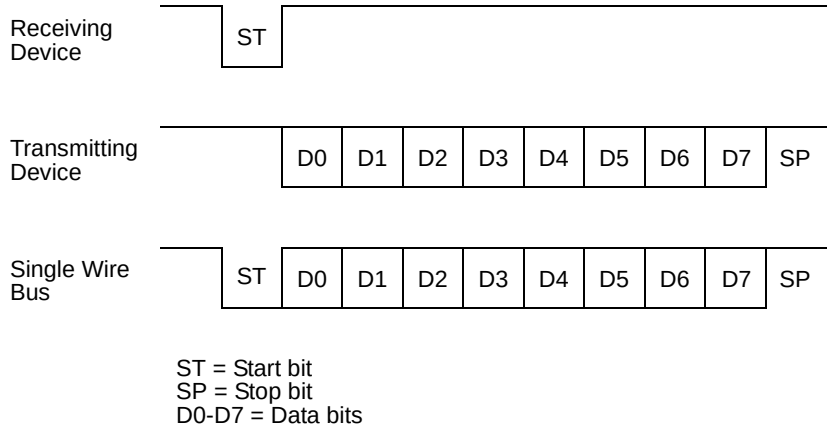



**Figure 61. Start Bit Flow Control**

## 23.2.8. Breakpoints

Execution breakpoints are generated using the BRK instruction (op code 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 a 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. Interrupts are typically disabled during critical sections of code where interrupts do not occur (such as adjusting the stack pointer or modifying shared data).

Through the OCD, host debugger software can poll the IDLE bit of the OCDSTAT Register to determine if the OCD is looping on a BRK instruction. When the host must stop the CPU on the BRK instruction on which it is looping, the host must not set the DBGMODE bit of the OCDCTL register. The CPU may have vectored to an interrupt service routine. Instead, the host clears the BRKLOOP bit, thereby allowing the CPU to finish the interrupt service routine 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.

#### 23.2.8.1. Breakpoints in Flash Memory

The BRK instruction is op code 00H, which corresponds to the fully programmed state of a byte in Flash memory. To implement a breakpoint, write 00H to the appropriate address, overwriting the current instruction. To remove a breakpoint, erase the corresponding page of Flash memory and reprogram with the original data.

### 23.2.9. OCDCNTR 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 exits 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 after the On-Chip Debugger exits 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.

## 23.3. 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, asserting the TESTMODE pad does NOT put the Z8 Encore! XP F1680 Series MCU in Flash Test mode.

Table 163 contains a summary of the On-Chip Debugger commands; each is further described 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 163. 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 | – | – |

Note: Unlisted command byte values are reserved.

**Table 163. 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 returned from the On-Chip Debugger 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 → ~OCDCNTR[15:8]
DBG → ~OCDCNTR[7:0]
```

**Write OCD Control Register (04H).** The Write OCD Control Register command writes the data that follows to the OCDCTL register.

```
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 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 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 op code. If the device is not in DEBUG mode or the Read Protect option bit is enabled, the OCD ignores this command.

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

**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]
```

## 23.4. On-Chip Debugger Control Register Definitions

This section defines the features of the following On-Chip Debugger Control registers.

OCD Control Register: see page 310

OCD Status Register: see page 312

Line Control Register: see page 313

Baud Reload Register: see page 314

## 23.4.1. OCD Control Register

The OCD Control Register, shown in Table 164, 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 164. OCD Control Register (OCDCTL)**

| Bit | 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 |

| Bit | Description |
|---|---|
| [7]<br>DBGMODE | **DEBUG Mode**<br>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.<br>0 = The device is running (operating in NORMAL Mode).<br>1 = The device is in DEBUG mode. |
| [6]<br>BRKEN | **Breakpoint Enable**<br>This bit controls the behavior of BRK instruction (op code 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.<br>0 = BRK instruction is disabled.<br>1 = BRK instruction is enabled. |
| [5]<br>DBGACK | **Debug Acknowledge**<br>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.<br>0 = Debug Acknowledge is disabled.<br>1 = Debug Acknowledge is enabled. |
| [4]<br>BRKLOOP | **Breakpoint Loop**<br>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.<br>0 = BRK instruction sets DBGMODE to 1.<br>1 = eZ8 CPU loops on BRK instruction. |

| Bit | Description |
|---|---|
| [3]<br>BRKPC | **Break when PC == OCDCNTR**<br>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.<br>0 = OCDCNTR is set up as a counter.<br>1 = OCDCNTR generates a hardware break when PC == OCDCNTR. |
| [2]<br>BRKZRO | **Break when OCDCNTR == 0000H**<br>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 exits DEBUG Mode.<br>0 = OCD does not generate BRK when OCDCNTR decrements to 0000H.<br>1 = OCD sets DBGMODE to 1 when OCDCNTR decrements to 0000H. |
| [1] | Reserved; must be 0. |
| [0]<br>RST | **Reset**<br>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.<br>0 = No effect.<br>1 = Reset the device. |

## 23.4.2. OCD Status Register

The OCD Status Register, shown in Table 165, reports status information about the current state of the debugger and the system.

**Table 165. OCD Status Register (OCDSTAT)**

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

| Bit | Description |
|---|---|
| [7]<br>IDLE | **CPU Idle**<br>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.<br>0 = The eZ8 CPU is running.<br>1 = The eZ8 CPU is either stopped or looping on a BRK instruction. |
| [6]<br>HALT | **HALT Mode**<br>0 = The device is not in HALT Mode.<br>1 = The device is in HALT Mode. |
| [5]<br>RPEN | **Read Protect Option Bit Enable**<br>0 = The Read Protect option bit is disabled (Flash option bit is 1).<br>1 = The Read Protect option bit is enabled (Flash option bit is 0), disabling many OCD commands. |
| [4:0] | Reserved; must be 0. |

## 23.4.3. Line Control Register

The Line Control Register, shown in Table 166, is used to configure the output driver characteristics during transmission. This register is only used in high-speed implementations.

**Table 166. OCD Line Control Register (OCDLCR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reset | | NBTX | NBEN | TXFC | TXDH | TXD | TXHD |
| Reset | 00 | | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Description |
|---|---|
| [7:6]<br>Reset | **Reset** |
| [5]<br>NBTX | **Nine Bit Transmit**<br>This control bit sets the polarity of the ninth bit when nine bit mode is enabled.<br>0 = Ninth bit is zero.<br>1 = Ninth bit is one. |
| [4]<br>NBEN | **Nine Bit Enable**<br>This control bit enables nine-bit mode; it is useful when transmit flow control using remote start bit is enabled to detect valid characters.<br>0 = Nine Bit mode disabled.<br>1 = Nine Bit mode enabled. |
| [3]<br>TXFC | **Transmit Flow Control**<br>0 = Transmit Flow Control disabled.<br>1 = Transmit Flow Control using remote start bit. |
| [2]<br>TXDH | **Transmit Drive High**<br>0 = Pin is not driven High during 0 to 1 transitions.<br>1 = Pin is driven High during 0 to 1 transitions. |
| [1]<br>TXD | **Transmit Drive**<br>0 = Pin is only driven Low during transmission (Open-Drain).<br>1 = Pin is always driven during transmission. |
| [0]<br>TXHD | **Transmit High Drive Strength**<br>0 = Pin output driver is Low drive strength.<br>1 = Pin output driver is High drive strength. |

## 23.4.4. Baud Reload Register

The Baud Reload Register, shown in Table 167, contains the measured Autobaud value.

**Table 167. Baud Reload Register**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | RELOAD | | | | | | | | | | | |
| Reset | 0H | | | | 000H | | | | | | | | | | | |
| R/W | R | | | | R | | | | | | | | | | | |

| Bit | Description |
|---|---|
| [15:12] | Reserved; must be 000. |
| [11:0] RELOAD | **Baud Reload Value**<br>This value is the measured Autobaud value. Its value can be calculated using the formula:<br>$RELOAD = \frac{SYSCLK}{BAUDRATE} \times 8$ |

# Chapter 24. Oscillator Control

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 32kHz watch crystal

## 24.1. Operation

This chapter discusses the logic used to select the system clock and handle primary oscillator failures. Descriptions of the specific operations of each oscillator are outlined elsewhere in this document. A detailed description of the Watchdog Timer oscillator starts on page 140, the Internal Precision Oscillator description starts on page 327 and the chapter outlining the Crystal Oscillator begins on page 321.

### 24.1.1. System Clock Selection

The oscillator control block is selected from the available clocks. Table 168 details each clock source and its usage.

**Table 168. Oscillator Configuration and Selection**

| Clock Source | Characteristics | Required Setup |
|---|---|---|
| Internal Precision RC Oscillator | • Selectable frequency 0.0432MHz, 0.0864MHz, 0.3456MHz, 0.6912MHz, 1.3824MHz, 2.7648MHz, 5.5296MHz and 11.0592MHz<br>• ± 4% accuracy when trimmed<br>• No external components required | • Unlock and write Oscillator Control Register (OSCCTL0) to enable and select oscillator frequency. |
| External Crystal/ Resonator | • 32kHz to 20MHz<br>• Very high accuracy (dependent on crystal or resonator used)<br>• External components required | • Configure Flash option bits for correct external oscillator mode<br>• 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 deasserted, no waiting is required) |
| External RC Oscillator | • 32kHz to 4MHz<br>• Accuracy dependent on external components | • Configure Flash option bits for correct external oscillator mode<br>• Unlock and write OSCCTL0 to enable crystal oscillator and select as system clock |
| External Clock Drive | • 0 to 20MHz<br>• Accuracy dependent on external clock source | • Write GPIO registers to configure PB3 pin for external clock function<br>• Unlock and write OSCCTL0 to select external system clock<br>• Apply external clock signal to GPIO |
| Internal WDT Oscillator | • 10kHz nominal<br>• ± 40% accuracy; no external components required<br>• Low power consumption | • Enable WDT if not enabled and wait until WDT Oscillator is operating.<br>• Unlock and write Oscillator Control Register (OSCCTL0) to enable and select oscillator |

**Caution:** Unintentional accesses to the Oscillator Control Register can actually stop the chip by switching to a nonfunctioning oscillator. To prevent this condition, the oscillator control block employs a register unlocking/locking scheme.

#### 24.1.1.1. OSC Control Register Unlocking/Locking

To write to the Oscillator Control Register (OSCCTL0 and OSCCTL1), unlock it by making two writes to the OSCCTLx Register with the value E7H followed by the value 18H. A third write to the OSCCTLx Register changes the value of the actual register and returns it 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 con-

secutive. 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.

## 24.1.2. Clock Failure Detection and Recovery

Clock failure detection and recovery are provided for the primary oscillator. Clock failure detection is provided for the Watchdog Timer oscillator.

### 24.1.2.1. Primary Oscillator Failure

The Z8 Encore! XP F1680 Series devices can generate nonmaskable 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 the Watchdog Timer chapter on page 140.

The primary oscillator failure detection circuitry asserts if the system clock frequency drops below 1 kHz ±50%. If an external signal is selected as the system oscillator, it is possible that a very slow but nonfailing clock can generate a failure condition. Under these conditions, do not enable the primary oscillator failure circuitry (i.e., clear the POFEN bit).

### 24.1.2.2. Watchdog Timer Failure

In the event of a Watchdog Timer oscillator failure, a similar nonmaskable 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 clearing 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.

## 24.2. Peripheral Clock

The peripheral clock is based on a low-frequency/low-power 32kHz 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 169 summarizes peripheral clock source features and usage.

**Table 169. Peripheral Clock Source and Usage**

| Peripheral Clock Source | Characteristics | Required Setup |
|---|---|---|
| Secondary Oscillator | • Optimized for use with a 32kHz Watch Crystal<br>• Very high accuracy<br>• Dedicated XTAL pins<br>• No external components | • Unlock and write OSCCTL1 to enable the secondary oscillator<br>• Select the peripheral clock at the timer clock source in the TxCTL2 Register |

## 24.3. Oscillator Control Register Definitions

The Oscillator Control registers enable and disable the various oscillator circuits, enable and disable the failure detection and recovery circuitry, and select the primary oscillator, which becomes the system clock.

### 24.3.1. Oscillator Control 0 Register

The Oscillator Control 0 Register (OSCCTL0) must be unlocked before writing. Unlock the Oscillator Control 0 Register by writing the two-step sequence of E7H followed by

18H. The register becomes locked upon successful completion of a register write to the OSCCTL0.

**Table 170. Oscillator Control 0 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 |
| Address | F86H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>INTEN | **Internal Precision Oscillator Enable**<br>1 = Internal precision oscillator is enabled.<br>0 = Internal precision oscillator is disabled. |
| [6]<br>XTLEN | **Crystal Oscillator Enable**<br>This setting overrides the GPIO register control for PA0 and PA1.<br>1 = Crystal oscillator is enabled.<br>0 = Crystal oscillator is disabled. |
| [5]<br>WDTEN | **Watchdog Timer Oscillator Enable**<br>1 = Watchdog Timer oscillator is enabled.<br>0 = Watchdog Timer oscillator is disabled. |
| [4]<br>POFEN | **Primary Oscillator Failure Detection Enable**<br>1 = Failure detection and recovery of primary oscillator is enabled.<br>0 = Failure detection and recovery of primary oscillator is disabled. |
| [3]<br>WDFEN | **Watchdog Timer Oscillator Failure Detection Enable**<br>1 = Failure detection of Watchdog Timer oscillator is enabled.<br>0 = Failure detection of Watchdog Timer oscillator is disabled. |
| [2:0]<br>SCKSEL | **System Clock Oscillator Select**<br>000 = Internal precision oscillator functions as system clock.<br>001 = Reserved.<br>010 = Crystal oscillator or external RC oscillator functions as system clock.<br>011 = Watchdog Timer oscillator functions as system.<br>100 = External clock signal on PB3 functions as system clock.<br>101 = Reserved.<br>110 = Reserved.<br>111 = Reserved. |

Note: The INTEN bit should be disabled when you use another clock as a system clock.

## 24.3.2. Oscillator Control1 Register

The Oscillator Control 1 Register (OSCCTL1) enables/disables the secondary oscillator circuits, which become 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. Unlock the Oscillator Control 1 Register by writing the two-step sequence of E7H followed by 18H. This register becomes locked upon successful completion of a register write to the OSCCTL1.

**Table 171. Oscillator Control 1 Register (OSCCTL1)**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SECEN | SECRDY | Reserved | | | INTSEL | | |
| Reset | 0 | 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 |
| Address | F87H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>SECEN | **Secondary Oscillator Enable**<br>1 = 32 kHz Secondary Oscillator is enabled.<br>0 = 32 kHz Secondary Oscillator is disabled. |
| [6]<br>SECRDY | **Secondary Oscillator Ready Flag**<br>1 = 32 kHz Secondary Oscillator is stable and running.<br>0 = 32 kHz Secondary Oscillator is not running. |
| [5:3] | Reserved; must be 0. |
| [2:0]<br>INTSEL | **Internal Precision Oscillator Frequency Select**<br>000 = Internal Precision Oscillator Frequency is 11.0592 MHz.<br>001 = Internal Precision Oscillator Frequency is 5.5296 MHz.<br>010 = Internal Precision Oscillator Frequency is 2.7648 MHz.<br>011 = Internal Precision Oscillator Frequency is 1.3824 MHz.<br>100 = Internal Precision Oscillator Frequency is 0.6912 MHz.<br>101 = Internal Precision Oscillator Frequency is 0.3456 MHz.<br>110 = Internal Precision Oscillator Frequency is 0.0864 MHz.<br>111 = Internal Precision Oscillator Frequency is 0.0432 MHz. |

# Chapter 25. Crystal Oscillator

The products in the Z8 Encore! XP F1680 Series contain a primary on-chip crystal oscillator for use with external crystals with 1MHz to 20MHz frequencies, plus a secondary 32K crystal oscillator. In addition, the external oscillator supports external RC networks with oscillation frequencies up to 4MHz. The 32K secondary crystal oscillator does not feature an 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. Additionally, the secondary 32K crystal oscillator can only be used to generate a clock for three timers.

Alternatively, the $X_{IN}$ and $X2_{IN}$ input pins can also accept a CMOS-level clock input signal (for $X_{IN}$, the frequency range is 32kHz–20MHz; for $X2_{IN}$, the range is below 4MHz). If an external clock generator is used, the $X_{OUT}$ or $X2_{OUT}$ pin must remain 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; the frequency of the signal on the $X2_{IN}$ determines the frequency of the timers.

**Note:** Although the $X_{IN}$ pin can be used as a primary system clock input for an external clock generator, the CLKIN pin is better suited for such use. For details, see the System Clock Selection section on page 315.

## 25.1. Operating Modes

The primary on-chip crystal oscillator supports three oscillator modes:

- Medium power for use with medium frequency crystals or ceramic resonators (1MHz to 8MHz)
- Maximum power for use with high-frequency crystals (8MHz to 20MHz)
- On-chip oscillator configured for use with external RC networks or external clock input (<4MHz)

The primary on-chip crystal oscillator mode is selected using user-programmable Flash option bits. For information, see the Flash Option Bits section on page 276. The secondary 32kHz crystal oscillator supports two oscillator modes:

- NORMAL Mode for use with 32kHz crystals
- On-chip oscillator configured for use with external clock input

The secondary 32K crystal oscillator mode is selected using user-programmable Flash option bits. For information, see the Flash Option Bits section on page 276.

## 25.2. Main Crystal Oscillator Operation

The Flash Option bit XTLDIS controls whether the crystal oscillator is enabled during reset. The crystal can 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 period completes, the crystal oscillator can be selected as the system clock.

Figure 62 displays the recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 20MHz. Recommended 20MHz crystal specifications are provided in Table 172. Printed circuit board layout must add no more than 4pF of stray capacitance to either the $X_{IN}$ or $X_{OUT}$ pins. If oscillation does not occur, reduce the values of capacitors $C_1$ and $C_2$ to decrease loading.

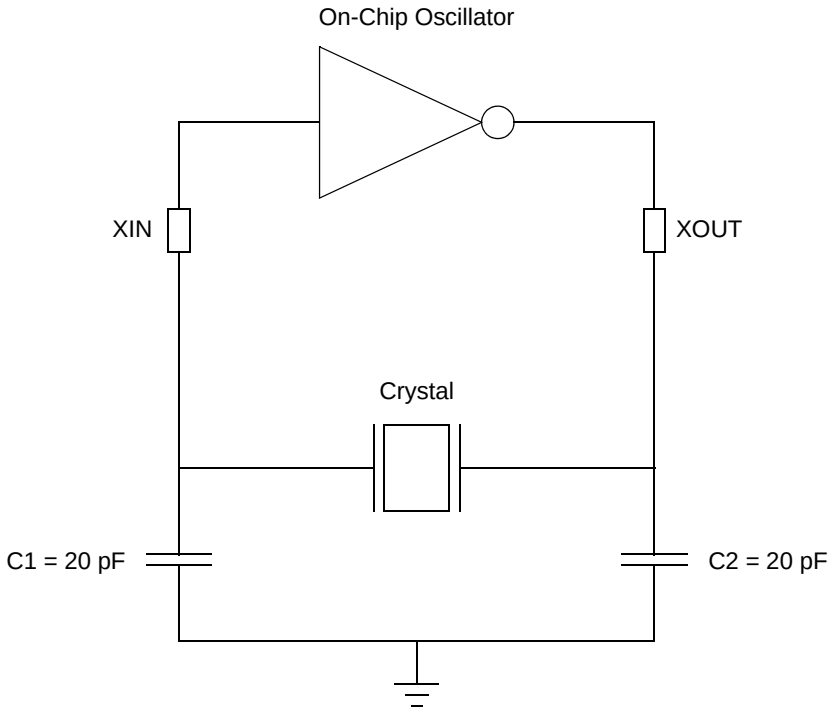



**Figure 62. Recommended 20MHz Crystal Oscillator Configuration**

**Table 172. Recommended Crystal Oscillator Specifications**

| Parameter | Value | Units | Comments |
|---|---|---|---|
| Frequency | 20 | MHz | |
| Resonance | Parallel | | |
| Mode | Fundamental | | |
| Series Resistance ($R_S$) | 60 | W | Maximum |
| Load Capacitance ($C_L$) | 30 | pF | Maximum |
| Shunt Capacitance ($C_0$) | 7 | pF | Maximum |
| Drive Level | 1 | mW | Maximum |

## 25.3. Main Oscillator Operation with External RC Network

Figure 63 displays the recommended configuration for connection with an external resistor-capacitor (RC) network.

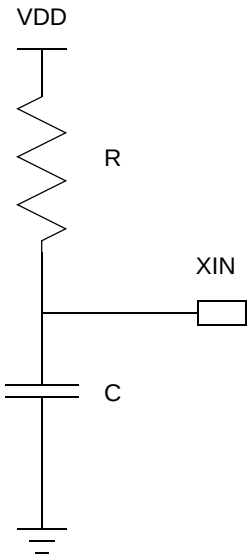



**Figure 63. Connecting the On-Chip Oscillator to an External RC Network**

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

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

Figure 64 displays the typical (3.3V and 25°C) oscillator frequency as a function of the capacitor (*C* in pF) employed in the RC network assuming a 45KΩ external resistor. For very small values of C, the parasitic capacitance of the oscillator $X_{IN}$ pin and the printed circuit board 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 20pF are recommended.

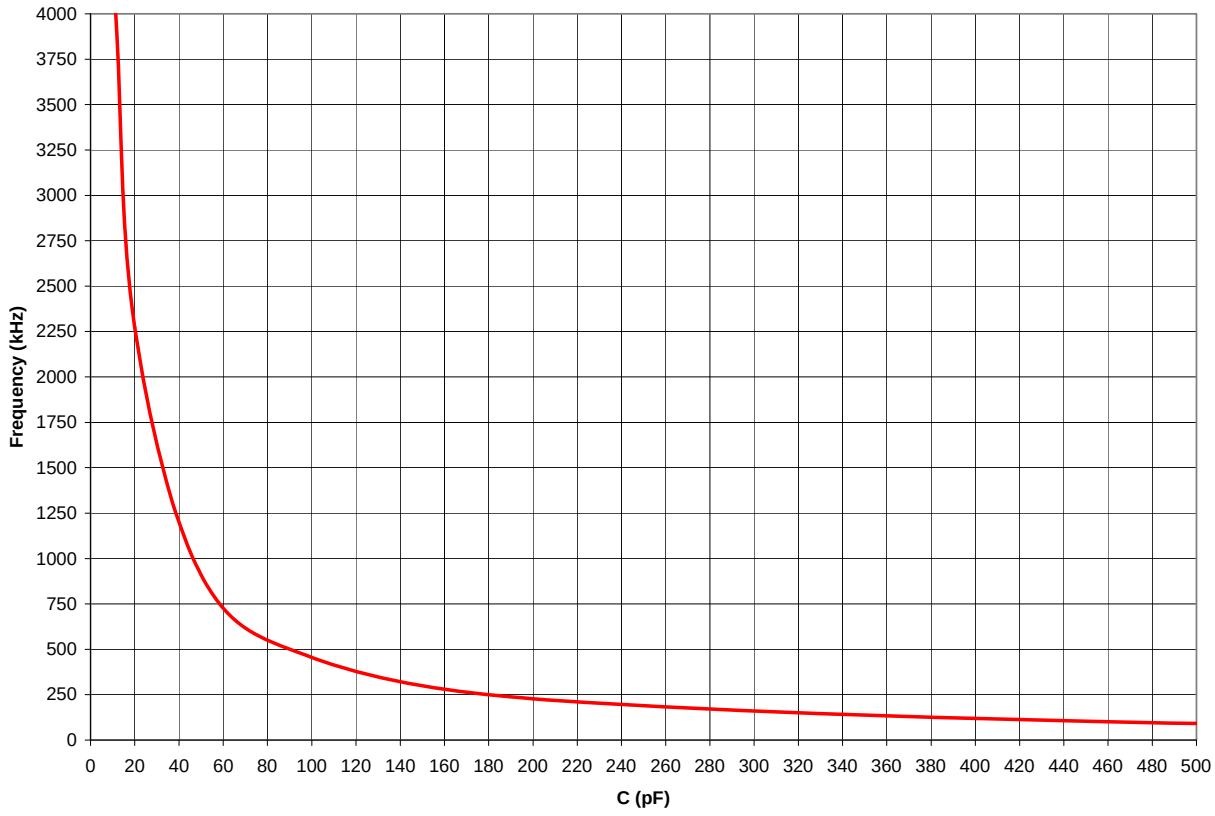



**Figure 64. Typical RC Oscillator Frequency as a Function of External Capacitance**

**Caution:** When using the external RC oscillator mode, the oscillator can stop oscillating if the power supply drops below 1.6V but remains above the Voltage Brown-Out threshold. The oscillator resumes oscillation when the supply voltage exceeds 1.6V.

## 25.4. Secondary Crystal Oscillator Operation

Figure 65 displays the recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 32kHz. The recommended 32kHz crystal specifications are provided in Table 173. Printed circuit board layout must add no more than 4pF of stray capacitance to either the $X_{IN}$ or $X_{OUT}$ pins. If oscillation does not occur, reduce the values of capacitors $C_1$ and $C_2$ to decrease loading.

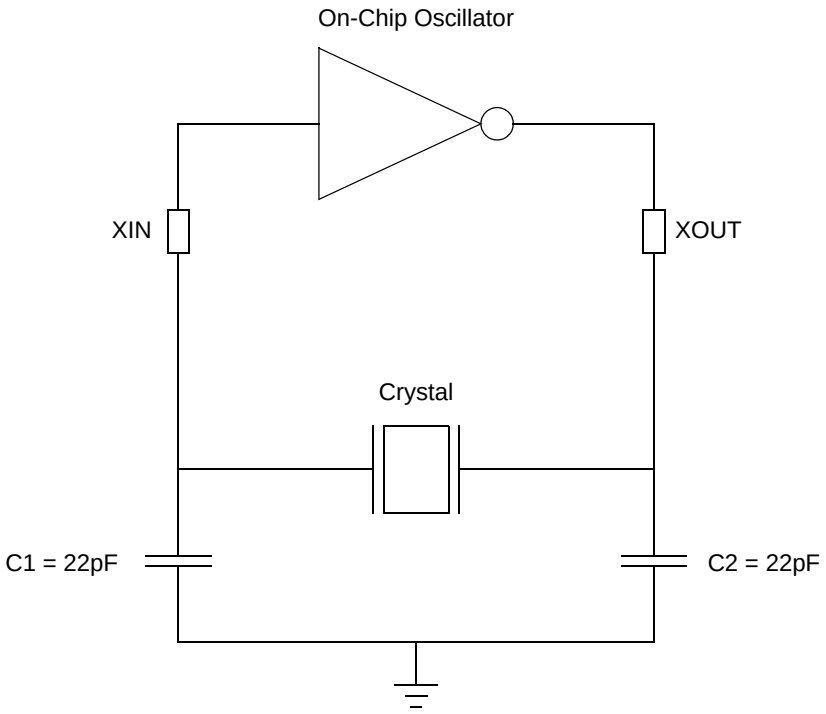



**Figure 65. Recommended 32kHz Crystal Oscillator Configuration**

**Table 173. Recommended Crystal Oscillator Specifications**

| Parameter | Value | Units | Comments |
|---|---|---|---|
| Frequency | 32 | kHz | |
| Resonance | Serial | | |
| Mode | Fundamental | | |
| Series Resistance ($R_S$) | 160K | W | Maximum |
| Load Capacitance ($C_L$) | 30 | pF | Maximum |
| Shunt Capacitance ($C_0$) | 5 | pF | Maximum |
| Drive Level | 0.1 | mW | Maximum |

# Chapter 26. Internal Precision Oscillator

The Internal Precision Oscillator (IPO) is designed for use without external components. You can either manually trim the oscillator for a nonstandard 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 frequencies: 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: ± 4% over temperature and voltage

## 26.1. 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. After it is 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 the Trim Bit Address Space section on page 282.

Select one of the frequencies for the oscillator using the INTSEL bits in the the Oscillator Control1 Register section on page 320.

# Chapter 27. eZ8 CPU Instruction Set

The eZ8 CPU instruction set is described in this chapter.

## 27.1. 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.

### 27.1.0.1. 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.
```

