# *Z8F640x, Z8F480x, Z8F320x, Z8F240x, and Z8F160x*

# *Z8 Encore!™ Microcontrollers with Flash Memory and 10-Bit A/D Converter*

## Product Specification

PS017610-0404

ZiLOG Worldwide Headquarters • 532 Race Street • San Jose, CA 95126-3432
Telephone: 408.558.8500 • Fax: 408.558.8300 • www.ZiLOG.com

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

**ZiLOG Worldwide Headquarters**
532 Race Street
San Jose, CA 95126
Telephone: 408.558.8500
Fax: 408.558.8300
www.ZiLOG.com

## Document Disclaimer

ZiLOG is a registered trademark of ZiLOG Inc. in the United States and in other countries. All other products and/or service names mentioned herein may be trademarks of the companies with which they are associated.

©2004 by ZiLOG, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZiLOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZiLOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. Devices sold by ZiLOG, Inc. are covered by warranty and limitation of liability provisions appearing in the ZiLOG, Inc. Terms and Conditions of Sale. ZiLOG, Inc. makes no warranty of merchantability or fitness for any purpose Except with the express written approval of ZiLOG, use of information, devices, or technology as critical components of life support systems is not authorized. No licenses are conveyed, implicitly or otherwise, by this document under any intellectual property rights.

# *Table of Contents*

**Introduction . . . . . 1**
Features . . . . . 1
Part Selection Guide . . . . . 2
Block Diagram . . . . . 3
CPU and Peripheral Overview . . . . . 3
eZ8 CPU Features . . . . . 3
General Purpose I/O . . . . . 4
Flash Controller . . . . . 4
10-Bit Analog-to-Digital Converter . . . . . 4
UARTs . . . . . 4
$I^2C$ . . . . . 4
Serial Peripheral Interface . . . . . 5
Timers . . . . . 5
Interrupt Controller . . . . . 5
Reset Controller . . . . . 5
On-Chip Debugger . . . . . 5
DMA Controller . . . . . 5
**Signal and Pin Descriptions . . . . . 6**
Overview . . . . . 6
Available Packages . . . . . 6
Pin Configurations . . . . . 7
Signal Descriptions . . . . . 13
Pin Characteristics . . . . . 15
**Address Space . . . . . 17**
Overview . . . . . 17
Register File . . . . . 17
Program Memory . . . . . 18
Data Memory . . . . . 19
**Register File Address Map . . . . . 0**
**Reset and Stop Mode Recovery . . . . . 25**
Overview . . . . . 25
Reset Types . . . . . 25
System and Short Resets . . . . . 26
Reset Sources . . . . . 26
Power-On Reset . . . . . 27
Voltage Brown-Out Reset . . . . . 27
Watch-Dog Timer Reset . . . . . 28

External Pin Reset . . . . . 29
Stop Mode Recovery . . . . . 29
Stop Mode Recovery Using Watch-Dog Timer Time-Out . . . . . 29
Stop Mode Recovery Using a GPIO Port Pin Transition . . . . . 30
**Low-Power Modes . . . . . 31**
Overview . . . . . 31
Stop Mode . . . . . 31
Halt Mode . . . . . 31
**General-Purpose I/O . . . . . 33**
Overview . . . . . 33
GPIO Port Availability By Device . . . . . 33
Architecture . . . . . 34
GPIO Alternate Functions . . . . . 34
GPIO Interrupts . . . . . 36
GPIO Control Register Definitions . . . . . 36
Port A-H Address Registers . . . . . 37
Port A-H Control Registers . . . . . 38
Port A-H Input Data Registers . . . . . 42
Port A-H Output Data Register . . . . . 43
**Interrupt Controller . . . . . 44**
Overview . . . . . 44
Interrupt Vector Listing . . . . . 44
Architecture . . . . . 46
Operation . . . . . 46
Master Interrupt Enable . . . . . 46
Interrupt Vectors and Priority . . . . . 47
Interrupt Assertion Types . . . . . 47
Interrupt Control Register Definitions . . . . . 48
Interrupt Request 0 Register . . . . . 48
Interrupt Request 1 Register . . . . . 49
Interrupt Request 2 Register . . . . . 50
IRQ0 Enable High and Low Bit Registers . . . . . 51
IRQ1 Enable High and Low Bit Registers . . . . . 52
IRQ2 Enable High and Low Bit Registers . . . . . 53
Interrupt Edge Select Register . . . . . 54
Interrupt Port Select Register . . . . . 55
Interrupt Control Register . . . . . 56
**Timers . . . . . 57**
Overview . . . . . 57
Architecture . . . . . 57
Operation . . . . . 58

Timer Operating Modes . . . . . 58
Reading the Timer Count Values . . . . . 66
Timer Output Signal Operation . . . . . 66
Timer Control Register Definitions . . . . . 66
Timer 0-3 High and Low Byte Registers . . . . . 66
Timer Reload High and Low Byte Registers . . . . . 67
Timer 0-3 PWM High and Low Byte Registers . . . . . 69
Timer 0-3 Control Registers . . . . . 70
**Watch-Dog Timer . . . . . 72**
Overview . . . . . 72
Operation . . . . . 72
Watch-Dog Timer Refresh . . . . . 73
Watch-Dog Timer Time-Out Response . . . . . 73
Watch-Dog Timer Reload Unlock Sequence . . . . . 74
Watch-Dog Timer Control Register Definitions . . . . . 75
Watch-Dog Timer Control Register . . . . . 75
Watch-Dog Timer Reload Upper, High and Low Byte Registers . 76
**UART . . . . . 78**
Overview . . . . . 78
Architecture . . . . . 78
Operation . . . . . 79
Data Format . . . . . 79
Transmitting Data using the Polled Method . . . . . 80
Transmitting Data using the Interrupt-Driven Method . . . . . 81
Receiving Data using the Polled Method . . . . . 82
Receiving Data using the Interrupt-Driven Method . . . . . 82
Receiving Data using the Direct Memory Access Controller (DMA) . . . . . 83
Multiprocessor (9-bit) Mode . . . . . 84
UART Interrupts . . . . . 85
UART Baud Rate Generator . . . . . 85
UART Control Register Definitions . . . . . 86
UARTx Transmit Data Register . . . . . 86
UARTx Receive Data Register . . . . . 87
UARTx Status 0 and Status 1 Registers . . . . . 87
UARTx Control 0 and Control 1 Registers . . . . . 89
UARTx Baud Rate High and Low Byte Registers . . . . . 91
**Infrared Encoder/Decoder . . . . . 95**
Overview . . . . . 95
Architecture . . . . . 95
Operation . . . . . 96

Transmitting IrDA Data . . . . . 96
Receiving IrDA Data . . . . . 97
Jitter . . . . . 98
Infrared Encoder/Decoder Control Register Definitions . . . . . 98
**Serial Peripheral Interface . . . . . 99**
Overview . . . . . 99
Architecture . . . . . 99
Operation . . . . . 100
SPI Signals . . . . . 101
SPI Clock Phase and Polarity Control . . . . . 102
Multi-Master Operation . . . . . 104
Error Detection . . . . . 105
SPI Interrupts . . . . . 105
SPI Baud Rate Generator . . . . . 105
SPI Control Register Definitions . . . . . 106
SPI Data Register . . . . . 106
SPI Control Register . . . . . 107
SPI Status Register . . . . . 108
SPI Mode Register . . . . . 109
SPI Baud Rate High and Low Byte Registers . . . . . 110
**I2C Controller . . . . . 111**
Overview . . . . . 111
Operation . . . . . 111
SDA and SCL Signals . . . . . 111
$I^2C$ Interrupts . . . . . 112
Start and Stop Conditions . . . . . 112
Writing a Transaction with a 7-Bit Address . . . . . 112
Writing a Transaction with a 10-Bit Address . . . . . 114
Reading a Transaction with a 7-Bit Address . . . . . 115
Reading a Transaction with a 10-Bit Address . . . . . 116
I2C Control Register Definitions . . . . . 118
I2C Data Register . . . . . 118
I2C Status Register . . . . . 118
I2C Control Register . . . . . 119
I2C Baud Rate High and Low Byte Registers . . . . . 121
**Direct Memory Access Controller . . . . . 122**
Overview . . . . . 122
Operation . . . . . 122
DMA0 and DMA1 Operation . . . . . 122
Configuring DMA0 and DMA1 for Data Transfer . . . . . 123

DMA_ADC Operation . . . . . 123
Configuring DMA_ADC for Data Transfer . . . . . 124
DMA Control Register Definitions . . . . . 124
DMAx Control Register . . . . . 124
DMAx I/O Address Register . . . . . 125
DMAx Address High Nibble Register . . . . . 126
DMAx Start/Current Address Low Byte Register . . . . . 127
DMAx End Address Low Byte Register . . . . . 128
DMA_ADC Address Register . . . . . 128
DMA_ADC Control Register . . . . . 130
DMA Status Register . . . . . 131
**Analog-to-Digital Converter . . . . . 132**
Overview . . . . . 132
Architecture . . . . . 132
Operation . . . . . 133
Automatic Power-Down . . . . . 133
Single-Shot Conversion . . . . . 133
Continuous Conversion . . . . . 134
DMA Control of the ADC . . . . . 135
ADC Control Register Definitions . . . . . 135
ADC Control Register . . . . . 135
ADC Data High Byte Register . . . . . 137
ADC Data Low Bits Register . . . . . 137
**Flash Memory . . . . . 138**
Overview . . . . . 138
Operation . . . . . 139
Flash Operation Timing Using the Flash Frequency Registers . . 141
Flash Code Protection Against External Access . . . . . 141
Flash Code Protection Against Accidental Program and Erasure 141
Byte Programming . . . . . 142
Page Erase . . . . . 143
Mass Erase . . . . . 143
Flash Controller Bypass . . . . . 143
Flash Control Register Definitions . . . . . 144
Flash Control Register . . . . . 144
Flash Status Register . . . . . 145
Flash Page Select Register . . . . . 146
Flash Frequency High and Low Byte Registers . . . . . 147