## 27.2. 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 174.

**Table 174. Assembly Language Syntax Example 1**

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

**Example 2.** In general, when an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or, using Escaped Mode Addressing, a Working Register R0–R15. If the contents of Register 43H and Working Register R8 are added and the result is stored in 43H, the assembly syntax and resulting object code is as listed in Table 175.

**Table 175. Assembly Language Syntax Example 2**

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

The register file size varies depending on the device type. See the device-specific product specification to determine the exact register file range available.

## 27.3. eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary section on page 336, the operands, condition codes, status flags and address modes are represented by the notational shorthand provided in Table 176.

**Table 176. Notational Shorthand**

| Notation | Description | Operand | Range |
|---|---|---|---|
| b | Bit | b | b represents a value from 0 to 7 (000B to 111B) |
| cc | Condition Code | — | See the Condition Codes overview in the eZ8 CPU Core User Manual (UM0128) |
| DA | Direct Address | Addrs | Addrs. represents a number in the range of 0000H to FFFFH |
| ER | Extended Addressing Register | Reg | Reg. represents a number in the range of 000H to FFFH |
| IM | Immediate Data | #Data | Data is a number between 00H to FFH |
| Ir | Indirect Working Register | @Rn | n = 0 –15 |
| IR | Indirect Register | @Reg | Reg. represents a number in the range of 00H to FFH |
| Irr | Indirect Working Register Pair | @RRp | p = 0, 2, 4, 6, 8, 10, 12 or 14 |
| IRR | Indirect Register Pair | @Reg | Reg. represents an even number in the range 00H to FEH |
| p | Polarity | p | Polarity is a single bit binary value of either 0B or 1B. |
| r | Working Register | Rn | n = 0–15 |
| R | Register | Reg | Reg. represents a number in the range of 00H to FFH |
| 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 177 contains additional symbols that are used throughout the eZ8 CPU Instruction Summary section on page 336.

**Table 177. 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, the statement:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location.