**Option Bits . . . . . 148**
Overview . . . . . 148
Operation . . . . . 148
Option Bit Configuration By Reset . . . . . 148
Option Bit Address Space . . . . . 148
Program Memory Address 0000H . . . . . 149
Program Memory Address 0001H . . . . . 150
**On-Chip Debugger . . . . . 151**
Overview . . . . . 151
Architecture . . . . . 151
Operation . . . . . 152
OCD Interface . . . . . 152
Debug Mode . . . . . 153
OCD Data Format . . . . . 154
OCD Auto-Baud Detector/Generator . . . . . 154
OCD Serial Errors . . . . . 155
Breakpoints . . . . . 155
Watchpoints . . . . . 155
Runtime Counter . . . . . 156
On-Chip Debugger Commands . . . . . 156
On-Chip Debugger Control Register Definitions . . . . . 161
OCD Control Register . . . . . 161
OCD Status Register . . . . . 162
OCD Watchpoint Control Register . . . . . 163
OCD Watchpoint Address Register . . . . . 164
OCD Watchpoint Data Register . . . . . 164
**On-Chip Oscillator . . . . . 165**
20MHz Crystal Oscillator Operation . . . . . 165
**Electrical Characteristics . . . . . 167**
Absolute Maximum Ratings . . . . . 167
DC Characteristics . . . . . 169
AC Characteristics . . . . . 172
On-Chip Peripheral AC and DC Electrical Characteristics . . . . . 173
General Purpose I/O Port Input Data Sample Timing . . . . . 176
General Purpose I/O Port Output Timing . . . . . 177
On-Chip Debugger Timing . . . . . 178
SPI Master Mode Timing . . . . . 179
SPI Slave Mode Timing . . . . . 180
I2C Timing . . . . . 181
**eZ8 CPU Instruction Set . . . . . 182**
Assembly Language Programming Introduction . . . . . 182

Assembly Language Syntax . . . . . 183
eZ8 CPU Instruction Notation . . . . . 183
Condition Codes . . . . . 186
eZ8 CPU Instruction Classes . . . . . 187
eZ8 CPU Instruction Summary . . . . . 191
Flags Register . . . . . 201
**Opcode Maps . . . . . 202**
**Packaging . . . . . 206**
**Ordering Information . . . . . 211**
Part Number Description . . . . . 214
Precharacterization Product . . . . . 215
**Document Information . . . . . 215**
Document Number Description . . . . . 215
**Customer Feedback Form . . . . . 216**

# List of Figures

Figure 1. Z8 Encore!® Block Diagram . . . . . 3
Figure 2. Z8Fxx01 in 40-Pin Dual Inline Package (DIP) . . . . . 7
Figure 3. Z8Fxx01 in 44-Pin Plastic Leaded Chip Carrier (PLCC) . . 8
Figure 4. Z8Fxx01 in 44-Pin Low-Profile Quad Flat Package (LQFP) 9
Figure 5. Z8Fxx02 in 64-Pin Low-Profile Quad Flat Package (LQFP) 10
Figure 6. Z8Fxx02 in 68-Pin Plastic Leaded Chip Carrier (PLCC) . 11
Figure 7. Z8Fxx03 in 80-Pin Quad Flat Package (QFP) . . . . . 12
Figure 8. Power-On Reset Operation . . . . . 27
Figure 9. Voltage Brown-Out Reset Operation . . . . . 28
Figure 10. GPIO Port Pin Block Diagram . . . . . 34
Figure 11. Interrupt Controller Block Diagram . . . . . 46
Figure 12. Timer Block Diagram . . . . . 58
Figure 13. UART Block Diagram . . . . . 79
Figure 14. UART Asynchronous Data Format without Parity . . . . . 80
Figure 15. UART Asynchronous Data Format with Parity . . . . . 80
Figure 16. UART Asynchronous Multiprocessor (9-bit) Mode Data Format . . . . . 84
Figure 17. Infrared Data Communication System Block Diagram . . . 95
Figure 18. Infrared Data Transmission . . . . . 97
Figure 19. Infrared Data Reception . . . . . 98
Figure 20. SPI Configured as a Master in a Single Master, Single Slave System . . . . . 99
Figure 21. SPI Configured as a Master in a Single Master, Multiple Slave System . . . . . 100
Figure 22. SPI Configured as a Slave . . . . . 100
Figure 23. SPI Timing When PHASE is 0 . . . . . 103
Figure 24. SPI Timing When PHASE is 1 . . . . . 104
Figure 25. 7-Bit Addressed Slave Data Transfer Format . . . . . 113
Figure 26. 10-Bit Addressed Slave Data Transfer Format . . . . . 114
Figure 27. Receive Data Transfer Format for a 7-Bit Addressed Slave . . . . . 115
Figure 28. Receive Data Format for a 10-Bit Addressed Slave . . . . 116
Figure 29. Analog-to-Digital Converter Block Diagram . . . . . 133
Figure 30. Flash Memory Arrangement . . . . . 139

Figure 31. Flash Controller Operation Flow Chart . . . . . 140
Figure 32. On-Chip Debugger Block Diagram . . . . . 151
Figure 33. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (1) . . . . . 152
Figure 34. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (2) . . . . . 153
Figure 35. OCD Data Format . . . . . 154
Figure 36. Recommended Crystal Oscillator Configuration (20MHz operation) . . . . . 166
Figure 37. Nominal ICC Versus System Clock Frequency . . . . . 170
Figure 38. Nominal Halt Mode ICC Versus System Clock Frequency . . . . . 171
Figure 39. Port Input Sample Timing . . . . . 176
Figure 40. GPIO Port Output Timing . . . . . 177
Figure 41. On-Chip Debugger Timing . . . . . 178
Figure 42. SPI Master Mode Timing . . . . . 179
Figure 43. SPI Slave Mode Timing . . . . . 180
Figure 44. $I^2C$ Timing . . . . . 181
Figure 45. Flags Register . . . . . 201
Figure 46. Opcode Map Cell Description . . . . . 202
Figure 47. First Opcode Map . . . . . 204
Figure 48. Second Opcode Map after 1FH . . . . . 205
Figure 49. 40-Lead Plastic Dual-Inline Package (PDIP) . . . . . 206
Figure 50. 44-Lead Low-Profile Quad Flat Package (LQFP) . . . . . 207
Figure 51. 44-Lead Plastic Lead Chip Carrier Package (PLCC) . . . 207
Figure 52. 64-Lead Low-Profile Quad Flat Package (LQFP) . . . . . 208
Figure 53. 68-Lead Plastic Lead Chip Carrier Package (PLCC) . . . 209
Figure 54. 80-Lead Quad-Flat Package (QFP) . . . . . 210

# List of Tables

Table 1. Z8F640x Family Part Selection Guide . . . . . 2
Table 2. Z8F640x Family Package Options . . . . . 6
Table 3. Signal Descriptions . . . . . 13
Table 4. Pin Characteristics of the Z8F640x family . . . . . 15
Table 5. Z8F640x Family Program Memory Maps . . . . . 18
Table 6. Z8F640x Family Data Memory Maps . . . . . 19
Table 7. Register File Address Map . . . . . 20
Table 8. Reset and STOP Mode Recovery Characteristics and Latency . . . . . 25
Table 9. Reset Sources and Resulting Reset Type . . . . . 26
Table 10. STOP Mode Recovery Sources and Resulting Action . . . 29
Table 11. Port Availability by Device and Package Type . . . . . 33
Table 12. Port Alternate Function Mapping . . . . . 35
Table 13. Port A-H GPIO Address Registers (PxADDR) . . . . . 37
Table 14. GPIO Port Registers and Sub-Registers . . . . . 37
Table 15. Port A-H Control Registers (PxCTL) . . . . . 38
Table 16. Port A-H Data Direction Sub-Registers . . . . . 39
Table 17. Port A-H Alternate Function Sub-Registers . . . . . 39
Table 18. Port A-H Output Control Sub-Registers . . . . . 40
Table 19. Port A-H High Drive Enable Sub-Registers . . . . . 41
Table 20. Port A-H Input Data Registers (PxIN) . . . . . 42
Table 21. Port A-H STOP Mode Recovery Source Enable Sub-Registers . . . . . 42
Table 22. Port A-H Output Data Register (PxOUT) . . . . . 43
Table 23. Interrupt Vectors in Order of Priority . . . . . 45
Table 24. Interrupt Request 0 Register (IRQ0) . . . . . 48
Table 25. Interrupt Request 1 Register (IRQ1) . . . . . 49
Table 26. Interrupt Request 2 Register (IRQ2) . . . . . 50
Table 27. IRQ0 Enable and Priority Encoding . . . . . 51
Table 28. IRQ0 Enable High Bit Register (IRQ0ENH) . . . . . 51
Table 29. IRQ0 Enable Low Bit Register (IRQ0ENL) . . . . . 52
Table 30. IRQ1 Enable and Priority Encoding . . . . . 52
Table 31. IRQ1 Enable Low Bit Register (IRQ1ENL) . . . . . 53

Table 32. IRQ2 Enable and Priority Encoding . . . . . 53
Table 33. IRQ1 Enable High Bit Register (IRQ1ENH) . . . . . 53
Table 34. IRQ2 Enable Low Bit Register (IRQ2ENL) . . . . . 54
Table 35. IRQ2 Enable High Bit Register (IRQ2ENH) . . . . . 54
Table 36. Interrupt Edge Select Register (IRQES) . . . . . 55
Table 37. Interrupt Port Select Register (IRQPS) . . . . . 55
Table 38. Interrupt Control Register (IRQCTL) . . . . . 56
Table 39. Timer 0-3 High Byte Register (TxH) . . . . . 67
Table 40. Timer 0-3 Low Byte Register (TxL) . . . . . 67
Table 41. Timer 0-3 Reload High Byte Register (TxRH) . . . . . 68
Table 42. Timer 0-3 Reload Low Byte Register (TxRL) . . . . . 68
Table 43. Timer 0-3 PWM High Byte Register (TxPWMH) . . . . . 69
Table 44. Timer 0-3 PWM Low Byte Register (TxPWML) . . . . . 69
Table 45. Timer 0-3 Control Register (TxCTL) . . . . . 70
Table 46. Watch-Dog Timer Approximate Time-Out Delays . . . . . 73
Table 47. Watch-Dog Timer Control Register (WDTCTL) . . . . . 75
Table 48. Watch-Dog Timer Reload Upper Byte Register (WDTU) 76
Table 49. Watch-Dog Timer Reload High Byte Register (WDTH) . 76
Table 50. Watch-Dog Timer Reload Low Byte Register (WDTL) . . 77
Table 51. UARTx Transmit Data Register (UxTXD) . . . . . 86
Table 52. UARTx Receive Data Register (UxRXD) . . . . . 87
Table 53. UARTx Status 0 Register (UxSTAT0) . . . . . 87
Table 54. UARTx Control 0 Register (UxCTL0) . . . . . 89
Table 55. UARTx Status 1 Register (UxSTAT1) . . . . . 89
Table 56. UARTx Control 1 Register (UxCTL1) . . . . . 90
Table 57. UARTx Baud Rate High Byte Register (UxBRH) . . . . . 91
Table 58. UARTx Baud Rate Low Byte Register (UxBRL) . . . . . 92
Table 59. UART Baud Rates . . . . . 93
Table 60. SPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation . . . . . 102
Table 61. SPI Data Register (SPIDATA) . . . . . 106
Table 62. SPI Control Register (SPICTL) . . . . . 107
Table 63. SPI Status Register (SPISTAT) . . . . . 108
Table 64. SPI Mode Register (SPIMODE) . . . . . 109
Table 65. SPI Baud Rate High Byte Register (SPIBRH) . . . . . 110
Table 66. SPI Baud Rate Low Byte Register (SPIBRL) . . . . . 110

Table 67. I2C Data Register (I2CDATA) . . . . . 118
Table 68. I2C Status Register (I2CSTAT) . . . . . 118
Table 69. I2C Control Register (I2CCTL) . . . . . 119
Table 70. I2C Baud Rate High Byte Register (I2CBRH) . . . . . 121
Table 71. I2C Baud Rate Low Byte Register (I2CBRL) . . . . . 121
Table 72. DMAx Control Register (DMAxCTL) . . . . . 124
Table 73. DMAx I/O Address Register (DMAxIO) . . . . . 126
Table 74. DMAx Address High Nibble Register (DMAxH) . . . . . 126
Table 75. DMAx End Address Low Byte Register (DMAxEND) . 128
Table 76. DMAx Start/Current Address Low Byte Register (DMAxSTART) . . . . . 128
Table 77. DMA_ADC Register File Address Example . . . . . 129
Table 78. DMA_ADC Address Register (DMAA_ADDR) . . . . . 129
Table 79. DMA_ADC Control Register (DMAACTL) . . . . . 130
Table 80. DMA_ADC Status Register (DMAA_STAT) . . . . . 131
Table 81. ADC Control Register (ADCCTL) . . . . . 135
Table 82. ADC Data High Byte Register (ADCD_H) . . . . . 137
Table 83. ADC Data Low Bits Register (ADCD_L) . . . . . 137
Table 84. Z8F640x family Flash Memory Configurations . . . . . 138
Table 85. Flash Code Protection Using the Option Bits . . . . . 142
Table 86. Flash Control Register (FCTL) . . . . . 144
Table 87. Flash Status Register (FSTAT) . . . . . 145
Table 88. Flash Page Select Register (FPS) . . . . . 146
Table 89. Flash Frequency High Byte Register (FFREQH) . . . . . 147
Table 90. Flash Frequency Low Byte Register (FFREQL) . . . . . 147
Table 91. Option Bits At Program Memory Address 0000H . . . . . 149
Table 92. Options Bits at Program Memory Address 0001H . . . . . 150
Table 93. OCD Baud-Rate Limits . . . . . 154
Table 94. On-Chip Debugger Commands . . . . . 156
Table 95. OCD Control Register (OCDCTL) . . . . . 161
Table 96. OCD Status Register (OCDSTAT) . . . . . 162
Table 97. OCD Watchpoint Control/Address (WPTCTL) . . . . . 163
Table 98. OCD Watchpoint Address (WPTADDR) . . . . . 164
Table 99. OCD Watchpoint Data (WPTDATA) . . . . . 164
Table 100. Recommended Crystal Oscillator Specifications (20MHz Operation) . . . . . 166

Table 101. Absolute Maximum Ratings . . . . . 167
Table 102. DC Characteristics . . . . . 169
Table 103. AC Characteristics . . . . . 172
Table 104. Power-On Reset and Voltage Brown-Out Electrical Characteristics and Timing . . . . . 173
Table 105. Flash Memory Electrical Characteristics and Timing . . . 173
Table 106. Watch-Dog Timer Electrical Characteristics and Timing 174
Table 107. Analog-to-Digital Converter Electrical Characteristics and Timing . . . . . 174
Table 108. GPIO Port Input Timing . . . . . 176
Table 109. GPIO Port Output Timing . . . . . 177
Table 110. On-Chip Debugger Timing . . . . . 178
Table 111. SPI Master Mode Timing . . . . . 179
Table 112. SPI Slave Mode Timing . . . . . 180
Table 113. I2C Timing . . . . . 181
Table 114. Assembly Language Syntax Example 1 . . . . . 183
Table 115. Assembly Language Syntax Example 2 . . . . . 183
Table 116. Notational Shorthand . . . . . 184
Table 117. Additional Symbols . . . . . 185
Table 118. Condition Codes . . . . . 186
Table 119. Arithmetic Instructions . . . . . 187
Table 120. Bit Manipulation Instructions . . . . . 188
Table 121. Block Transfer Instructions . . . . . 188
Table 122. CPU Control Instructions . . . . . 189
Table 123. Load Instructions . . . . . 189
Table 124. Logical Instructions . . . . . 190
Table 125. Program Control Instructions . . . . . 190
Table 126. Rotate and Shift Instructions . . . . . 191
Table 127. eZ8 CPU Instruction Summary . . . . . 191
Table 128. Opcode Map Abbreviations . . . . . 203
Table 129. Ordering Information . . . . . 211

# *Manual Objectives*

This Product Specification provides detailed operating information for the Z8F640x, Z8F480x, Z8F320x, Z8F240x, and Z8F160x devices within the Z8 Encore!™ Microcontroller (MCU) family of products. Within this document, the Z8F640x, Z8F480x, Z8F320x, Z8F240x, and Z8F160x are referred to collectively as Z8 Encore!™ or the Z8F640x family unless specifically stated otherwise.

## About This Manual

ZiLOG recommends that the user read and understand everything in this manual before setting up and using the product. However, we recognize that there are different styles of learning. Therefore, we have designed this Product Specification to be used either as a *how to* procedural manual or a reference guide to important data.

## Intended Audience

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

## Manual Conventions

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

### Courier Typeface

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

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

### Hexadecimal Values

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

- Example: R1 is set to `F8H`.

### Brackets

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

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

### Braces

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

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

### Parentheses

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

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

### Parentheses/Bracket Combinations

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

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

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

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

### Notation for Bits and Similar Registers

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

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

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

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

### Use of Initial Uppercase Letters

Initial uppercase letters designate settings, modes, and conditions in general text.

- Example 1: Stop mode.
- Example 2: The receiver forces the SCL line to Low.
- The Master can generate a Stop condition to abort the transfer.