## 27.4. 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

Tables 178 through 185 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 178. 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 179. 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 180. 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 181. 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 |

**Table 181. CPU Control Instructions (Continued)**

| Mnemonic | Operands | Instruction |
|---|---|---|
| STOP | — | STOP Mode |
| WDT | — | Watchdog Timer Refresh |

**Table 182. Load Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| CLR | dst | Clear |
| LD | dst, src | Load |
| LDC | dst, src | Load Constant to/from Program Memory |
| LDCI | dst, src | Load Constant to/from Program Memory and Auto-Increment Addresses |
| LDE | dst, src | Load External Data to/from Data Memory |
| LDEI | dst, src | Load External Data to/from Data Memory and Auto-Increment Addresses |
| LDWX | dst, src | Load Word using Extended Addressing |
| LDX | dst, src | Load using Extended Addressing |
| LEA | dst, X(src) | Load Effective Address |
| POP | dst | Pop |
| POPX | dst | Pop using Extended Addressing |
| PUSH | src | Push |
| PUSHX | src | Push using Extended Addressing |

**Table 183. Logical Instructions**

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

**Table 184. 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 185. 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 |

## 27.5. eZ8 CPU Instruction Summary

Table 186 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 186. eZ8 CPU Instruction Summary**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC dst, src | dst ← dst + src + C | r | r | 12 | * | * | * | * | 0 | * | 2 | 3 |
| | | r | Ir | 13 | | | | | | | 2 | 4 |
| | | R | R | 14 | | | | | | | 3 | 3 |
| | | R | IR | 15 | | | | | | | 3 | 4 |
| | | R | IM | 16 | | | | | | | 3 | 3 |
| | | IR | IM | 17 | | | | | | | 3 | 4 |
| ADCX dst, src | dst ← dst + src + C | ER | ER | 18 | * | * | * | * | 0 | * | 4 | 3 |
| | | ER | IM | 19 | | | | | | | 4 | 3 |
| ADD dst, src | dst ← dst + src | r | r | 02 | * | * | * | * | 0 | * | 2 | 3 |
| | | r | Ir | 03 | | | | | | | 2 | 4 |
| | | R | R | 04 | | | | | | | 3 | 3 |
| | | R | IR | 05 | | | | | | | 3 | 4 |
| | | R | IM | 06 | | | | | | | 3 | 3 |
| | | IR | IM | 07 | | | | | | | 3 | 4 |
| ADDX dst, src | dst ← dst + src | ER | ER | 08 | * | * | * | * | 0 | * | 4 | 3 |
| | | ER | IM | 09 | | | | | | | 4 | 3 |

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

**Table 186. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AND dst, src | dst ← dst AND src | r | r | 52 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 53 | | | | | | | 2 | 4 |
| | | R | R | 54 | | | | | | | 3 | 3 |
| | | R | IR | 55 | | | | | | | 3 | 4 |
| | | R | IM | 56 | | | | | | | 3 | 3 |
| | | IR | IM | 57 | | | | | | | 3 | 4 |
| ANDX dst, src | dst ← dst AND src | ER | ER | 58 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 59 | | | | | | | 4 | 3 |
| 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 |
| | | | Ir | F7 | | | | | | | 3 | 4 |
| BTJNZ bit, src, dst | if src[bit] = 1 PC ← PC + X | | r | F6 | – | – | – | – | – | – | 3 | 3 |
| | | | Ir | F7 | | | | | | | 3 | 4 |
| BTJZ bit, src, dst | if src[bit] = 0 PC ← PC + X | | r | F6 | – | – | – | – | – | – | 3 | 3 |
| | | | Ir | F7 | | | | | | | 3 | 4 |
| CALL dst | SP ← SP -2 @SP ← PC PC ← dst | IRR | | D4 | – | – | – | – | – | – | 2 | 6 |
| | | DA | | D6 | | | | | | | 3 | 3 |
| CCF | C ← ~C | | | EF | * | – | – | – | – | –- | 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 186. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLR dst | dst ← 00H | R | | B0 | – | – | – | – | – | – | 2 | 2 |
| | | IR | | B1 | | | | | | | 2 | 3 |
| COM dst | dst ← ~dst | R | | 60 | – | * | * | 0 | – | – | 2 | 2 |
| | | IR | | 61 | | | | | | | 2 | 3 |
| CP dst, src | dst – src | r | r | A2 | * | * | * | * | – | – | 2 | 3 |
| | | r | Ir | A3 | | | | | | | 2 | 4 |
| | | R | R | A4 | | | | | | | 3 | 3 |
| | | R | IR | A5 | | | | | | | 3 | 4 |
| | | R | IM | A6 | | | | | | | 3 | 3 |
| | | IR | IM | A7 | | | | | | | 3 | 4 |
| CPC dst, src | dst – src – C | r | r | 1F A2 | * | * | * | * | – | – | 3 | 3 |
| | | r | Ir | 1F A3 | | | | | | | 3 | 4 |
| | | R | R | 1F A4 | | | | | | | 4 | 3 |
| | | R | IR | 1F A5 | | | | | | | 4 | 4 |
| | | R | IM | 1F A6 | | | | | | | 4 | 3 |
| | | IR | IM | 1F A7 | | | | | | | 4 | 4 |
| CPCX dst, src | dst – src – C | ER | ER | 1F A8 | * | * | * | * | – | – | 5 | 3 |
| | | ER | IM | 1F A9 | | | | | | | 5 | 3 |
| CPX dst, src | dst – src | ER | ER | A8 | * | * | * | * | – | – | 4 | 3 |
| | | ER | IM | A9 | | | | | | | 4 | 3 |
| DA dst | dst ← DA(dst) | R | | 40 | * | * | * | X | – | – | 2 | 2 |
| | | IR | | 41 | | | | | | | 2 | 3 |
| DEC dst | dst ← dst – 1 | R | | 30 | – | * | * | * | – | – | 2 | 2 |
| | | IR | | 31 | | | | | | | 2 | 3 |
| DECW dst | dst ← dst – 1 | RR | | 80 | – | * | * | * | – | – | 2 | 5 |
| | | IRR | | 81 | | | | | | | 2 | 6 |
| DI | IRQCTL[7] ← 0 | | | 8F | – | – | – | – | – | – | 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 186. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DJNZ dst, RA | dst ← dst – 1<br>if dst ≠ 0<br>PC ← PC + X | r | | 0A-FA | – | – | – | – | – | – | 2 | 3 |
| EI | IRQCTL[7] ← 1 | | | 9F | – | – | – | – | – | – | 1 | 2 |
| HALT | Halt Mode | | | 7F | – | – | – | – | – | – | 1 | 2 |
| INC dst | dst ← dst + 1 | R | | 20 | – | * | * | – | – | – | 2 | 2 |
| | | IR | | 21 | | | | | | | 2 | 3 |
| | | r | | 0E-FE | | | | | | | 1 | 2 |
| INCW dst | dst ← dst + 1 | RR | | A0 | – | * | * | * | – | – | 2 | 5 |
| | | IRR | | A1 | | | | | | | 2 | 6 |
| IRET | FLAGS ← @SP<br>SP ← SP + 1<br>PC ← @SP<br>SP ← SP + 2<br>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<br>PC ← dst | DA | | 0D-FD | – | – | – | – | – | – | 3 | 2 |
| JR dst | PC ← PC + X | DA | | 8B | – | – | – | – | – | – | 2 | 2 |
| JR cc, dst | if cc is true<br>PC ← PC + X | DA | | 0B-FB | – | – | – | – | – | – | 2 | 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 186. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LD dst, rc | dst ← src | r | IM | 0C-FC | – | – | – | – | – | – | 2 | 2 |
| | | r | X(r) | C7 | | | | | | | 3 | 3 |
| | | X(r) | r | D7 | | | | | | | 3 | 4 |
| | | r | Ir | E3 | | | | | | | 2 | 3 |
| | | R | R | E4 | | | | | | | 3 | 2 |
| | | R | IR | E5 | | | | | | | 3 | 4 |
| | | R | IM | E6 | | | | | | | 3 | 2 |
| | | IR | IM | E7 | | | | | | | 3 | 3 |
| | | Ir | r | F3 | | | | | | | 2 | 3 |
| | | IR | R | F5 | | | | | | | 3 | 3 |
| LDC dst, src | dst ← src | r | Irr | C2 | – | – | – | – | – | – | 2 | 5 |
| | | Ir | Irr | C5 | | | | | | | 2 | 9 |
| | | Irr | r | D2 | | | | | | | 2 | 5 |
| LDCI dst, src | dst ← src<br>r ← r + 1<br>rr ← rr + 1 | Ir | Irr | C3 | – | – | – | – | – | – | 2 | 9 |
| | | Irr | Ir | D3 | | | | | | | 2 | 9 |
| LDE dst, src | dst ← src | r | Irr | 82 | – | – | – | – | – | – | 2 | 5 |
| | | Irr | r | 92 | | | | | | | 2 | 5 |
| LDEI dst, src | dst ← src<br>r ← r + 1<br>rr ← rr + 1 | Ir | Irr | 83 | – | – | – | – | – | – | 2 | 9 |
| | | Irr | Ir | 93 | | | | | | | 2 | 9 |
| LDWX dst, src | dst ← src | ER | ER | 1FE8 | – | – | – | – | – | – | 5 | 4 |

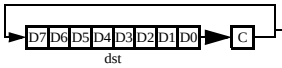
Flags notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 186. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LDX dst, src | dst ← src | r | ER | 84 | – | – | – | – | – | – | 3 | 2 |
| | | Ir | ER | 85 | | | | | | | 3 | 3 |
| | | R | IRR | 86 | | | | | | | 3 | 4 |
| | | IR | IRR | 87 | | | | | | | 3 | 5 |
| | | r | X(rr) | 88 | | | | | | | 3 | 4 |
| | | X(rr) | r | 89 | | | | | | | 3 | 4 |
| | | ER | r | 94 | | | | | | | 3 | 2 |
| | | ER | Ir | 95 | | | | | | | 3 | 3 |
| | | IRR | R | 96 | | | | | | | 3 | 4 |
| | | IRR | IR | 97 | | | | | | | 3 | 5 |
| | | ER | ER | E8 | | | | | | | 4 | 2 |
| | | ER | IM | E9 | | | | | | | 4 | 2 |
| LEA dst, X(src) | dst ← src + X | r | X(r) | 98 | – | – | – | – | – | – | 3 | 3 |
| | | rr | X(rr) | 99 | | | | | | | 3 | 5 |
| MULT dst | dst[15:0] ← dst[15:8] * dst[7:0] | RR | | F4 | – | – | – | – | – | – | 2 | 8 |
| NOP | No operation | | | 0F | – | – | – | – | – | – | 1 | 2 |
| OR dst, src | dst ← dst OR src | r | r | 42 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 43 | | | | | | | 2 | 4 |
| | | R | R | 44 | | | | | | | 3 | 3 |
| | | R | IR | 45 | | | | | | | 3 | 4 |
| | | R | IM | 46 | | | | | | | 3 | 3 |
| | | IR | IM | 47 | | | | | | | 3 | 4 |
| ORX dst, src | dst ← dst OR src | ER | ER | 48 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 49 | | | | | | | 4 | 3 |
| POP dst | dst ← @SP<br>SP ← SP + 1 | R | | 50 | – | – | – | – | – | – | 2 | 2 |
| | | IR | | 51 | | | | | | | 2 | 3 |

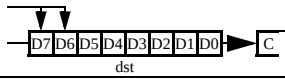
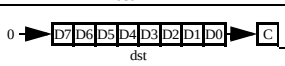
Flags notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 186. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POPX dst | dst ← @SP<br>SP ← SP + 1 | ER | | D8 | – | – | – | – | – | – | 3 | 2 |
| PUSH src | SP ← SP – 1<br>@SP ← src | R | | 70 | – | – | – | – | – | – | 2 | 2 |
| | | IR | | 71 | | | | | | | 2 | 3 |
| | | IM | | IF70 | | | | | | | 3 | 2 |
| | | | | | | | | | | | | |
| PUSHX src | SP ← SP – 1<br>@SP ← src | ER | | C8 | – | – | – | – | – | – | 3 | 2 |
| RCF | C ← 0 | | | CF | 0 | – | – | – | – | – | 1 | 2 |
| RET | PC ← @SP<br>SP ← SP + 2 | | | AF | – | – | – | – | – | – | 1 | 4 |
| RL dst | | R | | 90 | * | * | * | * | – | – | 2 | 2 |
| | | IR | | 91 | | | | | | | 2 | 3 |
| RLC dst | | R | | 10 | * | * | * | * | – | – | 2 | 2 |
| | | IR | | 11 | | | | | | | 2 | 3 |
| RR dst | | R | | E0 | * | * | * | * | – | – | 2 | 2 |
| | | IR | | E1 | | | | | | | 2 | 3 |
| RRC dst |  | R | | C0 | * | * | * | * | – | – | 2 | 2 |
| | | IR | | C1 | | | | | | | 2 | 3 |
| SBC dst, src | dst ← dst – src – C | r | r | 32 | * | * | * | * | 1 | * | 2 | 3 |
| | | r | Ir | 33 | | | | | | | 2 | 4 |
| | | R | R | 34 | | | | | | | 3 | 3 |
| | | R | IR | 35 | | | | | | | 3 | 4 |
| | | R | IM | 36 | | | | | | | 3 | 3 |
| | | IR | IM | 37 | | | | | | | 3 | 4 |
| SBCX dst, src | dst ← dst – src – C | ER | ER | 38 | * | * | * | * | 1 | * | 4 | 3 |
| | | ER | IM | 39 | | | | | | | 4 | 3 |

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

**Table 186. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCF | C ← 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 ← src | | IM | 01 | – | – | – | – | – | – | 2 | 2 |
| STOP | STOP Mode | | | 6F | – | – | – | – | – | – | 1 | 2 |
| SUB dst, src | dst ← dst – src | r | r | 22 | * | * | * | * | 1 | * | 2 | 3 |
| | | r | Ir | 23 | | | | | | | 2 | 4 |
| | | R | R | 24 | | | | | | | 3 | 3 |
| | | R | IR | 25 | | | | | | | 3 | 4 |
| | | R | IM | 26 | | | | | | | 3 | 3 |
| | | IR | IM | 27 | | | | | | | 3 | 4 |
| SUBX dst, src | dst ← dst – src | ER | ER | 28 | * | * | * | * | 1 | * | 4 | 3 |
| | | ER | IM | 29 | | | | | | | 4 | 3 |
| SWAP dst | dst[7:4] ↔ dst[3:0] | R | | F0 | X | * | * | X | – | – | 2 | 2 |
| | | IR | | F1 | | | | | | | 2 | 3 |
| TCM dst, src | (NOT dst) AND src | r | r | 62 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 63 | | | | | | | 2 | 4 |
| | | R | R | 64 | | | | | | | 3 | 3 |
| | | R | IR | 65 | | | | | | | 3 | 4 |
| | | R | IM | 66 | | | | | | | 3 | 3 |
| | | IR | IM | 67 | | | | | | | 3 | 4 |
| TCMX dst, src | (NOT dst) AND src | ER | ER | 68 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 69 | | | | | | | 4 | 3 |

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

**Table 186. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Op Code(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TM dst, src | dst AND src | r | r | 72 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 73 | | | | | | | 2 | 4 |
| | | R | R | 74 | | | | | | | 3 | 3 |
| | | R | IR | 75 | | | | | | | 3 | 4 |
| | | R | IM | 76 | | | | | | | 3 | 3 |
| | | IR | IM | 77 | | | | | | | 3 | 4 |
| TMX dst, src | dst AND src | ER | ER | 78 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 79 | | | | | | | 4 | 3 |
| TRAP Vector | SP ← SP – 2<br>@SP ← PC<br>SP ← SP – 1<br>@SP ← FLAGS<br>PC ← @Vector | | Vector | F2 | – | – | – | – | – | – | 2 | 6 |
| WDT | | | | 5F | – | – | – | – | – | – | 1 | 2 |
| XOR dst, src | dst ← dst XOR src | r | r | B2 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | B3 | | | | | | | 2 | 4 |
| | | R | R | B4 | | | | | | | 3 | 3 |
| | | R | IR | B5 | | | | | | | 3 | 4 |
| | | R | IM | B6 | | | | | | | 3 | 3 |
| | | IR | IM | B7 | | | | | | | 3 | 4 |
| XORX dst, src | dst ← dst XOR src | ER | ER | B8 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | B9 | | | | | | | 4 | 3 |

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

# Chapter 28. Op Code Maps

Figure 66 displays a description of the opcode map data and the abbreviations. Table 187 lists Op Code Map abbreviations.