### Use of All Uppercase Letters

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

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

### Bit Numbering

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

## Safeguards

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

**Caution:** Indicates a procedure or file may become corrupted if the user does not follow directions.

## Trademarks

ZiLOG, eZ8, Z8 Encore!, and Z8 are trademarks of ZiLOG, Inc. in the U.S.A. and other countries. All other trademarks are the property of their respective corporations.

# Introduction

The Z8 Encore!® MCU family of products are the first in a line of ZiLOG microcontroller products based upon the new 8-bit eZ8 CPU. The Z8F640x/Z8F480x/Z8F320x/Z8F240x/Z8F160x products are referred to collectively as either Z8 Encore!® or the Z8F640x family. The Z8F640x family of products introduce Flash memory to ZiLOG's extensive line of 8-bit microcontrollers. The Flash in-circuit programming capability allows for faster development time and program changes in the field. The new eZ8 CPU is upward compatible with existing Z8 instructions. The rich peripheral set of the Z8F640x family makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

## Features

- eZ8 CPU, 20 MHz operation
- 12-channel, 10-bit analog-to-digital converter (ADC)
- 3-channel DMA
- Up to 64KB Flash memory with in-circuit programming capability
- Up to 4KB register RAM
- Serial communication protocols
  - Serial Peripheral Interface
  - $I^2C$
- Two full-duplex 9-bit UARTs
- 24 interrupts with programmable priority
- Three or four 16-bit timers with capture, compare, and PWM capability
- Single-pin On-Chip Debugger
- Two Infrared Data Association (IrDA)-compliant infrared encoder/decoders integrated with the UARTs
- Watch-Dog Timer (WDT) with internal RC oscillator
- Up to 60 I/O pins
- Voltage Brown-out Protection (VBO)

- Power-On Reset (POR)
- 3.0-3.6V operating voltage with 5V-tolerant inputs
- 0° to +70°C standard temperature and -40° to +105°C extended temperature operating ranges

## Part Selection Guide

Table 1 identifies the basic features and package styles available for each device within the Z8F640x family product line.

**Table 1. Z8F640x Family Part Selection Guide**

| Part Number | Flash (KB) | RAM (KB) | I/O | 16-bit Timers with PWM | ADC Inputs | UARTs with IrDA | $I^2C$ | SPI | 40/44-pin packages | 64/68-pin packages | 80-pin package |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Z8F1601 | 16 | 2 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F1602 | 16 | 2 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F2401 | 24 | 2 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F2402 | 24 | 2 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F3201 | 32 | 2 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F3202 | 32 | 2 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F4801 | 48 | 4 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F4802 | 48 | 4 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F4803 | 48 | 4 | 60 | 4 | 12 | 2 | 1 | 1 | | | X |
| Z8F6401 | 64 | 4 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F6402 | 64 | 4 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F6403 | 64 | 4 | 60 | 4 | 12 | 2 | 1 | 1 | | | X |

## Block Diagram

Figure 55 illustrates the block diagram of the architecture of the Z8 Encore!™.

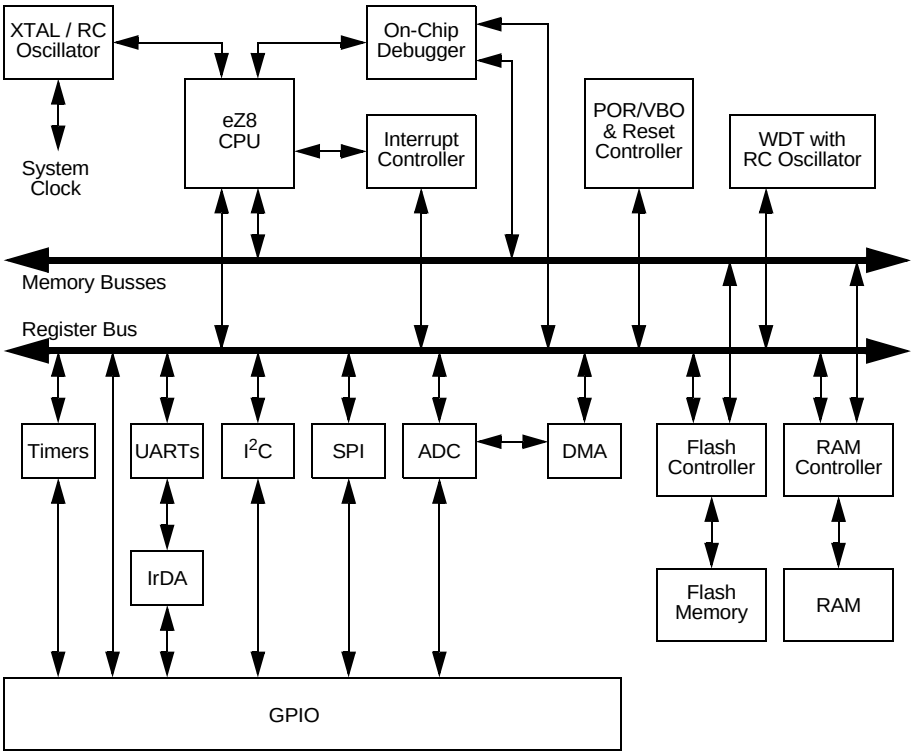



**Figure 55. Z8 Encore!® Block Diagram**

## CPU and Peripheral Overview

### eZ8 CPU Features

The eZ8, ZiLOG's latest 8-bit Central Processing Unit (CPU), meets the continuing demand for faster and more code-efficient microcontrollers. The eZ8 CPU executes a superset of the original Z8 instruction set. The eZ8 CPU features include:

- Direct register-to-register architecture allows each register to function as an accumulator, improving execution time and decreasing the required program memory

- Software stack allows much greater depth in subroutine calls and interrupts than hardware stacks
- Compatible with existing Z8 code
- Expanded internal Register File allows access of up to 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
- 2-9 clock cycles per instruction

For more information regarding the eZ8 CPU, refer to the *eZ8 CPU User Manual* available for download at www.zilog.com.

### General Purpose I/O

The Z8 Encore!® features seven 8-bit ports (Ports A-G) and one 4-bit port (Port H) for general purpose I/O (GPIO). Each pin is individually programmable.

### Flash Controller

The Flash Controller programs and erases the Flash memory.

### 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 accepts inputs from up to 12 different analog input sources.

### UARTs

Each UART is full-duplex and capable of handling asynchronous data transfers. The UARTs support 8- and 9-bit data modes and selectable parity.

### $I^2C$

The inter-integrated circuit ($I^2C$®) controller makes the Z8 Encore!® compatible with the $I^2C$ protocol. The $I^2C$ controller consists of two bidirectional bus lines, a serial data (SDA) line and a serial clock (SCL) line.

## Serial Peripheral Interface

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

## Timers

Up to four 16-bit reloadable timers can be used for timing/counting events or for motor control operations. These timers provide a 16-bit programmable reload counter and operate in One-Shot, Continuous, Gated, Capture, Compare, Capture and Compare, and PWM modes. Only 3 timers (Timers 0-2) are available in the 40- and 44-pin packages.

## Interrupt Controller

The Z8F640x family products support up to 24 interrupts. These interrupts consist of 12 internal and 12 general-purpose I/O pins. The interrupts have 3 levels of programmable interrupt priority.

## Reset Controller

The Z8F640x family can be reset using the $\overline{\text{RESET}}$ pin, power-on reset, Watch-Dog Timer (WDT), Stop mode exit, or Voltage Brown-Out (VBO) warning signal.

## On-Chip Debugger

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

## DMA Controller

The Z8F640x family features three channels of DMA. Two of the channels are for register RAM to and from I/O operations. The third channel automatically controls the transfer of data from the ADC to the memory.

# Signal and Pin Descriptions

## Overview

The Z8F640x family products are available in a variety of packages styles and pin configurations. This chapter describes the signals and available pin configurations for each of the package styles. For information regarding the physical package specifications, please refer to the chapter Packaging on page 206.

## Available Packages

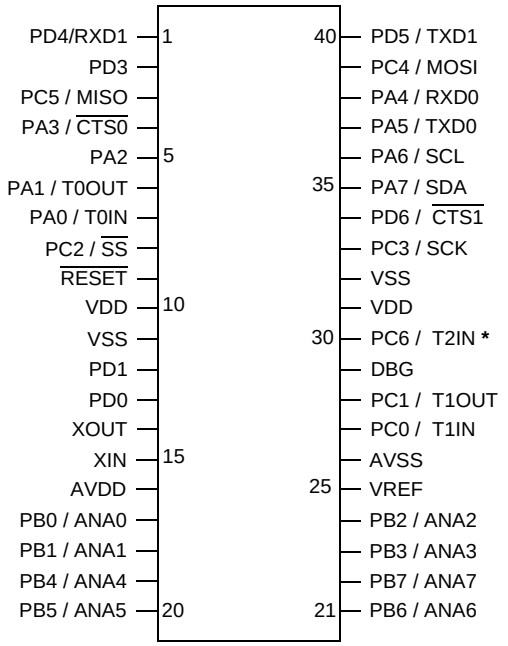
Table 2 identifies the package styles that are available for each device within the Z8F640x family product line.