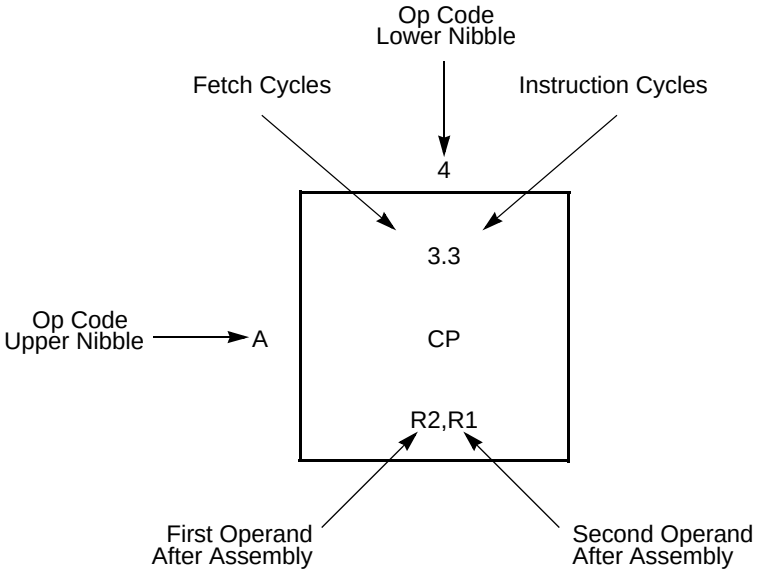



**Figure 66. Op Code Map Cell Description**

**Table 187. Op Code Map Abbreviations**

| Abbreviation | Description | Abbreviation | Description |
|---|---|---|---|
| b | Bit position | IRR | Indirect Register Pair |
| cc | Condition code | p | Polarity (0 or 1) |
| X | 8-bit signed index or displacement | r | 4-bit Working Register |
| DA | Destination address | R | 8-bit register |
| ER | Extended Addressing register | r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1 | Destination address |
| IM | Immediate data value | r2, R2, Ir2, Irr2, IR2, rr2, RR2, IRR2, ER2 | Source address |
| Ir | Indirect Working Register | RA | Relative |
| IR | Indirect register | rr | Working Register Pair |
| Irr | Indirect Working Register Pair | RR | Register Pair |

Figures 67 and 68 provide information about each of the eZ8 CPUinstructions.

| Upper Nibble (Hex) \ Lower Nibble (Hex) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.1 **BRK** | 2.2 **SRP** IM | 2.3 **ADD** r1,r2 | 2.4 **ADD** r1,Ir2 | 3.3 **ADD** R2,R1 | 3.4 **ADD** IR2,R1 | 3.3 **ADD** R1,IM | 3.4 **ADD** IR1,IM | 4.3 **ADDX** ER2,ER1 | 4.3 **ADDX** IM,ER1 | 2.3 **DJNZ** r1,X | 2.2 **JR** cc,X | 2.2 **LD** r1,IM | 3.2 **JP** cc,DA | 1.2 **INC** r1 | 1.2 **NOP** |
| 1 | 2.2 **RLC** R1 | 2.3 **RLC** IR1 | 2.3 **ADC** r1,r2 | 2.4 **ADC** r1,Ir2 | 3.3 **ADC** R2,R1 | 3.4 **ADC** IR2,R1 | 3.3 **ADC** R1,IM | 3.4 **ADC** IR1,IM | 4.3 **ADCX** ER2,ER1 | 4.3 **ADCX** IM,ER1 | | | | | | See 2nd Op Code Map |
| 2 | 2.2 **INC** R1 | 2.3 **INC** IR1 | 2.3 **SUB** r1,r2 | 2.4 **SUB** r1,Ir2 | 3.3 **SUB** R2,R1 | 3.4 **SUB** IR2,R1 | 3.3 **SUB** R1,IM | 3.4 **SUB** IR1,IM | 4.3 **SUBX** ER2,ER1 | 4.3 **SUBX** IM,ER1 | | | | | | 1, 2 ATM |
| 3 | 2.2 **DEC** R1 | 2.3 **DEC** IR1 | 2.3 **SBC** r1,r2 | 2.4 **SBC** r1,Ir2 | 3.3 **SBC** R2,R1 | 3.4 **SBC** IR2,R1 | 3.3 **SBC** R1,IM | 3.4 **SBC** IR1,IM | 4.3 **SBCX** ER2,ER1 | 4.3 **SBCX** IM,ER1 | | | | | | |
| 4 | 2.2 **DA** R1 | 2.3 **DA** IR1 | 2.3 **OR** r1,r2 | 2.4 **OR** r1,Ir2 | 3.3 **OR** R2,R1 | 3.4 **OR** IR2,R1 | 3.3 **OR** R1,IM | 3.4 **OR** IR1,IM | 4.3 **ORX** ER2,ER1 | 4.3 **ORX** IM,ER1 | | | | | | |
| 5 | 2.2 **POP** R1 | 2.3 **POP** IR1 | 2.3 **AND** r1,r2 | 2.4 **AND** r1,Ir2 | 3.3 **AND** R2,R1 | 3.4 **AND** IR2,R1 | 3.3 **AND** R1,IM | 3.4 **AND** IR1,IM | 4.3 **ANDX** ER2,ER1 | 4.3 **ANDX** IM,ER1 | | | | | | 1.2 **WDT** |
| 6 | 2.2 **COM** R1 | 2.3 **COM** IR1 | 2.3 **TCM** r1,r2 | 2.4 **TCM** r1,Ir2 | 3.3 **TCM** R2,R1 | 3.4 **TCM** IR2,R1 | 3.3 **TCM** R1,IM | 3.4 **TCM** IR1,IM | 4.3 **TCMX** ER2,ER1 | 4.3 **TCMX** IM,ER1 | | | | | | 1.2 **STOP** |
| 7 | 2.2 **PUSH** R2 | 2.3 **PUSH** IR2 | 2.3 **TM** r1,r2 | 2.4 **TM** r1,Ir2 | 3.3 **TM** R2,R1 | 3.4 **TM** IR2,R1 | 3.3 **TM** R1,IM | 3.4 **TM** IR1,IM | 4.3 **TMX** ER2,ER1 | 4.3 **TMX** IM,ER1 | | | | | | 1.2 **HALT** |
| 8 | 2.5 **DECW** RR1 | 2.6 **DECW** IRR1 | 2.5 **LDE** r1,Irr2 | 2.9 **LDEI** Ir1,Irr2 | 3.2 **LDX** r1,ER2 | 3.3 **LDX** Ir1,ER2 | 3.4 **LDX** IRR2,R1 | 3.5 **LDX** IRR2,IR1 | 3.4 **LDX** r1,rr2,X | 3.4 **LDX** rr1,r2,X | | | | | | 1.2 **DI** |
| 9 | 2.2 **RL** R1 | 2.3 **RL** IR1 | 2.5 **LDE** r2,Irr1 | 2.9 **LDEI** Ir2,Irr1 | 3.2 **LDX** r2,ER1 | 3.3 **LDX** Ir2,ER1 | 3.4 **LDX** R2,IRR1 | 3.5 **LDX** IR2,IRR1 | 3.3 **LEA** r1,r2,X | 3.5 **LEA** rr1,rr2,X | | | | | | 1.2 **EI** |
| A | 2.5 **INCW** RR1 | 2.6 **INCW** IRR1 | 2.3 **CP** r1,r2 | 2.4 **CP** r1,Ir2 | 3.3 **CP** R2,R1 | 3.4 **CP** IR2,R1 | 3.3 **CP** R1,IM | 3.4 **CP** IR1,IM | 4.3 **CPX** ER2,ER1 | 4.3 **CPX** IM,ER1 | | | | | | 1.4 **RET** |
| B | 2.2 **CLR** R1 | 2.3 **CLR** IR1 | 2.3 **XOR** r1,r2 | 2.4 **XOR** r1,Ir2 | 3.3 **XOR** R2,R1 | 3.4 **XOR** IR2,R1 | 3.3 **XOR** R1,IM | 3.4 **XOR** IR1,IM | 4.3 **XORX** ER2,ER1 | 4.3 **XORX** IM,ER1 | | | | | | 1.5 **IRET** |
| C | 2.2 **RRC** R1 | 2.3 **RRC** IR1 | 2.5 **LDC** r1,Irr2 | 2.9 **LDCI** Ir1,Irr2 | 2.3 **JP** IRR1 | 2.9 **LDC** Ir1,Irr2 | | 3.4 **LD** r1,r2,X | 3.2 **PUSHX** ER2 | | | | | | | 1.2 **RCF** |
| D | 2.2 **SRA** R1 | 2.3 **SRA** IR1 | 2.5 **LDC** r2,Irr1 | 2.9 **LDCI** Ir2,Irr1 | 2.6 **CALL** IRR1 | 2.2 **BSWAP** R1 | 3.3 **CALL** DA | 3.4 **LD** r2,r1,X | 3.2 **POPX** ER1 | | | | | | | 1.2 **SCF** |
| E | 2.2 **RR** R1 | 2.3 **RR** IR1 | 2.2 **BIT** p,b,r1 | 2.3 **LD** r1,Ir2 | 3.2 **LD** R2,R1 | 3.3 **LD** IR2,R1 | 3.2 **LD** R1,IM | 3.3 **LD** IR1,IM | 4.2 **LDX** ER2,ER1 | 4.2 **LDX** IM,ER1 | | | | | | 1.2 **CCF** |
| F | 2.2 **SWAP** R1 | 2.3 **SWAP** IR1 | 2.6 **TRAP** Vector | 2.3 **LD** Ir1,r2 | 2.8 **MULT** RR1 | 3.3 **LD** R2,IR1 | 3.3 **BTJ** p,b,r1,X | 3.4 **BTJ** p,b,Ir1,X | | | | | | | | |

**Figure 67. First Op Code Map**

| Upper Nibble (Hex) \ Lower Nibble (Hex) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | |
| 7 | 3, 2 **PUSH** IM | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | |
| A | | | 3.3 **CPC** r1,r2 | 3.4 **CPC** r1,Ir2 | 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 68. Second Op Code Map after 1FH**

# Chapter 29. Electrical Characteristics

The data in this chapter is prequalification and precharacterization and is subject to change. Additional electrical characteristics can be found in the individual chapters.

## 29.1. Absolute Maximum Ratings

Stresses greater than those listed in Table 188 can cause permanent damage to the device. These ratings are stress ratings only. Operation of the device at any condition outside those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods can affect device reliability. For improved reliability, tie unused inputs to one of the supply voltages ($V_{DD}$ or $V_{SS}$).

**Table 188. 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 | |
| **20-pin Packages Maximum Ratings at 0°C to 70°C** | | | | |
| Total power dissipation | | 430 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 120 | mA | |
| **28-pin Packages Maximum Ratings at 0°C to 70°C** | | | | |
| Total power dissipation | | 450 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 125 | mA | |
| **40-pin PDIP Maximum Ratings at –40°C to 70°C** | | | | |
| Total power dissipation | | 1000 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 275 | mA | |
| **40-pin PDIP Maximum Ratings at 70°C to 105°C** | | | | |
| Total power dissipation | | 540 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 150 | mA | |

Notes:
*Operating temperature is specified in DC Characteristics.
1. This voltage applies to all pins except the following: $V_{DD}$, $AV_{DD}$.

**Table 188. Absolute Maximum Ratings* (Continued)**

| Parameter | Min | Max | Units | Notes |
|---|---|---|---|---|
| **44-Pin QFN Maximum Ratings at –40°C to 70°C** | | | | |
| Total power dissipation | | 750 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 200 | mA | |
| **44-Pin QFN Maximum Ratings at 70°C to 105°C** | | | | |
| Total power dissipation | | 295 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 83 | mA | |
| **44-pin LQFP Maximum Ratings at –40°C to 70°C** | | | | |
| Total power dissipation | | 750 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 200 | mA | |
| **44-pin LQFP Maximum Ratings at 70°C to 105°C** | | | | |
| Total power dissipation | | 410 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 114 | mA | |

Notes:
*Operating temperature is specified in DC Characteristics.
1. This voltage applies to all pins except the following: $V_{DD}$, $AV_{DD}$.

## 29.2. DC Characteristics

Table 189 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 189. DC Characteristics**

| | | $T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $V_{DD}$ | Supply Voltage | 1.8 | – | 3.6 | V | |
| $V_{IL1}$ | Low Level Input Voltage | –0.3 | – | 0.3*$V_{DD}$ | V | For all input pins except $\overline{\text{RESET}}$, DBG, XIN |
| $V_{IL2}$ | Low Level Input Voltage | –0.3 | – | 0.2*$V_{DD}$ | V | For $\overline{\text{RESET}}$, DBG, XIN |
| $V_{IH1}$ | High Level Input Voltage | 0.7*$V_{DD}$ | – | 5.5 | V | Port A, B, C, D and E pins (Digital inputs) |

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

**Table 189. DC Characteristics (Continued)**

| | | $T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $V_{IH2}$ | High Level Input Voltage | 0.7*$V_{DD}$ | – | $V_{DD}$+0.3 | V | Ports B and C (Analog) |
| $V_{OL1}$ | Low Level Output Voltage | – | – | 0.4 | V | $I_{OL}$ = 2 mA; $V_{DD}$ = 3.0V<br>High Output Drive disabled. |
| $V_{OH1}$ | High Level Output Voltage | $V_{DD}$–0.5 | – | – | V | $I_{OH}$ = –2 mA; $V_{DD}$ = 3.0V<br>High Output Drive disabled. |
| $V_{OL2}$ | Low Level Output Voltage | – | – | 0.6 | V | $I_{OL}$ = 20 mA; $V_{DD}$ = 3.3V<br>High Output Drive enabled. |
| $V_{OH2}$ | High Level Output Voltage | $V_{DD}$–0.5 | – | – | V | $I_{OH}$ = –20 mA; $V_{DD}$ = 3.3V<br>High Output Drive enabled. |
| $I_{IL}$ | Input Leakage Current | –5 | – | +5 | µA | $V_{DD}$ = 3.6V;<br>$V_{IN}$ = $V_{DD}$ or $V_{SS}$[1] |
| $I_{TL}$ | Tristate Leakage Current | –5 | – | +5 | µA | $V_{DD}$ = 3.6V |
| $I_{LED}$ | Controlled LED Current Drive | 1.5 | 3 | 4.5 | mA | {AFS2,AFS1} = {0,0}, $V_{DD}$ = 3.3V |
| | | 2.8 | 7 | 10.5 | mA | {AFS2,AFS1} = {0,1}, $V_{DD}$ = 3.3V |
| | | 7.8 | 13 | 19.5 | mA | {AFS2,AFS1} = {1,0}, $V_{DD}$ = 3.3V |
| | | 12 | 20 | 30 | mA | {AFS2,AFS1} = {1,1}, $V_{DD}$ = 3.3V |
| $C_{PAD}$ | 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 | µA | $V_{DD}$ = 3.0V–3.6V |

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

The currents in Table 190 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 190. Supply Current Characteristics**

| | | $T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typical[1]** | **Max** | **Units** | **Conditions[2]** |
| $I_{DDA1}$ | Active Mode Device Current Executing from Flash | | 8.5 | | mA | Typical: 20MHz[3, 4, 5, 6], $V_{DD}$ = 3V, Flash, 25°C |
| $I_{DDA2}$ | Active Mode Device Current Executing from PRAM | | 6 | | mA | Typical: 20MHz[3, 4, 5, 6], $V_{DD}$ = 3V, PRAM, 25°C |
| $I_{DDH}$ | Halt Mode Device Current | | TBD | | mA | Typical: 20MHz[3, 4, 5], $V_{DD}$ typical, 25°C |
| $I_{DDS1}$ | Stop Mode Device Current | | 2.5 | | µA | Typical: WDT, $V_{DD}$ typical, 25 °C, all peripherals including VBO disabled[3, 4, 6] |
| $I_{DDS2}$ | Stop Mode Device Current | | <1 | | µA | Typical: $V_{DD}$ typical, 25°C, all peripherals disabled including VBO and WDT[3, 4, 6] |

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 $V_{DD}/AV_{DD}$ 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 $V_{DD}/AV_{DD}$ or $V_{SS}/AV_{SS}$ as appropriate.

Figures 69 through 72 display the typical current consumption at voltages of 1.8V, 2.0V, 2.7V, 3.0V, 3.3V and 3.6V, respectively, versus different system clock frequencies while operating at a temperature of 25ºC.