**Table 2. Z8F640x family Package Options**

| Part Number | 40-pin PDIP | 44-pin LQFP | 44-pin PLCC | 64-pin LQFP | 68-pin PLCC | 80-pin QFP |
|---|---|---|---|---|---|---|
| Z8F1601 | X | X | X | | | |
| Z8F1602 | | | | X | X | |
| Z8F2401 | X | X | X | | | |
| Z8F2402 | | | | X | X | |
| Z8F3201 | X | X | X | | | |
| Z8F3202 | | | | X | X | |
| Z8F4801 | X | X | X | | | |
| Z8F4802 | | | | X | X | |
| Z8F4803 | | | | | | X |
| Z8F6401 | X | X | X | | | |
| Z8F6402 | | | | X | X | |
| Z8F6403 | | | | | | X |

## Pin Configurations

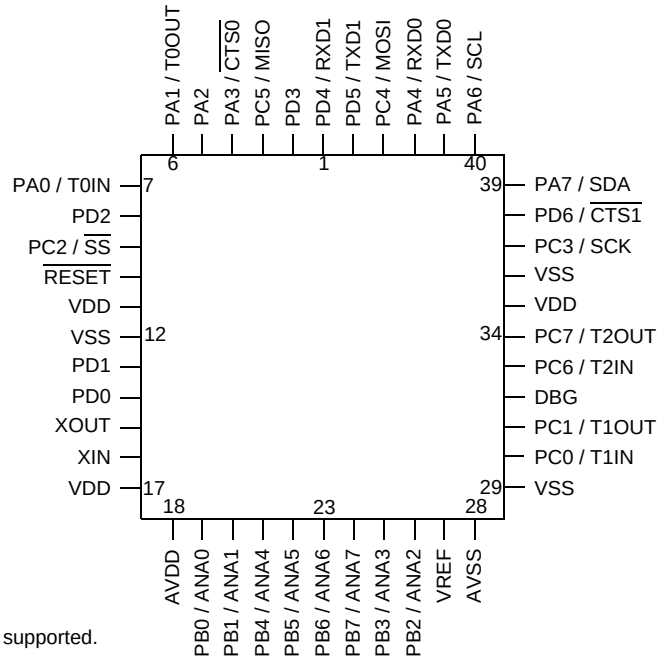
Figures 56 through 61 illustrate the pin configurations for all of the packages available in the Z8 Encore!® MCU family. Refer to Table 2 for a description of the signals.




**Note:** Timer 3 is not supported.

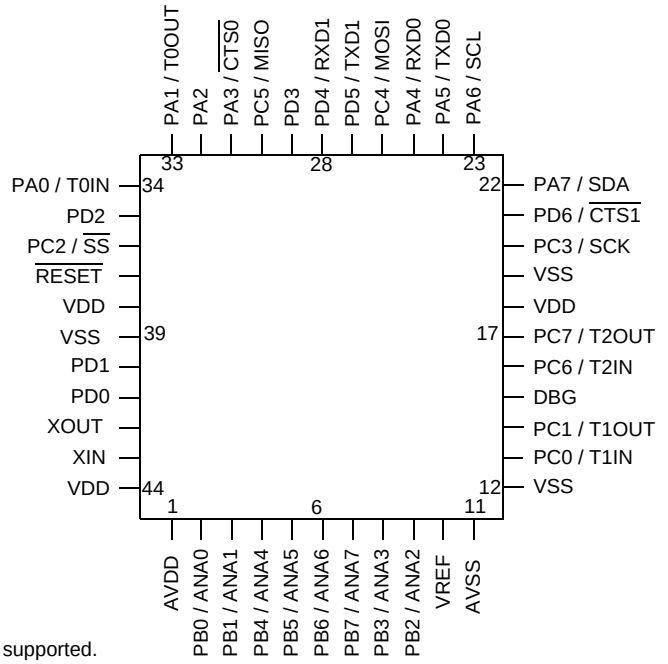
***** T2OUT is not supported.