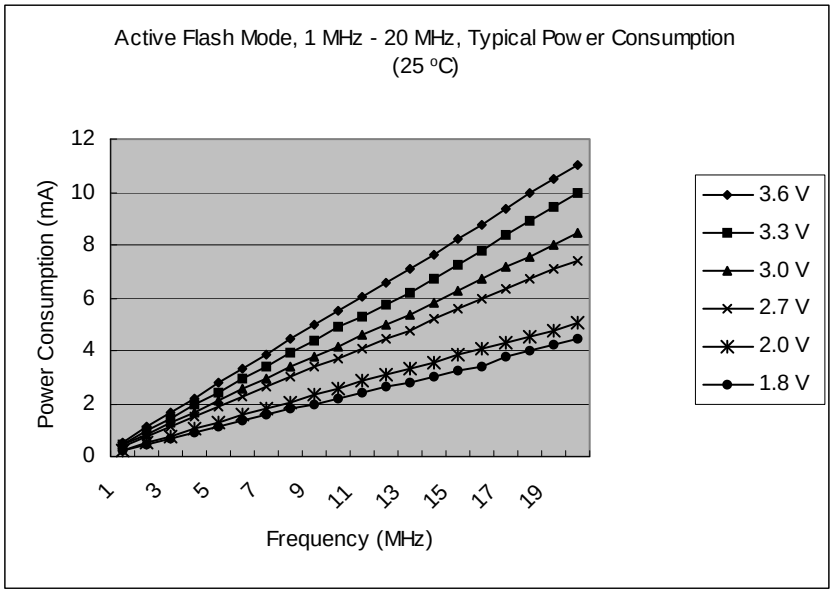



**Figure 69. Typical Active Flash Mode Supply Current (1–20MHz)**

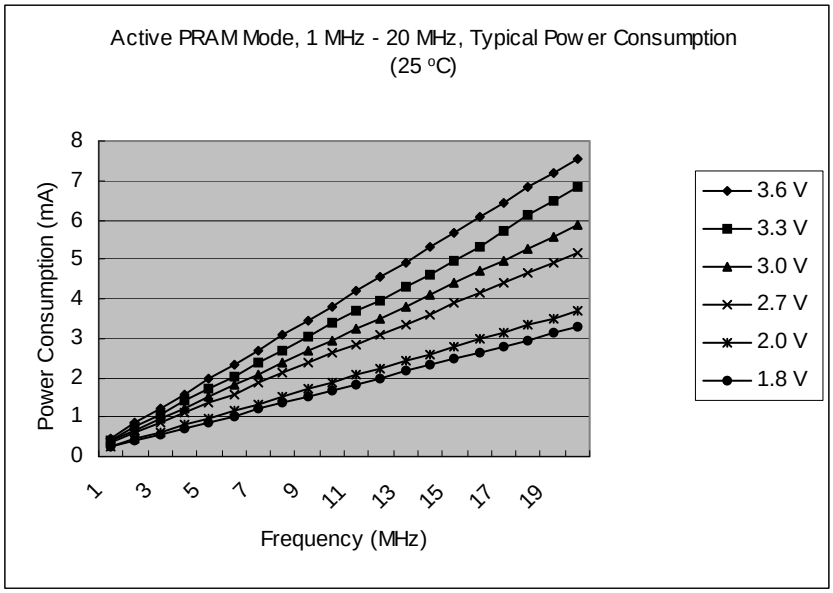



**Figure 70. Typical Active PRAM Mode Supply Current (1–20 MHz)**

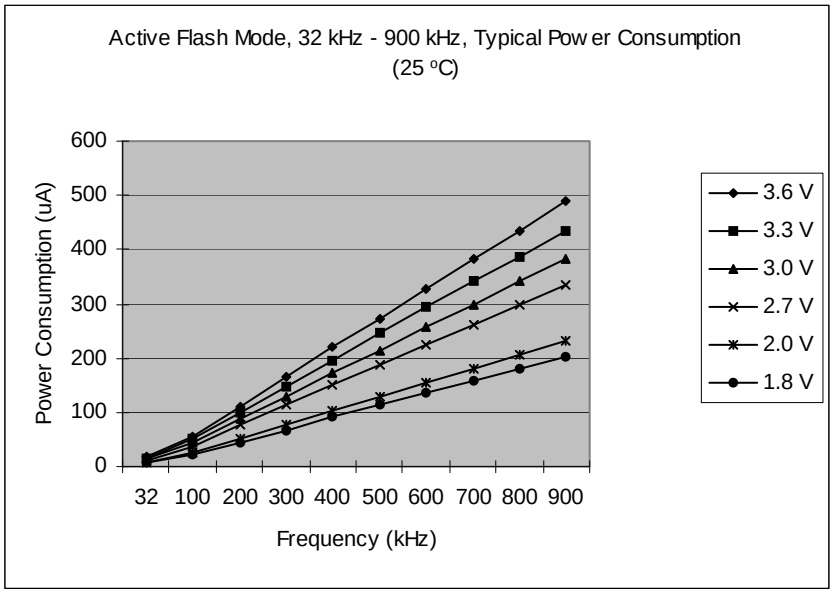



**Figure 71. Typical Active Flash Mode Supply Current (32–900 kHz)**

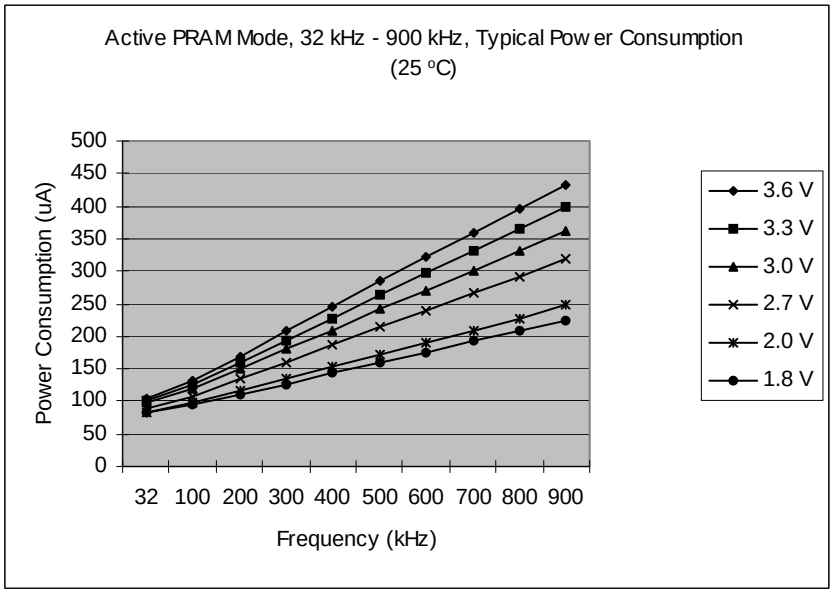



**Figure 72. Typical Active PRAM Mode Supply Current (32–900 kHz)**

Figure 73 displays the STOP Mode supply current versus ambient temperature and $V_{DD}$ level with all peripherals disabled.

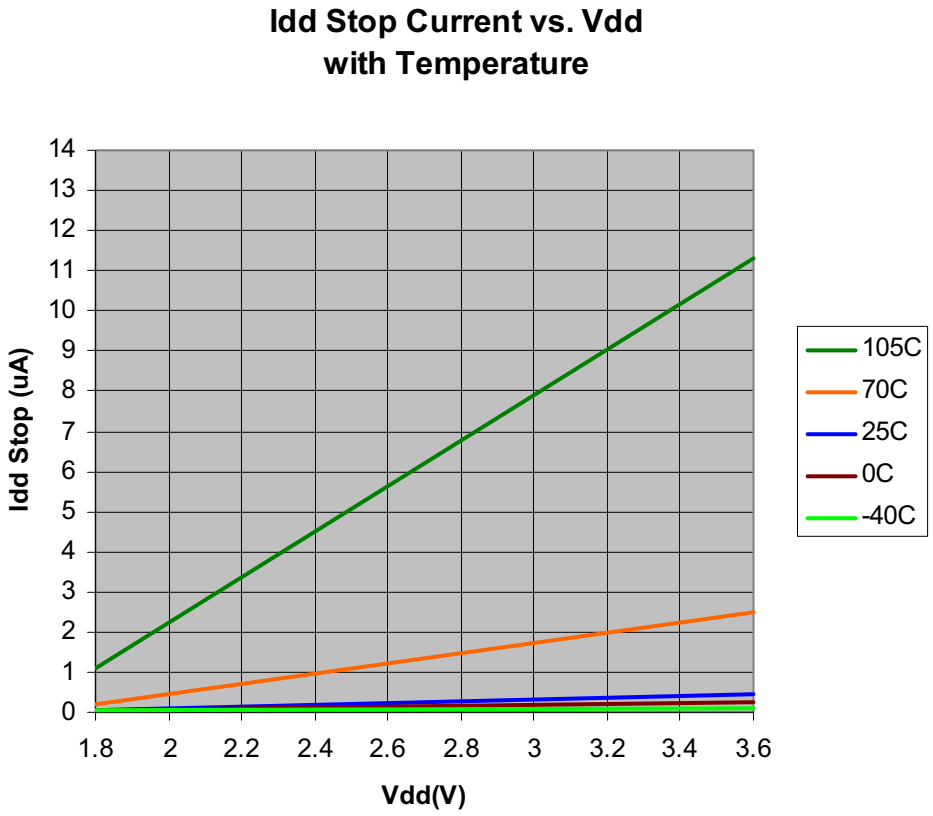



**Figure 73. STOP Mode Current Consumption as a Function of $V_{DD}$ with Temperature as a Parameter; all Peripherals Disabled**

## 29.3. AC Characteristics

Table 191 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 191. AC Characteristics**

| | | $V_{DD}$ = 1.8 to 3.6 V<br>$T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $F_{SYSCLK}$ | System Clock Frequency | – | | 20.0 | MHz | Read-only from Flash memory. See Figure 74. |
| | | 0.032768 | | 20.0 | MHz | Program or erasure of Flash memory |
| $T_{XIN}$ | System Clock Period | 50 | | – | ns | $T_{CLK} = 1/F_{sysclk}$ |
| $T_{XINH}$ | System Clock High Time | 20 | | 30 | ns | $T_{CLK}$ = 50 ns |
| $T_{XINL}$ | System Clock Low Time | 20 | | 30 | ns | $T_{CLK}$ = 50 ns |
| $T_{XINR}$ | System Clock Rise Time | – | | 3 | ns | $T_{CLK}$ = 50 ns |
| $T_{XINF}$ | System Clock Fall Time | – | | 3 | ns | $T_{CLK}$ = 50 ns |

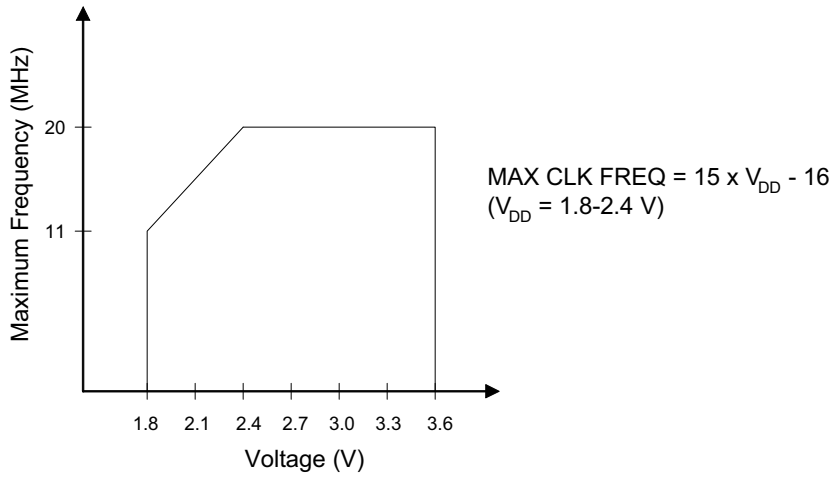



**Figure 74. $V_{DD}$ Versus Maximum System Clock Frequency**

## 29.4. On-Chip Peripheral AC and DC Electrical Characteristics

Tables 192 through 203 provide AC and DC electrical characteristics data for the on-chip peripherals.

**Table 192. Power-On Reset and Voltage Brown-Out Electrical Characteristics and Timing**

| | | $T_A$ = 0°C to +70°C | | | $T_A$ = –40°C to +105°C | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Min** | **Typ[1]** | **Max** | **Units** | **Conditions** |
| $V_{POR}$ | Power-On Reset Voltage Threshold | | | | 1.36 | 1.6 | 1.84 | V | $V_{DD} = V_{POR}$ |
| $V_{TH}$ | POR Start Voltage | | | | – | 0.6 | – | V | |
| $I_{DD}POR$ | POR Active Current | | | | – | – | 3 | µA | |
| $I_{DDQ}POR$ | POR Quiescent Current | | | | – | 5 | – | nA | |
| $T_{ANA}$ | Power-On Reset Analog Delay | | | | – | 50 | – | µs | $V_{DD} > V_{POR}$ |
| $T_{SMR}$ | Stop Mode Recovery Delay | | | | – | – | 6 | µs | |
| $V_{VBO}$ | Voltage Brown-Out Reset Voltage Threshold | | | | 1.6 | 1.8 | 2.0 | V | $V_{DD} = V_{VBO}$ |
| $V_{HYS}$ | Hysteresis of $V_{VBO}$ | | | | – | 80 | – | mV | |
| $T_{VBO}$ | Voltage Brown-Out Pulse Rejection Period | | | | – | 10 | – | µs | |
| $I_{DD}VBO$ | VBO Active Current | | | | – | – | 5 | µA | |
| $I_{DDQ}VBO$ | VBO Quiescent Current | | | | – | 5 | – | nA | |
| $T_{RAMP}$ | $V_{DD}$ rampup time | | | | 0.10 | – | 100 | ms | |

[1]Data in the typical column is from characterization at 3.3 V and 0°C. These values are provided for design guidance only and are not tested in production.

**Table 193. Flash Memory Electrical Characteristics and Timing**

| | VDD = 2.7V to 3.6V<br>TA = 0°C to +70°C<br>TA = –40°C to +105°C | | | | |
|---|---|---|---|---|---|
| **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| Flash Byte Read Time | 100 | – | – | ns | $V_{DD}$ = 1.8 V to 3.6V |
| Flash Byte Program Time | 20 | – | 40 | µs | |
| 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°C |
| Endurance | 5,000 | – | – | cycles | Program/erase cycles |

**Table 194. Watchdog Timer Electrical Characteristics and Timing**

| | | VDD = 1.8V to 3.6V<br>TA = 0°C to +70°C<br>TA = –40°C to +105°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Unit** | **Conditions** |
| $T_{STARTUP}$ | | – | – | 10 | ms | After pd disable only |
| $I_{DD}$WDT | WDT Active Current | – | – | 5 | µA | |
| $I_{DDQ}$WDT | WDT Quiescent Current | – | 5 | – | nA | |
| $F_{WDT}$ | WDT Oscillator Frequency | 2.5 | 5 | 20 | kHz | |

**Table 195. Non-Volatile Data Storage**

| | VDD = 2.7V to 3.6V<br>TA = 0°C to +70°C<br>TA = –40°C to +105°C | | | | |
|---|---|---|---|---|---|
| **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| NVDS Byte Read Time | 34 | – | 519 | µs | With system clock at 20MHz |
| NVDS Byte Program Time | 0.171 | – | 39.7 | ms | With system clock at 20MHz |
| Data Retention | 20 | – | – | years | 25°C |
| Endurance | 50,000 | – | – | cycles | Cumulative write cycles for entire memory |

**Table 196. Analog-to-Digital Converter Electrical Characteristics and Timing**

| | | $T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| 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 |
| $I_{DD}$ADC | ADC Active Current | – | – | 2.5 | mA | |
| $I_{DDQ}$ADC | ADC Quiescent Current | – | 5 | – | nA | |
| $V_{INT_REF}$ | Internal Reference Voltage | – | 1.6 | – | V | REFEN=1, INTREF_SEL=0. See Table 101 on page 189. |
| | | – | AVDD | – | V | REFEN=1, INTREF_SEL=1. See Table 101 on page 189. |
| $V_{EXT_REF}$ | External Reference Voltage | 1.6 | – | 90% $AV_{DD}$ | V | REFEN=0. See Table 101 on page 189. |
| $V_{INANA}$ | Analog Input Range | 0 | – | 1.6 | V | Internal reference = 1.6V |
| | | 0 | – | 90% $AV_{DD}$ | V | External reference or use AVDD as internal reference |
| $C_{IN}$ | Analog Input Load | – | – | 5 | pF | |
| $T_S$ | Sample Time | 1.8 | – | – | µs | |
| $T_H$ | Hold Time | 0.5 | – | – | µs | |
| $T_{CONV}$ | Conversion Time | – | 13 | – | clock cycles | |
| $GBW_{IN}$ | Input Bandwidth | – | 200 | – | kHz | |
| $T_{WAKE}$ | Wake-up Time | – | – | 10 | µs | External reference |
| | | – | – | 10 | µs | Internal reference |
| $f_{ADC_CLK}$ | Maximum Frequency of adc_clk | – | – | 5 | MHz | $V_{DD}$ = 2.7V to 3.6V |
| | | – | – | 2.5 | MHz | $V_{DD}$ = 1.8V to 2.7V |

**Table 197. Comparator Electrical Characteristics**

| | | TA = 0°C to +70°C<br>TA = –40°C to +105°C<br>$V_{DD}$ = 1.8V to 3.6V | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $V_{OS}$ | Input DC Offset | – | 5 | – | mV | |
| $V_{CREF_P}$ | Programmable Internal Reference Voltage Range | 0 | – | 1.8 | V | |
| $V_{CREF_D}$ | Default Internal Reference Voltage | 0.90 | 1.0 | 1.10 | V | |
| $I_{DD}$CMP | Comparator Active Current | – | – | 400 | µA | |
| $I_{DDQ}$CMP | Comparator Quiescent Current | – | 5 | – | nA | |
| $V_{HYS}$ | Input Hysteresis | – | 8 | – | mV | |
| $T_{PROP}$ | Propagation Delay | – | 100 | – | ns | |

**Table 198. Temperature Sensor Electrical Characteristics**

| | | TA = 0°C to +70°C<br>TA = –40°C to +105°C | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ = 2.7 to 3.6V | | | $V_{DD}$ = 1.8 to 2.7V | | | | |
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $T_{AERR}$ | 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) |
| $I_{DD}$TEMP | Temperature Sensor Active Current | – | – | 100 | – | – | 100 | µA | |
| $I_{DDQ}$TEMP | Temperature Sensor Quiescent Current | – | 5 | – | – | 5 | – | nA | |
| $T_{WAKE}$ | Time for Wake up | – | 80 | 100 | – | 80 | 100 | µs | |

**Table 199. Low Power Operational Amplifier Characteristics**

| | | TA = 0°C to +70°C<br>TA = –40°C to +105°C | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ = 2.7 to 3.6V | | | $V_{DD}$ = 1.8 to 2.7V | | | | |
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| AV | DC Gain | – | 80 | – | – | 60 | – | dB | |
| PM | Phase Margin | – | 53 | – | – | 45 | – | deg | 13 pF loading |
| GBW | Gain Bandwidth Product | – | 0.3 | – | – | 0.3 | – | MHz | |
| $V_{OS}$ | Input Offset Voltage | –4 | – | 4 | –4 | – | 4 | mV | |
| $V_{OSTA}$ | Input Offset Temperature Drift | – | 1 | 10 | – | 1 | 10 | µV/°C | |
| $I_{outTA}$ | Output Current (Drive ability of LPO) | 50 | – | – | 40 | – | – | µA | |
| $I_{DD}$LPO | LPO Active Current | – | 10 | – | – | 10 | – | µA | |
| $I_{DDQ}$LPO | LPO Quiescent Current | – | 5 | – | – | 5 | – | nA | |
| $V_{COM}$ | Maximum Common Input Voltage | – | – | 1.4 | – | – | 0.7 | V | |

**Table 200. IPO Electrical Characteristics**

| | | $V_{DD}$ = 1.8 to 3.6V<br>$T_A$ = –40°C to +105°C | | | $V_{DD}$ = 2.7 to 3.6V<br>$T_A$ = 0°C to +70°C | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $T_{SETUP}$ | Setup Time for Output Frequency | | | 15 | | | 15 | µs | |
| $I_{DD}$IPO | IPO Active Supply Current | | 500 | | | 500 | | µA | |
| $I_{DDQ}$IPO | IPO Quiescent Current | | 5 | | | 5 | | nA | |

**Table 200. IPO Electrical Characteristics (Continued)**

| | | $V_{DD}$ = 1.8 to 3.6V<br>$T_A$ = –40°C to +105°C | | | $V_{DD}$ = 2.7 to 3.6V<br>$T_A$ = 0°C to +70°C | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $F_{IPO}$ | 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;<br>±4% 1.8 to 2.7 V, 0–70°C<br>±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 201. Low Voltage Detect Electrical Characteristics**

| | | $T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C<br>$V_{DD}$ = 1.8 to 3.6 V | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $I_{DD}LVD$ | LVD Active Current | – | – | 50 | µA | |
| $I_{DDQ}LVD$ | LVD Quiescent Current | – | 5 | – | nA | |
| $V_{TH}$ | Detected Source Voltage | $V_{TP}$ – 10% | $V_{TP}$[1] | $V_{TP}$ + 10% | V | |

**Table 201. Low Voltage Detect Electrical Characteristics (Continued)**

| | | $T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C<br>$V_{DD}$ = 1.8 to 3.6 V | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $V_{TH_PRO}$ | Detected Source Voltage for Flash Protection | 2.4 | 2.5 | 2.6 | V | |
| $T_{DELAY}$ | Delay from source voltage falling lower than $V_{TP}$ to $I_{VD_OUT}$ output logic High | 50 | 1000 | – | ns | |

Note: [1] $V_{TP}$ is a user-set threshold voltage to be detected.

**Table 202. Crystal Oscillator Characteristics**

| | | $T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ = 2.7 to 3.6V | | | $V_{DD}$ = 1.8 to 2.7V | | | | |
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $I_{DD}XTAL$ | Crystal Oscillator Active Supply Current | – | – | 500 | – | – | 300 | µA | |
| $I_{DDQ}XTAL$ | Crystal Oscillator Quiescent Current | – | 5 | – | – | 5 | – | nA | |
| $S_{CLK}$ | Clk_out State in Crystal Disable | 1 | 1 | 1 | 1 | 1 | 1 | | |
| $F_{XTAL}$ | External Crystal Oscillator Frequency | 1 | – | 20 | 1 | – | 20 | MHz | See Figure 74. |
| $T_{SET}$ | 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 203. Low Power 32 kHz Secondary Oscillator Characteristics**

| | | $T_A$ = 0°C to +70°C<br>$T_A$ = –40°C to +105°C | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ = 2.7 to 3.6 V | | | $V_{DD}$ = 1.8 to 2.7 V | | | | |
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| $I_{DD}XTAL2$ | 32 kHz Secondary Oscillator Active Current | – | – | 20 | – | – | 10 | µA | |
| $I_{DDQ}XTAL2$ | 32 kHz Secondary Oscillator Quiescent Current | – | 5 | – | – | 5 | – | nA | |
| $S_{CLK}$ | Clk_out State in Crystal Disable | 1 | 1 | 1 | 1 | 1 | 1 | | |
| $F_{XTAL2}$ | External Crystal Oscillator Frequency | – | 32.768 | – | – | 32.768 | – | kHz | |
| $T_{SET}$ | Startup Time After Enable | – | 400 | 1000 | – | 400 | 1000 | mS | |
| | Clk_out Duty Cycle | 40 | 50 | 60 | 40 | 50 | 60 | % | |
| | Clk_out Jitter | – | 1 | – | – | 1 | – | % | |

## 29.4.1. General Purpose I/O Port Input Data Sample Timing

Figure 75 displays 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.

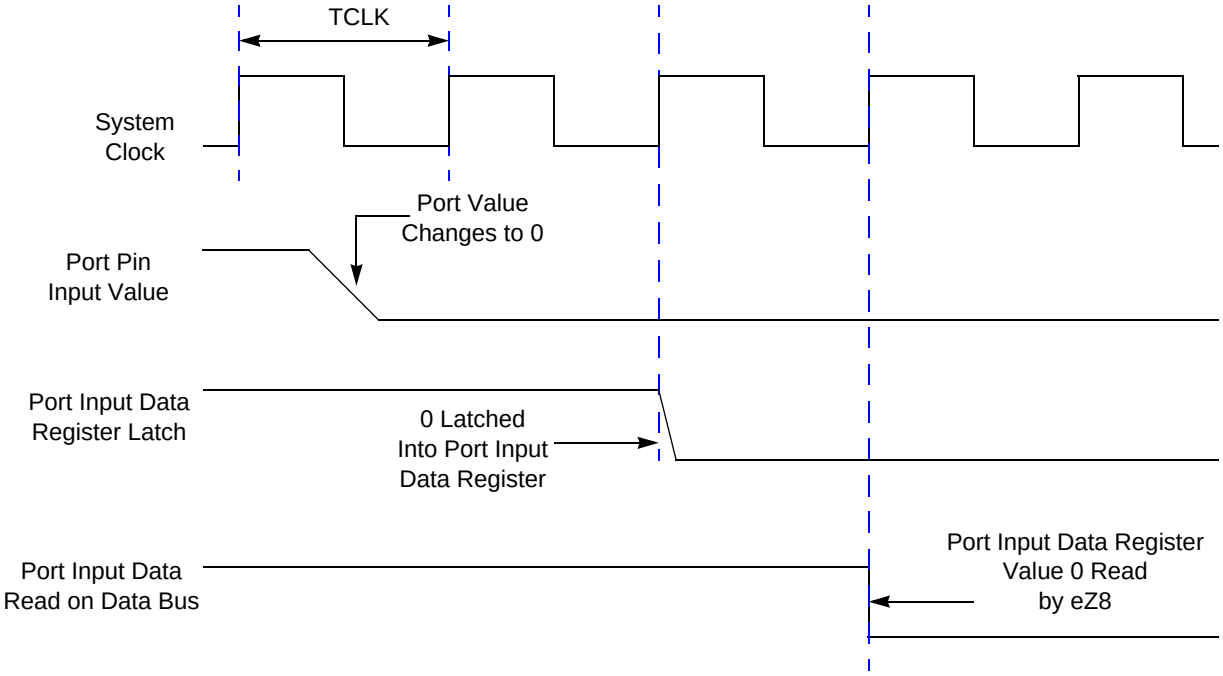



**Figure 75. Port Input Sample Timing**

**Table 204. GPIO Port Input Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **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 µs | |

## 29.4.2. General Purpose I/O Port Output Timing

Figure 76 and Table 205 provide timing information for the GPIO port pins.

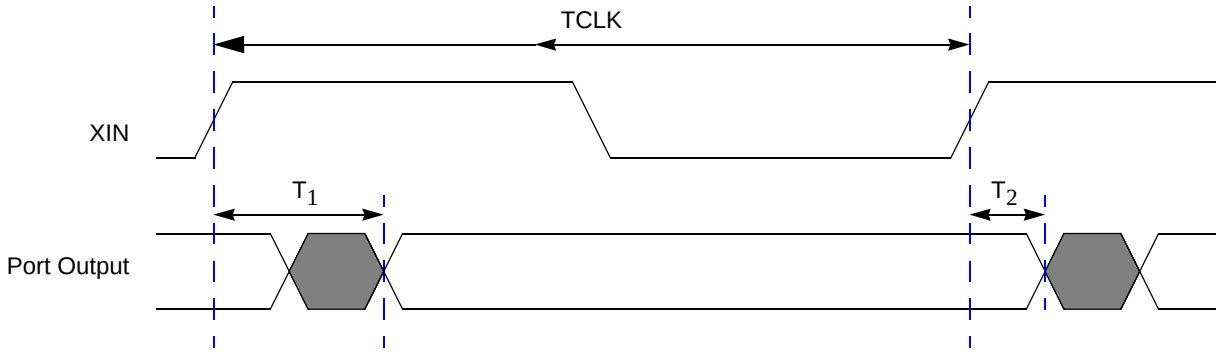



**Figure 76. GPIO Port Output Timing**

**Table 205. GPIO Port Output Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Min** | **Max** |
| **GPIO Port Pins** | | | |
| $T_1$ | XIN Rise to Port Output Valid Delay | – | 15 |
| $T_2$ | XIN Rise to Port Output Hold Time | 2 | – |

## 29.4.3. On-Chip Debugger Timing

Figure 77 and Table 206 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4ns maximum rise and fall time.

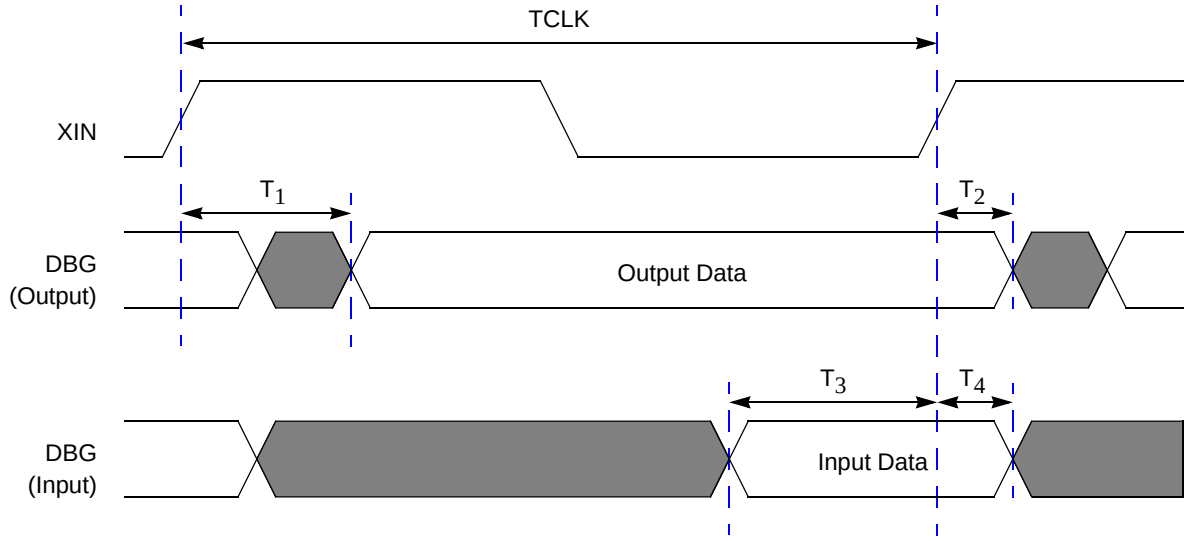



**Figure 77. On-Chip Debugger Timing**

**Table 206. On-Chip Debugger Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Min** | **Max** |
| **DBG** | | | |
| $T_1$ | $X_{IN}$ Rise to DBG Valid Delay | – | 15 |
| $T_2$ | $X_{IN}$ Rise to DBG Output Hold Time | 2 | – |
| $T_3$ | DBG to XIN Rise Input Setup Time | 5 | – |
| $T_4$ | DBG to XIN Rise Input Hold Time | 5 | – |

## 29.4.4. UART Timing

Figure 78 and Table 207 provide timing information for the UART pins for situations in which CTS is used for flow control. The CTS to DE assertion delay (T1) assumes that the Transmit Data Register has been loaded with data prior to CTS assertion.

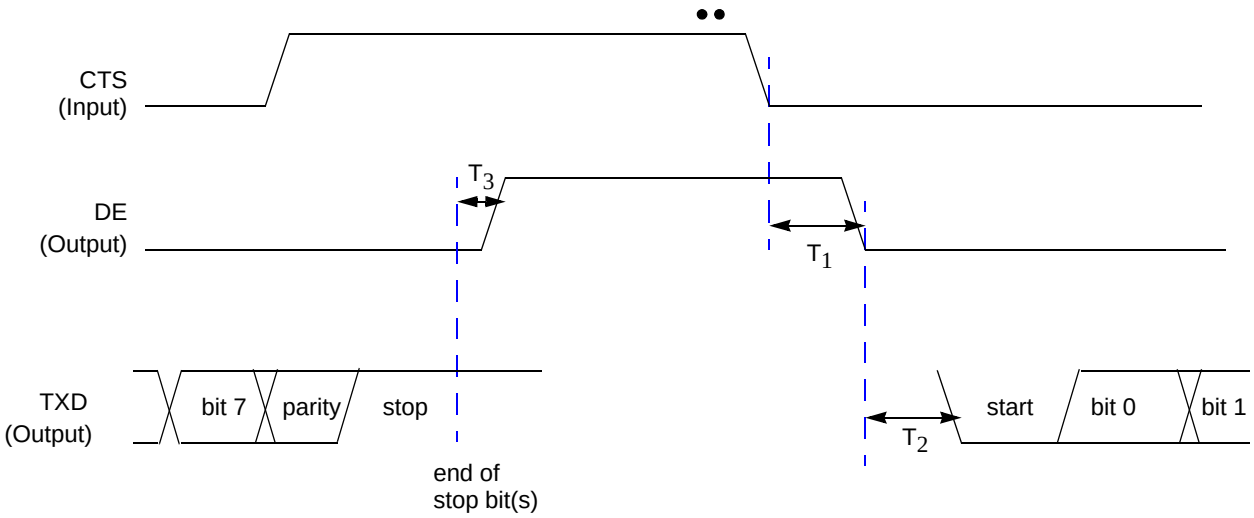



**Figure 78. UART Timing With CTS**

**Table 207. UART Timing with CTS**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Min** | **Max** |
| **UART** | | | |
| $T_1$ | CTS Fall to DE output delay | 2 * $X_{IN}$ period | 2 * $X_{IN}$ period + 1 bit time |
| $T_2$ | DE assertion to TXD falling edge (start bit) delay | ± 5 | |
| $T_3$ | End of stop bit(s) to DE deassertion delay | ± 5 | |

Figure 79 and Table 208 provide timing information for the UART pins for situations in which 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.