**Figure 56. Z8Fxx01 in 40-Pin Dual Inline Package (DIP)**




**Note:** Timer 3 is not supported.

**Figure 57. Z8Fxx01 in 44-Pin Plastic Leaded Chip Carrier (PLCC)**




**Note:** Timer 3 is not supported.

**Figure 58. Z8Fxx01 in 44-Pin Low-Profile Quad Flat Package (LQFP)**

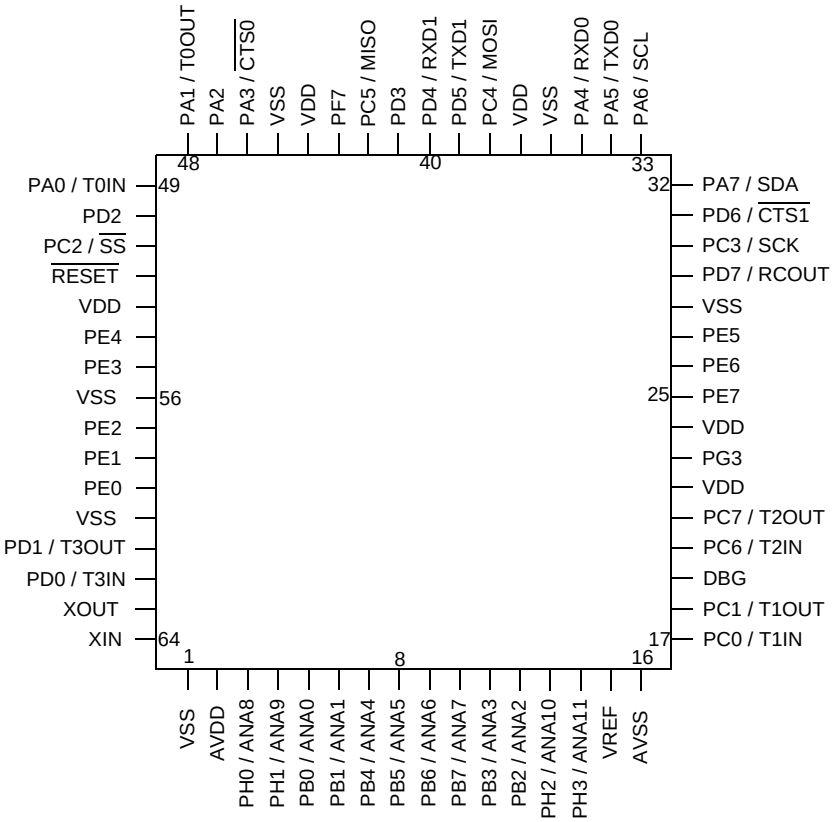



**Figure 59. Z8Fxx02 in 64-Pin Low-Profile Quad Flat Package (LQFP)**

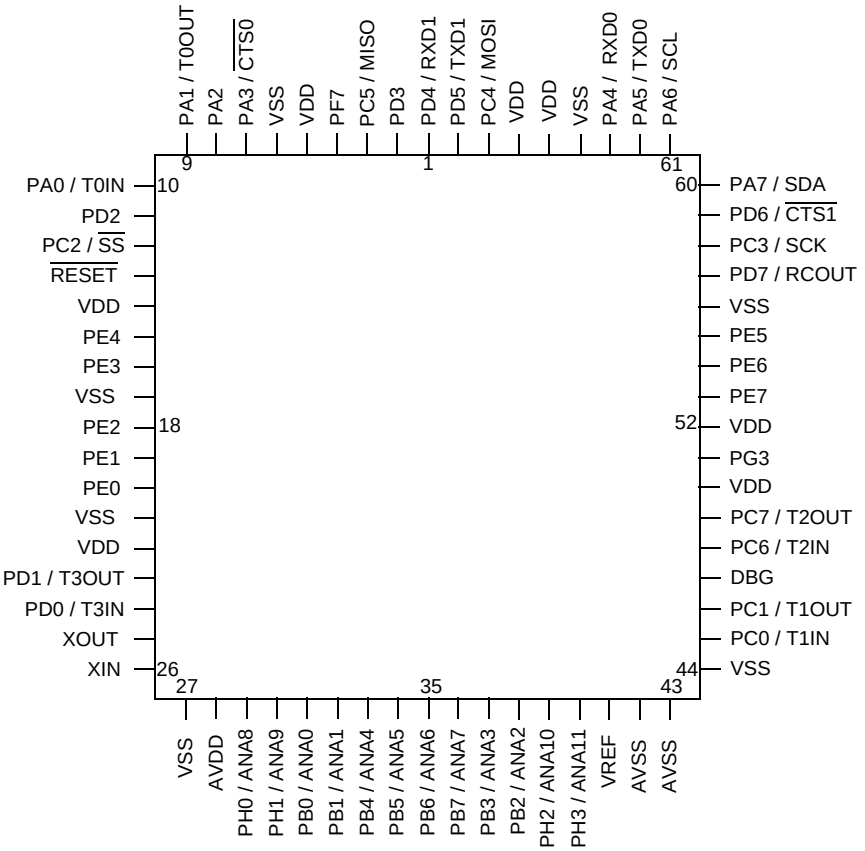



**Figure 60. Z8Fxx02 in 68-Pin Plastic Leaded Chip Carrier (PLCC)**

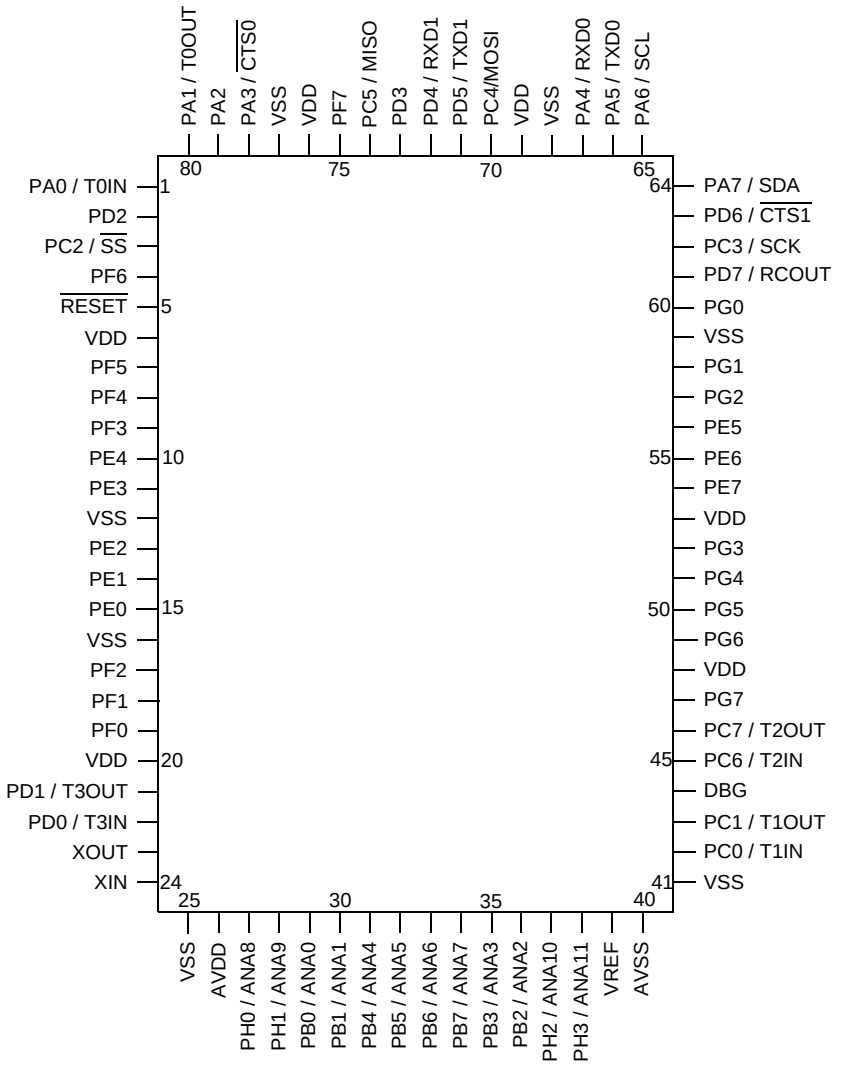



**Figure 61. Z8Fxx03 in 80-Pin Quad Flat Package (QFP)**

## Signal Descriptions

Table 2 describes the Z8F640x family signals. Refer to the section **Pin Configurations on page 7** to determine the signals available for the specific package styles.

**Table 2. Signal Descriptions**

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **General-Purpose I/O Ports A-H** | | |
| PA[7:0] | I/O | Port A[7:0]. These pins are used for general-purpose I/O. |
| PB[7:0] | I/O | Port B[7:0]. These pins are used for general-purpose I/O. |
| PC[7:0] | I/O | Port C[7:0]. These pins are used for general-purpose I/O. |
| PD[7:0] | I/O | Port D[7:0]. These pins are used for general-purpose I/O. |
| PE[7:0] | I/O | Port E[7:0]. These pins are used for general-purpose I/O. |
| PF[7:0] | I/O | Port F[7:0]. These pins are used for general-purpose I/O. |
| PG[7:0] | I/O | Port G[7:0]. These pins are used for general-purpose I/O. |
| PH[3:0] | I/O | Port H[3:0]. These pins are used for general-purpose I/O. |
| **$I^2C$ Controller** | | |
| SCL | O | Serial Clock. This is the output clock for the $I^2C$. This pin is multiplexed with a general-purpose I/O pin. When the general-purpose I/O pin is configured for alternate function to enable the SCL function, this pin is open-drain. |
| SDA | I/O | Serial Data. This open-drain pin is used to transfer data between the $I^2C$ and a slave. This pin is multiplexed with a general-purpose I/O pin. When the general-purpose I/O pin is configured for alternate function to enable the SDA function, this pin is open-drain. |
| **SPI Controller** | | |
| $\overline{SS}$ | I/O | Slave Select. This signal can be an output or an input. If the Z8 Encore! is the SPI master, this pin may be configured as the Slave Select output. If the Z8 Encore! is the SPI slave, this pin is the input slave select. It is multiplexed with a general-purpose I/O pin. |
| SCK | I/O | SPI Serial Clock. The SPI master supplies this pin. If the Z8 Encore! is the SPI master, this pin is an output. If the Z8 Encore! is the SPI slave, this pin is an input. It is multiplexed with a general-purpose I/O pin. |
| MOSI | I/O | Master Out Slave In. This signal is the data output from the SPI master device and the data input to the SPI slave device. It is multiplexed with a general-purpose I/O pin. |
| MISO | I/O | Master In Slave Out. This pin is the data input to the SPI master device and the data output from the SPI slave device. It is multiplexed with a general-purpose I/O pin. |

**Table 2. Signal Descriptions (Continued)**

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **UART Controllers** | | |
| TXD0 / TXD1 | O | Transmit Data. These signals are the transmit outputs from the UARTs. The TXD signals are multiplexed with general-purpose I/O pins. |
| RXD0 / RXD1 | I | Receive Data. These signals are the receiver inputs for the UARTs and IrDAs. The RXD signals are multiplexed with general-purpose I/O pins. |
| $\overline{\text{CTS0}}$ / $\overline{\text{CTS1}}$ | I | Clear To Send. These signals are control inputs for the UARTs. The $\overline{\text{CTS}}$ signals are multiplexed with general-purpose I/O pins. |
| **Timers (Timer 3 is unavailable in the 40-and 44-pin packages)** | | |
| T0OUT / T1OUT/ T2OUT / T3OUT | O | Timer Output 0-3. These signals are output pins from the timers. The Timer Output signals are multiplexed with general-purpose I/O pins. T2OUT is not supported in the 40-pin package. T3OUT is not supported in the 40- and 44-pin packages. |
| T0IN / T1IN/ T2IN / T3IN | I | Timer Input 0-3. These signals are used as the capture, gating and counter inputs. The Timer Input signals are multiplexed with general-purpose I/O pins. T3IN is not supported in the 40- and 44-pin packages. |
| **Analog** | | |
| ANA[11:0] | I | Analog Input. These signals are inputs to the analog-to-digital converter (ADC). The ADC analog inputs are multiplexed with general-purpose I/O pins. |
| VREF | I | Analog-to-digital converter reference voltage input. The VREF pin should be left unconnected (or capacitively coupled to analog ground) if the internal voltage reference is selected as the ADC reference voltage. |
| **Oscillators** | | |
| XIN | I | External Crystal Input. This is the input pin to the crystal oscillator. A crystal can be connected between it and the XOUT pin to form the oscillator. |
| 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. When the system clock is referred to in this manual, it refers to the frequency of the signal at this pin. |
| RCOUT | O | RC Oscillator Output. This signal is the output of the RC oscillator. It is multiplexed with a general-purpose I/O pin. |
| **On-Chip Debugger** | | |
| DBG | I/O | Debug. This pin is the control and data input and output to and from the On-Chip Debugger. For operation of the On-chip debugger, all power pins ($V_{DD}$ and $AV_{DD}$ must be supplied with power, and all ground pins ($V_{SS}$ and $AV_{SS}$ must be grounded. This pin is open-drain and must have an external pull-up resistor to ensure proper operation. |

**Table 2. Signal Descriptions (Continued)**

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **Reset** | | |
| $\overline{\text{RESET}}$ | I | RESET. Generates a Reset when asserted (driven Low). |
| **Power Supply** | | |
| VDD | I | Power Supply. |
| AVDD | I | Analog Power Supply. |
| VSS | I | Ground. |
| AVSS | I | Analog Ground. |

## Pin Characteristics

Table 3 provides detailed information on the characteristics for each pin available on the Z8F640x family products. Data in Table 3 is sorted alphabetically by the pin symbol mnemonic.