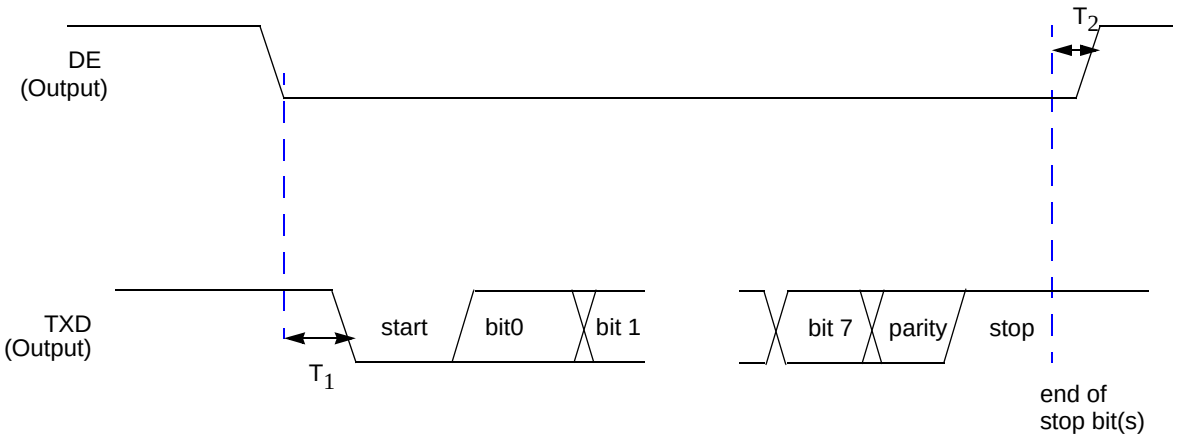



**Figure 79. UART Timing Without CTS**

**Table 208. UART Timing Without CTS**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Min** | **Max** |
| **UART** | | | |
| $T_1$ | DE assertion to TXD falling edge (start bit) delay | 1 * $X_{IN}$ period | 1 bit time |
| $T_2$ | End of stop bit(s) to DE deassertion delay (Tx Data Register is empty) | ± 5 | |

# Chapter 30. Packaging

Zilog's F1680 Series of MCUs includes the Z8F0880, Z8F1680 and Z8F2480 devices, which are available in the following packages:

- 20-pin Plastic Dual-Inline Package (PDIP)
- 20-pin Small Outline Integrated Circuit Package (SOIC)
- 20-pin Small Shrink Outline Package (SSOP)
- 28-pin Plastic Dual-Inline Package (PDIP)
- 28-pin Small Outline Integrated Circuit Package (SOIC)
- 28-pin Small Shrink Outline Package (SSOP)
- 40-pin Plastic Dual-Inline Package (PDIP)
- 44-pin Low-Profile Quad Flat Package (LQFP)
- 44-pin Quad Flat No Lead (QFN)

Current diagrams for each of these packages are published in Zilog's Packaging Product Specification (PS0072), which is available free for download from the Zilog website.

# Chapter 31. Ordering Information

Order the Z8 Encore! XP® F1680 Series from Zilog® using the part numbers listed in Table 209. For more information about ordering, please consult your local Zilog sales office. The Zilog website (www.zilog.com) lists all regional offices and provides additional Z8 Encore! XP product information.

**Table 209. Ordering Information for the Z8 Encore! XP F1680 Series of MCUs**

| Part Number | Flash | Register RAM | Program RAM | NVDS | $I^2C$ | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Z8 Encore! XP F1680 Series with 24KB Flash, 10-Bit Analog-to-Digital Converter** | | | | | | | | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | | | | | | | | |
| Z8F2480SH020SG | 24KB | 2KB | 1KB | 0 | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SOIC 20-pin package |
| Z8F2480HH020SG | 24KB | 2KB | 1KB | 0 | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SSOP 20-pin package |
| Z8F2480PH020SG | 24KB | 2KB | 1KB | 0 | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | PDIP 20-pin package |
| Z8F2480SJ020SG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SOIC 28-pin package |
| Z8F2480HJ020SG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SSOP 28-pin package |
| Z8F2480PJ020SG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | PDIP 28-pin package |
| Z8F2480PM020SG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 33 | 23 | 3 | 8 | 2 | 2 | 1 | 0 | PDIP 40-pin package |
| Z8F2480AN020SG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | LQFP 44-pin package |
| Z8F2480QN020SG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | QFN 44-pin package |
| **Extended Temperature: –40°C to 105°C** | | | | | | | | | | | | | | | |
| Z8F2480SH020EG | 24KB | 2KB | 1KB | 0 | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SOIC 20-pin package |
| Z8F2480HH020EG | 24KB | 2KB | 1KB | 0 | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SSOP 20-pin package |
| Z8F2480PH020EG | 24KB | 2KB | 1KB | 0 | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | PDIP 20-pin package |
| Z8F2480SJ020EG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SOIC 28-pin package |
| Z8F2480HJ020EG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SSOP 28-pin package |
| Z8F2480PJ020EG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | PDIP 28-pin package |
| Z8F2480PM020EG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 33 | 23 | 3 | 8 | 2 | 2 | 1 | 0 | PDIP 40-pin package |
| Z8F2480AN020EG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | LQFP 44-pin package |
| Z8F2480QN020EG | 24KB | 2KB | 1KB | 0 | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | QFN 44-pin package |

**Table 209. Ordering Information for the Z8 Encore! XP F1680 Series of MCUs (Continued)**

| Part Number | Flash | Register RAM | Program RAM | NVDS | I²C | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Z8 Encore! XP F1680 Series with 16KB Flash, 10-Bit Analog-to-Digital Converter** | | | | | | | | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | | | | | | | | |
| Z8F1680SH020SG | 16KB | 2KB | 1KB | 256 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SOIC 20-pin package |
| Z8F1680HH020SG | 16KB | 2KB | 1KB | 256 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SSOP 20-pin package |
| Z8F1680PH020SG | 16KB | 2KB | 1KB | 256 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | PDIP 20-pin package |
| Z8F1680SJ020SG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SOIC 28-pin package |
| Z8F1680HJ020SG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SSOP 28-pin package |
| Z8F1680PJ020SG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | PDIP 28-pin package |
| Z8F1680PM020SG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 33 | 23 | 3 | 8 | 2 | 2 | 1 | 0 | PDIP 40-pin package |
| Z8F1680AN020SG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | LQFP 44-pin package |
| Z8F1680QN020SG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | QFN 44-pin package |
| **Extended Temperature: –40°C to 105°C** | | | | | | | | | | | | | | | |
| Z8F1680SH020EG | 16KB | 2KB | 1KB | 256 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SOIC 20-pin package |
| Z8F1680HH020EG | 16KB | 2KB | 1KB | 256 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SSOP 20-pin package |
| Z8F1680PH020EG | 16KB | 2KB | 1KB | 256 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | PDIP 20-pin package |
| Z8F1680SJ020EG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SOIC 28-pin package |
| Z8F1680HJ020EG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SSOP 28-pin package |
| Z8F1680PJ020EG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | PDIP 28-pin package |
| Z8F1680PM020EG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 33 | 23 | 3 | 8 | 2 | 2 | 1 | 0 | PDIP 40-pin package |
| Z8F1680AN020EG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | LQFP 44-pin package |
| Z8F1680QN020EG | 16KB | 2KB | 1KB | 256 B | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | QFN 44-pin package |

**Table 209. Ordering Information for the Z8 Encore! XP F1680 Series of MCUs (Continued)**

| Part Number | Flash | Register RAM | Program RAM | NVDS | I²C | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Z8 Encore! XP F1680 Series with 8KB Flash, 10-Bit Analog-to-Digital Converter** | | | | | | | | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | | | | | | | | |
| Z8F0880SH020SG | 8KB | 1KB | 1KB | 128 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SOIC 20-pin package |
| Z8F0880HH020SG | 8KB | 1KB | 1KB | 128 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SSOP 20-pin package |
| Z8F0880PH020SG | 8KB | 1KB | 1KB | 128 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | PDIP 20-pin package |
| Z8F0880SJ020SG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SOIC 28-pin package |
| Z8F0880HJ020SG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SSOP 28-pin package |
| Z8F0880PJ020SG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | PDIP 28-pin package |
| Z8F0880PM020SG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 33 | 23 | 3 | 8 | 2 | 2 | 1 | 0 | PDIP 40-pin package |
| Z8F0880AN020SG | 8KB | 1K B | 1KB | 128 B | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | LQFP 44-pin package |
| Z8F0880QN020SG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | QFN 44-pin package |
| **Extended Temperature: –40°C to 105°C** | | | | | | | | | | | | | | | |
| Z8F0880SH020EG | 8KB | 1KB | 1KB | 128 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SOIC 20-pin package |
| Z8F0880HH020EG | 8KB | 1KB | 1KB | 128 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | SSOP 20-pin package |
| Z8F0880PH020EG | 8KB | 1KB | 1KB | 128 B | 1 | 0 | 17 | 20 | 3 | 7 | 1 | 1 | 1 | 0 | PDIP 20-pin package |
| Z8F0880SJ020EG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SOIC 28-pin package |
| Z8F0880HJ020EG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | SSOP 28-pin package |
| Z8F0880PJ020EG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 23 | 21 | 3 | 8 | 1 | 1 | 1 | 0 | PDIP 28-pin package |
| Z8F0880PM020EG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 33 | 23 | 3 | 8 | 2 | 2 | 1 | 0 | PDIP 40-pin package |
| Z8F0880AN020EG | 8KB | 1KB | 1KB | 128 B | 1 | 1 | 37 | 24 | 3 | 8 | 2 | 2 | 1 | 1 | LQFP 44-pin package |
| Z8F0880QN020EG | 8KB | 1KB | 1KB | 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 | | | | | | | | | | | | | | |
| ZUSBSC00100ZACG | USB Smart Cable Accessory Kit | | | | | | | | | | | | | | |
| ZUSBOPTSC01ZACG | USB Opto-Isolated Smart Cable Accessory Kit | | | | | | | | | | | | | | |
| ZENETSC0100ZACG | Ethernet Smart Cable Accessory Kit | | | | | | | | | | | | | | |

# 31.1. Part Number Suffix Designations

Zilog part numbers consist of a number of components, as indicated in the following example.

**Example.** Part number Z8F1680SH020SG is an 8-bit, 20MHz Flash Motor Controller with 16KB of Program memory in a 20-pin SOIC package, operating within a 0ºC to +70ºC temperature range and built using lead-free solder.

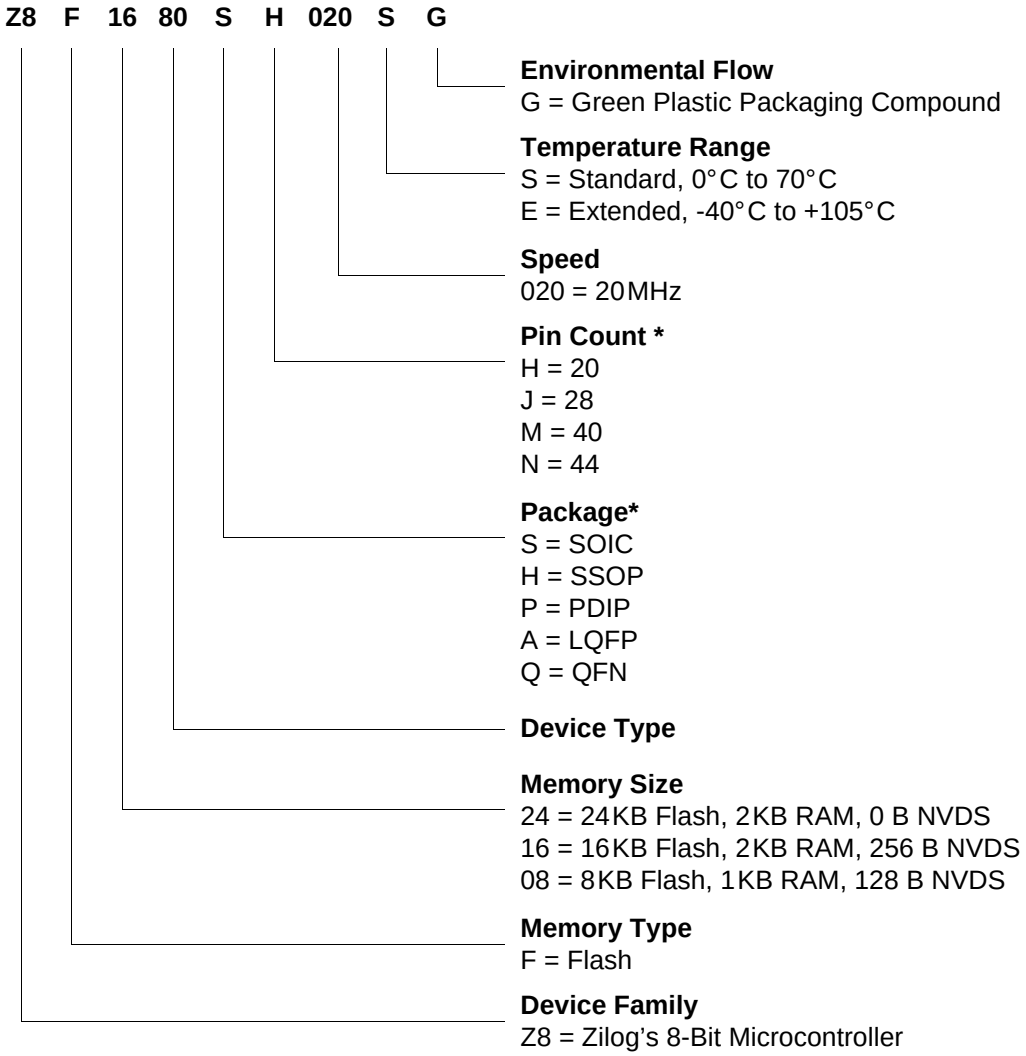


**Table 210. Package and Pin Count Description**

| | Pin Count | | | |
|---|---|---|---|---|
| **Package** | **20** | **28** | **40** | **44** |
| SOIC | √ | √ | | |
| SSOP | √ | √ | | |
| PDIP | √ | √ | √ | |
| LQFP | | | | √ |
| QFN | | | | √ |

### 31.1.0.1. 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


## Numerics

10-bit ADC 4

## A

absolute maximum ratings 349
AC characteristics 357
ADC 332
    block diagram 187
    electrical characteristics and timing 360
    overview 186
ADC Channel Register 1 (ADCCTL) 189
ADC Data High Byte Register (ADCDH) 191
ADC Data Low Bit Register (ADCDL) 192, 193, 194, 195
ADCX 332
ADD 332
add - extended addressing 332
add with carry 332
add with carry - extended addressing 332
additional symbols 331
address space 19
ADDX 332
analog block/PWM signal synchronization 188
analog signals 15
analog-to-digital converter
    overview 186
AND 334
ANDX 334
architecture
    voltage measurements 186
arithmetic instructions 332
assembly language syntax 329