**Table 3. Pin Characteristics of the Z8F640x family**

| Symbol Mnemonic | Direction | Reset Direction | Active Low or Active High | Tri-State Output | Internal Pull-up or Pull-down | Schmitt Trigger Input | Open Drain Output |
|---|---|---|---|---|---|---|---|
| AVSS | N/A | N/A | N/A | N/A | No | No | N/A |
| AVDD | N/A | N/A | N/A | N/A | No | No | N/A |
| DBG | I/O | I | N/A | Yes | No | Yes | Yes |
| VSS | N/A | N/A | N/A | N/A | No | No | N/A |
| PA[7:0] | I/O | I | N/A | Yes | No | Yes | Yes, Programmable |
| PB[7:0] | I/O | I | N/A | Yes | No | Yes | Yes, Programmable |
| PC[7:0] | I/O | I | N/A | Yes | No | Yes | Yes, Programmable |
| PD[7:0] | I/O | I | N/A | Yes | No | Yes | Yes, Programmable |
| PE7:0] | I/O | I | N/A | Yes | No | Yes | Yes, Programmable |

*x* represents integer 0, 1,... to indicate multiple pins with symbol mnemonics that differ only by the integer

**Table 3. Pin Characteristics of the Z8F640x family**

| Symbol Mnemonic | Direction | Reset Direction | Active Low or Active High | Tri-State Output | Internal Pull-up or Pull-down | Schmitt Trigger Input | Open Drain Output |
|---|---|---|---|---|---|---|---|
| PF[7:0] | I/O | I | N/A | Yes | No | Yes | Yes, Programmable |
| PG[7:0] | I/O | I | N/A | Yes | No | Yes | Yes, Programmable |
| PH[3:0] | I/O | I | N/A | Yes | No | Yes | Yes, Programmable |
| $\overline{\text{RESET}}$ | I | I | Low | N/A | Pull-up | Yes | N/A |
| VDD | N/A | N/A | N/A | N/A | No | No | N/A |
| XIN | I | I | N/A | N/A | No | No | N/A |
| XOUT | O | O | N/A | Yes, in Stop mode | No | No | No |

*x* represents integer 0, 1,... to indicate multiple pins with symbol mnemonics that differ only by the integer

# Address Space

## Overview

The eZ8 CPU can access three distinct address spaces:

- The Register File contains addresses for the general-purpose registers and the eZ8 CPU, peripheral, and general-purpose I/O port control registers.
- The Program Memory contains addresses for all memory locations having executable code and/or data.
- The Data Memory contains addresses for all memory locations that hold data only.

These three address spaces are covered briefly in the following subsections. For more detailed information regarding the eZ8 CPU and its address space, refer to the *eZ8 CPU User Manual* available for download at www.zilog.com.

## Register File

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

The upper 256 bytes of the 4KB Register File address space are reserved for control of the eZ8 CPU, the on-chip peripherals, and the I/O ports. These registers are located at addresses from F00H to FFFH. Some of the addresses within the 256-byte control register section are reserved (unavailable). Reading from an reserved Register File addresses returns an undefined value. Writing to reserved Register File addresses is not recommended and can produce unpredictable results.

The on-chip RAM always begins at address 000H in the Register File address space. The Z8F640x family products contain 2KB to 4KB of on-chip RAM depending upon the device. Reading from Register File addresses outside the available RAM addresses (and not within in the control register address space) returns an undefined value. Writing to these Register File addresses produces no effect. Refer to the **Part Selection Guide** section of the **Introduction** chapter to determine the amount of RAM available for the specific Z8F640x family device.

## Program Memory

The eZ8 CPU supports 64KB of Program Memory address space. The Z8F640x family devices contain 16KB to 64KB of on-chip Flash memory in the Program Memory address space. Reading from Program Memory addresses outside the available Flash memory addresses returns FFH. Writing to these unemployments Program Memory addresses produces no effect. Table 4 describes the Program Memory Maps for the Z8F640x family products.

**Table 4. Z8F640x Family Program Memory Maps**

| Program Memory Address (Hex) | Function |
|---|---|
| **Z8F160x Products** | |
| 0000-0001 | Flash Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-3FFFH | Program Memory |
| **Z8F240x Products** | |
| 0000-0001 | Flash Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-5FFFH | Program Memory |
| **Z8F320x Products** | |
| 0000-0001 | Flash Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-7FFFH | Program Memory |

*** See Table 22 on page 45 for a list of the interrupt vectors.**

**Table 4. Z8F640x Family Program Memory Maps (Continued)**

| Program Memory Address (Hex) | Function |
|---|---|
| **Z8F480x Products** | |
| 0000-0001 | Flash Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-BFFFH | Program Memory |
| **Z8F640x Products** | |
| 0000-0001 | Flash Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-FFFFH | Program Memory |

*** See Table 22 on page 45 for a list of the interrupt vectors.**

## Data Memory

The Z8F640x family devices contain 128 bytes of read-only memory at the top of the eZ8 CPU's 64KB Data Memory address space. The eZ8 CPU's LDE and LDEI instructions provide access to the Data Memory information. Table 5 describes the Z8F640x family's Data Memory Map.

**Table 5. Z8F640x family Data Memory Maps**

| Data Memory Address (Hex) | Function |
|---|---|
| 0000H-FFBFH | Reserved |
| FFC0H-FFD3H | Part Number<br>20-character ASCII alphanumeric code<br>Left justified and filled with zeros |
| FFD4H-FFFFH | Reserved |

# Register File Address Map

Table 6 provides the address map for the Register File of the Z8F640x family of products. Not all devices and package styles in the Z8F640x family support Timer 3 and all of the GPIO Ports. Consider registers for unimplemented peripherals as Reserved.

**Table 6. Register File Address Map**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page # |
|---|---|---|---|---|
| **General Purpose RAM** | | | | |
| 000-EFF | General-Purpose Register File RAM | — | XX | |
| **Timer 0** | | | | |
| F00 | Timer 0 High Byte | T0H | 00 | 66 |
| F01 | Timer 0 Low Byte | T0L | 01 | 66 |
| F02 | Timer 0 Reload High Byte | T0RH | FF | 67 |
| F03 | Timer 0 Reload Low Byte | T0RL | FF | 67 |
| F04 | Timer 0 PWM High Byte | T0PWMH | 00 | 69 |
| F05 | Timer 0 PWM Low Byte | T0PWML | 00 | 69 |
| F06 | Reserved | — | XX | |
| F07 | Timer 0 Control | T0CTL | 00 | 70 |
| **Timer 1** | | | | |
| F08 | Timer 1 High Byte | T1H | 00 | 66 |
| F09 | Timer 1 Low Byte | T1L | 01 | 66 |
| F0A | Timer 1 Reload High Byte | T1RH | FF | 67 |
| F0B | Timer 1 Reload Low Byte | T1RL | FF | 67 |
| F0C | Timer 1 PWM High Byte | T1PWMH | 00 | 69 |
| F0D | Timer 1 PWM Low Byte | T1PWML | 00 | 69 |
| F0E | Reserved | — | XX | |
| F0F | Timer 1 Control | T1CTL | 00 | 70 |
| **Timer 2** | | | | |
| F10 | Timer 2 High Byte | T2H | 00 | 66 |
| F11 | Timer 2 Low Byte | T2L | 01 | 66 |
| F12 | Timer 2 Reload High Byte | T2RH | FF | 67 |
| F13 | Timer 2 Reload Low Byte | T2RL | FF | 67 |
| F14 | Timer 2 PWM High Byte | T2PWMH | 00 | 69 |
| F15 | Timer 2 PWM Low Byte | T2PWML | 00 | 69 |
| F16 | Reserved | — | XX | |
| F17 | Timer 2 Control | T2CTL | 00 | 70 |
| XX=Undefined | | | | |

**Table 6. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page # |
|---|---|---|---|---|
| **Timer 3 (not available in 40- and 44- Pin Packages)** | | | | |
| F18 | Timer 3 High Byte | T3H | 00 | 66 |
| F19 | Timer 3 Low Byte | T3L | 01 | 66 |
| F1A | Timer 3 Reload High Byte | T3RH | FF | 67 |
| F1B | Timer 3 Reload Low Byte | T3RL | FF | 67 |
| F1C | Timer 3 PWM High Byte | T3PWMH | 00 | 69 |
| F1D | Timer 3 PWM Low Byte | T3PWML | 00 | 69 |
| F1E | Reserved | — | XX | |
| F1F | Timer 3 Control | T3CTL | 00 | 70 |
| F20-F3F | Reserved | — | XX | |
| **UART 0** | | | | |
| F40 | UART0 Transmit Data | U0TXD | XX | 86 |
| | UART0 Receive Data | U0RXD | XX | 87 |
| F41 | UART0 Status 0 | U0STAT0 | 0000011Xb | 87 |
| F42 | UART0 Control 0 | U0CTL0 | 00 | 89 |
| F43 | UART0 Control 1 | U0CTL1 | 00 | 89 |
| F44 | UART0 Status 1 | U0STAT1 | 00 | 87 |
| F45 | Reserved | — | XX | |
| F46 | UART0 Baud Rate High Byte | U0BRH | FF | 91 |
| F47 | UART0 Baud Rate Low Byte | U0BRL | FF | 91 |
| **UART 1** | | | | |
| F48 | UART1 Transmit Data | U1TXD | XX | 86 |
| | UART1 Receive Data | U1RXD | XX | 87 |
| F49 | UART1 Status 0 | U1STAT0 | 0000011Xb | 87 |
| F4A | UART1 Control 0 | U1CTL0 | 00 | 89 |
| F4B | UART1 Control 1 | U1CTL1 | 00 | 89 |
| F4C | UART1 Status 1 | U1STAT1 | 00 | 87 |
| F4D | Reserved | — | XX | |
| F4E | UART1 Baud Rate High Byte | U1BRH | FF | 91 |
| F4F | UART1 Baud Rate Low Byte | U1BRL | FF | 91 |
| **$I^2C$** | | | | |
| F50 | $I^2C$ Data | I2CDATA | 00 | 118 |
| F51 | $I^2C$ Status | I2CSTAT | 80 | 118 |
| F52 | $I^2C$ Control | I2CCTL | 00 | 119 |
| F53 | $I^2C$ Baud Rate High Byte | I2CBRH | FF | 121 |
| F54 | $I^2C$ Baud Rate Low Byte | I2CBRL | FF | 121 |
| F55-F5F | Reserved | — | XX | |
| **Serial Peripheral Interface (SPI)** | | | | |
| F60 | SPI Data | SPIDATA | XX | 106 |
| F61 | SPI Control | SPICTL | 00 | 107 |