## B

B 331
b 330
baud rate generator, UART 160
BCLR 333
binary number suffix 331
BIT 333
bit 330
    clear 333
    manipulation instructions 333
    set 333
    set or clear 333
    swap 333
    test and jump 335
    test and jump if non-zero 335
    test and jump if zero 335
bit jump and test if non-zero 335
bit swap 335
block diagram 3
block transfer instructions 333
BRK 335
BSET 333
BSWAP 333, 335
BTJ 335
BTJNZ 335
BTJZ 335

## C

calibration and compensation, motor control measurements 189
CALL procedure 335
capture mode 114, 115
capture/compare mode 114
cc 330
CCF 333
characteristics, electrical 349
clear 334
clock phase (SPI) 201
CLR 334
COM 334
compare - extended addressing 332
compare with carry 332
compare with carry - extended addressing 332
complement 334
complement carry flag 333
condition code 330
control register definition, UART 163

control register, I2C 247
Control Registers 19
CP 332
CPC 332
CPCX 332
CPU and peripheral overview 4
CPU control instructions 333
CPX 332
current measurement
  architecture 186
  operation 186
Customer Feedback Form 387

**D**

DA 330, 332
data memory 21
data register, I2C 243
DC characteristics 350
debugger, on-chip 294
DEC 332
decimal adjust 332
decrement 332
decrement and jump non-zero 335
decrement word 332
DECW 332
destination operand 331
device, port availability 46
DI 333
direct address 330
disable interrupts 333
DJNZ 335
dst 331

**E**

EI 333
electrical characteristics 349
  ADC 360
  flash memory and timing 359
  GPIO input data sample timing 366
  watch-dog timer 359, 361
electrical noise 186
enable interrupt 333
ER 330
extended addressing register 330
external pin reset 37
eZ8 CPU features 4
eZ8 CPU instruction classes 331
eZ8 CPU instruction notation 330
eZ8 CPU instruction set 328
eZ8 CPU instruction summary 336

**F**

FCTL register 272, 281
features, Z8 Encore! 1
first opcode map 347
FLAGS 331
flags register 331
flash
  controller 4
  option bit configuration - reset 276
flash memory 262
  arrangement 263, 264, 265
  byte programming 269
  code protection 267
  configurations 262
  control register definitions 271, 278
  controller bypass 270
  electrical characteristics and timing 359
  flash control register 272, 281
  flash option bits 268
  flash status register 272
  flow chart 266
  frequency high and low byte registers 274
  mass erase 270
  operation 265
  operation timing 267
  page erase 270
  page select register 273, 274
FPS register 273, 274
FSTAT register 272

**G**

gated mode 114
general-purpose I/O 46
GPIO 4, 46
  alternate functions 47
  architecture 47
  control register definitions 58

input data sample timing 366
interrupts 58
port A-C pull-up enable sub-registers 63, 64
port A-H address registers 59
port A-H alternate function sub-registers 61
port A-H control registers 60
port A-H data direction sub-registers 60
port A-H high drive enable sub-registers 62
port A-H input data registers 65
port A-H output control sub-registers 62
port A-H output data registers 66
port A-H stop mode recovery sub-registers 63
port availability by device 46
port input timing 366
port output timing 367

**H**

H 331
HALT 333
halt mode 43, 333
hexadecimal number prefix/suffix 331

**I**

I2C 4
10-bit address read transaction 234
10-bit address transaction 231
10-bit addressed slave data transfer format 231, 239
7-bit address transaction 228, 236
7-bit address, reading a transaction 233
7-bit addressed slave data transfer format 230, 238
7-bit receive data transfer format 234, 240, 242
baud high and low byte registers 248, 250, 255
C status register 251
controller 223
controller signals 14
interrupts 226
operation 225
SDA and SCL signals 225
stop and start conditions 228
I2CBRH register 250, 255
I2CCTL register 247
I2CSTAT register 251
IM 330
immediate data 330
immediate operand prefix 331
INC 332
increment 332
increment word 332
INCW 332
indexed 330
indirect address prefix 331
indirect register 330
indirect register pair 330
indirect working register 330
indirect working register pair 330
infrared encoder/decoder (IrDA) 182
Instruction Set 328
instruction set, ez8 CPU 328
instructions
ADC 332
ADCX 332
ADD 332
ADDX 332
AND 334
ANDX 334
arithmetic 332
BCLR 333
BIT 333
bit manipulation 333
block transfer 333
BRK 335
BSET 333
BSWAP 333, 335
BTJ 335
BTJNZ 335
BTJZ 335
CALL 335
CCF 333
CLR 334
COM 334
CP 332
CPC 332
CPCX 332
CPU control 333
CPX 332
DA 332

DEC 332
DECW 332
DI 333
DJNZ 335
EI 333
HALT 333
INC 332
INCW 332
IRET 335
JP 335
LD 334
LDC 334
LDCI 333, 334
LDE 334
LDEI 333
LDX 334
LEA 334
load 334
logical 334
MULT 332
NOP 333
OR 334
ORX 334
POP 334
POPX 334
program control 335
PUSH 334
PUSHX 334
RCF 333
RET 335
RL 335
RLC 335
rotate and shift 335
RR 335
RRC 335
SBC 332
SCF 333
SRA 335
SRL 335
SRP 333
STOP 334
SUB 332
SUBX 332
SWAP 335
TCM 333
TCMX 333
TM 333
TMX 333
TRAP 335
watch-dog timer refresh 334
XOR 334
XORX 334
instructions, eZ8 classes of 331
interrupt control register 83
interrupt controller 68
architecture 68
interrupt assertion types 71
interrupt vectors and priority 71
operation 70
register definitions 72
software interrupt assertion 72
interrupt edge select register 82
interrupt request 0 register 73
interrupt request 1 register 74
interrupt request 2 register 75
interrupt return 335
interrupt vector listing 68
interrupts
SPI 211
UART 157
IR 330
Ir 330
IrDA
architecture 106, 161, 182
block diagram 107, 161, 182
control register definitions 185
operation 107, 161, 182
receiving data 184
transmitting data 183
IRET 335
IRQ0 enable high and low bit registers 76
IRQ1 enable high and low bit registers 77
IRQ2 enable high and low bit registers 79
IRR 330
Irr 330

## J

JP 335

jump, conditional, relative, and relative conditional 335

**L**

LD 334
LDC 334
LDCI 333, 334
LDE 334
LDEI 333, 334
LDX 334
LEA 334
load 334
load constant 333
load constant to/from program memory 334
load constant with auto-increment addresses 334
load effective address 334
load external data 334
load external data to/from data memory and auto-increment addresses 333
load external to/from data memory and auto-increment addresses 334
load instructions 334
load using extended addressing 334
logical AND 334
logical AND/extended addressing 334
logical exclusive OR 334
logical exclusive OR/extended addressing 334
logical instructions 334
logical OR 334
logical OR/extended addressing 334
low power modes 42

**M**

master interrupt enable 70
master-in, slave-out and-in 199
memory
  data 21
  program 20
MISO 199
mode
  capture 114, 115
  capture/compare 114
  gated 114
  PWM 114, 115
MOSI 199
motor control measurements
  ADC Control register definitions 189
  calibration and compensation 189
  interrupts 188, 189
  overview 186
MULT 332
multiply 332
multiprocessor mode, UART 151

**N**

noise, electrical 186
NOP (no operation) 333
notation
  b 330
  cc 330
  DA 330
  ER 330
  IM 330
  IR 330
  Ir 330
  IRR 330
  Irr 330
  p 330
  R 330
  r 330
  RA 330
  RR 330
  rr 330
  vector 330
  X 330
notational shorthand 330

**O**

OCD
  architecture 294
  auto-baud detector/generator 298
  baud rate limits 299
  block diagram 294
  breakpoints 301
  commands 303
  control register 309
  data format 298
  DBG pin to RS-232 Interface 296

debug mode 297
debugger break 335
interface 295
serial errors 300
status register 312
timing 368
OCD commands
execute instruction (12H) 308
read data memory (0DH) 307
read OCD control register (05H) 306
read OCD revision (00H) 305
read OCD status register (02H) 305
read program counter (07H) 306
read program memory (0BH) 307
read program memory CRC (0EH) 308
read register (09H) 306
read runtime counter (03H) 305
step instruction (10H) 308
stuff instruction (11H) 308
write data memory (0CH) 307
write OCD control register (04H) 306
write program counter (06H) 306
write program memory (0AH) 307
write register (08H) 306
on-chip debugger (OCD) 294
on-chip debugger signals 16
on-chip oscillator 321
opcode map
abbreviations 346
cell description 345
first 347
second after 1FH 348
operation 188
current measurement 186
voltage measurement timing diagram 188
Operational Description 31, 42, 46, 68, 84, 120, 140, 144, 182, 186, 195, 196, 256, 260, 262, 276, 290, 294, 315, 321, 327
OR 334
ordering information 372
ORX 334
Oscillator 318
oscillator signals 16

## P

p 330
Packaging 371
part selection guide 2
PC 331
peripheral AC and DC electrical characteristics 358
PHASE=0 timing (SPI) 202
PHASE=1 timing (SPI) 203
pin characteristics 17
Pin Descriptions 10
polarity 330
POP 334
pop using extended addressing 334
POPX 334
port availability, device 46
port input timing (GPIO) 366
port output timing, GPIO 367
power supply signals 16
power-on reset (POR) 34
program control instructions 335
program counter 331
program memory 20
PUSH 334
push using extended addressing 334
PUSHX 334
PWM mode 114, 115
PxADDR register 59
PxCTL register 60

## R

R 330
r 330
RA
register address 330
RCF 333
receive
7-bit data transfer format (I2C) 234, 240, 242
IrDA data 184
receiving UART data-interrupt-driven method 149
receiving UART data-polled method 148
register 217, 330
baud low and high byte (I2C) 248, 250, 255
baud rate high and low byte (SPI) 221
control, I2C 247

data, SPI 213, 214
flash control (FCTL) 272, 281
flash high and low byte (FFREQH and FREEQL) 274
flash page select (FPS) 273, 274
flash status (FSTAT) 272
GPIO port A-H address (PxADDR) 59
GPIO port A-H alternate function sub-registers 61
GPIO port A-H control address (PxCTL) 60
GPIO port A-H data direction sub-registers 60
I2C baud rate high (I2CBRH) 250, 255
I2C control (I2CCTL) 247
I2C status 251
I2C status (I2CSTAT) 251
mode, SPI 217
OCD control 309
OCD status 312
SPI baud rate high byte (SPIBRH) 222
SPI baud rate low byte (SPIBRL) 222
SPI data (SPIDATA) 214
SPI status (SPISTAT) 219
status, SPI 219
UARTx baud rate high byte (UxBRH) 177
UARTx baud rate low byte (UxBRL) 178
UARTx Control 0 (UxCTL0) 170, 177
UARTx control 1 (UxCTL1) 119, 172, 174, 175
UARTx receive data (UxRXD) 164
UARTx status 0 (UxSTAT0) 165, 166
UARTx status 1 (UxSTAT1) 168
UARTx transmit data (UxTXD) 163
watch-dog timer control (WDTCTL) 257, 258, 319, 320
watchdog timer control (WDTCTL) 40
watch-dog timer reload high byte (WDTH) 143
register pair 330
register pointer 331
registers
ADC channel 1 189
ADC data high byte 191
ADC data low bit 192, 193, 194, 195
reset
and stop mode characteristics 32
carry flag 333
sources 33
RET 335
return 335
RL 335
RLC 335
rotate and shift instructions 335
rotate left 335
rotate left through carry 335
rotate right 335
rotate right through carry 335
RP 331
RR 330, 335
rr 330
RRC 335

## S

SBC 332
SCF 333
SCK 199
SDA and SCL (IrDA) signals 225
second opcode map after 1FH 348
serial clock 199
serial peripheral interface (SPI) 197
set carry flag 333
set register pointer 333
shift right arithmetic 335
shift right logical 335
signal descriptions 14
slave data transfer formats (I2C) 231, 239
slave select 200
software trap 335
source operand 331
SP 331
SPI
architecture 197
baud rate generator 212
baud rate high and low byte register 221
clock phase 201
configured as slave 210
control register definitions 213
data register 213, 214
error detection 210
interrupts 211

mode fault error 210
mode register 217
multi-master operation 207
operation 199
overrun error 210, 211
signals 199
single master, multiple slave system 208
single master, single slave system 208
status register 219
timing, PHASE = 0 202
timing, PHASE=1 203
SPI controller signals 14
SPI mode (SPIMODE) 217
SPIBRH register 222
SPIBRL register 222
SPIDATA register 214
SPIMODE register 217
SPISTAT register 219
SRA 335
src 331
SRL 335
SRP 333
SS, SPI signal 199
stack pointer 331
STOP 334
stop mode 42, 334
stop mode recovery
sources 37, 39
using a GPIO port pin transition 39
using watch-dog timer time-out 38
SUB 332
subtract 332
subtract - extended addressing 332
subtract with carry 332
subtract with carry - extended addressing 332
SUBX 332
SWAP 335
swap nibbles 335
symbols, additional 331

## T

TCM 333
TCMX 333
test complement under mask 333
test complement under mask - extended addressing 333
test under mask 333
test under mask - extended addressing 333
timer signals 15
timers 84
architecture 85, 120
block diagram 85, 121
capture mode 97, 114, 115
capture/compare mode 102, 114
compare mode 100
continuous mode 90
counter mode 91, 92
gated mode 100, 114
operating mode 87
PWM mode 93, 95, 114, 115
reading the timer count values 106
reload high and low byte registers 109, 130, 131
timer control register definitions 108
timer output signal operation 106
timers 0-3
control registers 112, 113, 117, 118
high and low byte registers 109, 110, 111, 130
timing diagram, voltage measurement 188
TM 333
TMX 333
transmit
IrDA data 183
transmitting UART data-interrupt-driven method 147
transmitting UART data-polled method 146
TRAP 335
Trim Bit Address Option Bits 281
Trim Bit Address Space 282
Trim Bit Data Option Bits 281

## U

UART 4
architecture 144
baud rate generator 160
baud rates table 179, 180, 181
control register definitions 163
controller signals 14
data format 145

interrupts 157
multiprocessor mode 151
receiving data using interrupt-driven method 149
receiving data using the polled method 148
transmitting data using the interrupt-driven method 147
transmitting data using the polled method 146
x baud rate high and low registers 177
x control 0 and control 1 registers 170, 171
x status 0 and status 1 registers 165, 168
User Option Bits 278
UxBRH register 177
UxBRL register 178
UxCTL0 register 170, 177
UxCTL1 register 119, 172, 174, 175
UxRXD register 164
UxSTAT0 register 165, 166
UxSTAT1 register 168
UxTXD register 163

## V

vector 330
voltage brownout reset (VBR) 35
voltage measurement timing diagram 188

## W

watch-dog timer
approximate time-out delay 141
approximate time-out delays 140
CNTL 35
control register 257, 258, 320
electrical characteristics and timing 361
operation 140
refresh 141
reload unlock sequence 142
reload upper, high and low registers 143
reset 36
reset in normal operation 142
reset in STOP mode 142
time-out response 141
watchdog timer
electrical characteristics and timing 359
refresh 334
WDTCTL register 40, 257, 258, 319, 320
WDTH register 143
working register 330
working register pair 330

## X

X 330
XOR 334
XORX 334

## Z

Z8 Encore!
block diagram 3
features 1
part selection guide 2
Zilog Calibration Option Bits 289

# Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

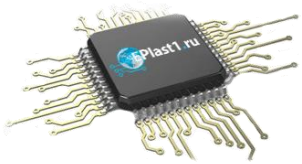
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.




**Как с нами связаться**

**Телефон:** 8 (812) 309 58 32 (многоканальный)
**Факс:** 8 (812) 320-02-42
**Электронная почта:** org@eplast1.ru
**Адрес:** 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.