XX=Undefined

**Table 6. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page # |
|---|---|---|---|---|
| F62 | SPI Status | SPISTAT | 01 | 108 |
| F63 | SPI Mode | SPIMODE | 00 | 109 |
| F64-F65 | Reserved | — | XX | |
| F66 | SPI Baud Rate High Byte | SPIBRH | FF | 110 |
| F67 | SPI Baud Rate Low Byte | SPIBRL | FF | 110 |
| F68-F69 | Reserved | — | XX | |
| **Analog-to-Digital Converter (ADC)** | | | | |
| F70 | ADC Control | ADCCTL | 20 | 135 |
| F71 | Reserved | — | XX | |
| F72 | ADC Data High Byte | ADCD_H | XX | 137 |
| F73 | ADC Data Low Bits | ADCD_L | XX | 137 |
| F74-FAF | Reserved | — | XX | |
| **DMA 0** | | | | |
| FB0 | DMA0 Control | DMA0CTL | 00 | 124 |
| FB1 | DMA0 I/O Address | DMA0IO | XX | 125 |
| FB2 | DMA0 End/Start Address High Nibble | DMA0H | XX | 126 |
| FB3 | DMA0 Start Address Low Byte | DMA0START | XX | 127 |
| FB4 | DMA0 End Address Low Byte | DMA0END | XX | 128 |
| **DMA 1** | | | | |
| FB8 | DMA1 Control | DMA1CTL | 00 | 124 |
| FB9 | DMA1 I/O Address | DMA1IO | XX | 125 |
| FBA | DMA1 End/Start Address High Nibble | DMA1H | XX | 126 |
| FBB | DMA1 Start Address Low Byte | DMA1START | XX | 127 |
| FBC | DMA1 End Address Low Byte | DMA1END | XX | 128 |
| **DMA ADC** | | | | |
| FBD | DMA_ADC Address | DMAA_ADDR | XX | 128 |
| FBE | DMA_ADC Control | DMAACTL | 00 | 130 |
| FBF | DMA_ADC Status | DMAASTAT | 00 | 131 |
| **Interrupt Controller** | | | | |
| FC0 | Interrupt Request 0 | IRQ0 | 00 | 48 |
| FC1 | IRQ0 Enable High Bit | IRQ0ENH | 00 | 51 |
| FC2 | IRQ0 Enable Low Bit | IRQ0ENL | 00 | 51 |
| FC3 | Interrupt Request 1 | IRQ1 | 00 | 49 |
| FC4 | IRQ1 Enable High Bit | IRQ1ENH | 00 | 52 |
| FC5 | IRQ1 Enable Low Bit | IRQ1ENL | 00 | 52 |
| FC6 | Interrupt Request 2 | IRQ2 | 00 | 50 |
| FC7 | IRQ2 Enable High Bit | IRQ2ENH | 00 | 53 |
| FC8 | IRQ2 Enable Low Bit | IRQ2ENL | 00 | 53 |
| FC9-FCC | Reserved | — | XX | |
| FCD | Interrupt Edge Select | IRQES | 00 | 54 |

XX=Undefined

**Table 6. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page # |
|---|---|---|---|---|
| FCE | Interrupt Port Select | IRQPS | 00 | 55 |
| FCF | Interrupt Control | IRQCTL | 00 | 56 |
| **GPIO Port A** | | | | |
| FD0 | Port A Address | PAADDR | 00 | 37 |
| FD1 | Port A Control | PACTL | 00 | 38 |
| FD2 | Port A Input Data | PAIN | XX | 42 |
| FD3 | Port A Output Data | PAOUT | 00 | 43 |
| **GPIO Port B** | | | | |
| FD4 | Port B Address | PBADDR | 00 | 37 |
| FD5 | Port B Control | PBCTL | 00 | 38 |
| FD6 | Port B Input Data | PBIN | XX | 42 |
| FD7 | Port B Output Data | PBOUT | 00 | 43 |
| **GPIO Port C** | | | | |
| FD8 | Port C Address | PCADDR | 00 | 37 |
| FD9 | Port C Control | PCCTL | 00 | 38 |
| FDA | Port C Input Data | PCIN | XX | 42 |
| FDB | Port C Output Data | PCOUT | 00 | 43 |
| **GPIO Port D** | | | | |
| FDC | Port D Address | PDADDR | 00 | 37 |
| FDD | Port D Control | PDCTL | 00 | 38 |
| FDE | Port D Input Data | PDIN | XX | 42 |
| FDF | Port D Output Data | PDOUT | 00 | 43 |
| **GPIO Port E** | | | | |
| FE0 | Port E Address | PEADDR | 00 | 37 |
| FE1 | Port E Control | PECTL | 00 | 38 |
| FE2 | Port E Input Data | PEIN | XX | 42 |
| FE3 | Port E Output Data | PEOUT | 00 | 43 |
| **GPIO Port F** | | | | |
| FE4 | Port F Address | PFADDR | 00 | 37 |
| FE5 | Port F Control | PFCTL | 00 | 38 |
| FE6 | Port F Input Data | PFIN | XX | 42 |
| FE7 | Port F Output Data | PFOUT | 00 | 43 |
| **GPIO Port G** | | | | |
| FE8 | Port G Address | PGADDR | 00 | 37 |
| FE9 | Port G Control | PGCTL | 00 | 38 |
| FEA | Port G Input Data | PGIN | XX | 42 |
| FEB | Port G Output Data | PGOUT | 00 | 43 |
| **GPIO Port H** | | | | |
| FEC | Port H Address | PHADDR | 00 | 37 |

XX=Undefined

**Table 6. Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page # |
|---|---|---|---|---|
| FED | Port H Control | PHCTL | 00 | 38 |
| FEE | Port H Input Data | PHIN | XX | 42 |
| FEF | Port H Output Data | PHOUT | 00 | 43 |
| **Watch-Dog Timer (WDT)** | | | | |
| FF0 | Watch-Dog Timer Control | WDTCTL | XXX00000b | 75 |
| FF1 | Watch-Dog Timer Reload Upper Byte | WDTU | FF | 76 |
| FF2 | Watch-Dog Timer Reload High Byte | WDTH | FF | 76 |
| FF3 | Watch-Dog Timer Reload Low Byte | WDTL | FF | 76 |
| FF4--FF7 | Reserved | — | XX | |
| **Flash Memory Controller** | | | | |
| FF8 | Flash Control | FCTL | 00 | 144 |
| FF8 | Flash Status | FSTAT | 00 | 145 |
| FF9 | Flash Page Select | FPS | 00 | 146 |
| FFA | Flash Programming Frequency High Byte | FFREQH | 00 | 147 |
| FFB | Flash Programming Frequency Low Byte | FFREQL | 00 | 147 |
| **eZ8 CPU** | | | | |
| FFC | Flags | — | XX | Refer to the *eZ8 CPU User Manual* |
| FFD | Register Pointer | RP | XX | |
| FFE | Stack Pointer High Byte | SPH | XX | |
| FFF | Stack Pointer Low Byte | SPL | XX | |

XX=Undefined

# Reset and Stop Mode Recovery

## Overview

The Reset Controller within the Z8F640x family devices controls Reset and STOP Mode Recovery operation. In typical operation, the following events cause a Reset to occur:

- Power-On Reset (POR)
- Voltage Brown-Out (VBO)
- Watch-Dog Timer time-out (when configured via the WDT_RES Option Bit to initiate a reset)
- External $\overline{\text{RESET}}$ pin assertion
- On-Chip Debugger initiated Reset (OCDCTL[1] set to 1)

When the Z8F640x family device is in Stop mode, a Stop Mode Recovery is initiated by either of the following:

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

## Reset Types

The Z8F640x family provides several different types of Reset operation. Stop Mode Recovery is considered a form of Reset. The type of Reset is a function of both the current operating mode of the Z8F640x family device and the source of the Reset. Table 7 lists the types of Reset and their operating characteristics. The System Reset is longer than the Short Reset to allow additional time for external oscillator start-up.

**Table 7. Reset and Stop Mode Recovery Characteristics and Latency**

| | Reset Characteristics and Latency | | |
|---|---|---|---|
| **Reset Type** | **Control Registers** | **eZ8 CPU** | **Reset Latency (Delay)** |
| System Reset | Reset (as applicable) | Reset | 514 WDT Oscillator cycles + 16 System Clock cycles |
| Short Reset | Reset (as applicable) | Reset | 66 WDT Oscillator cycles + 16 System Clock cycles |
| Stop Mode Recovery | Unaffected, except WDT_CTL register | Reset | 514 WDT Oscillator cycles + 16 System Clock cycles |

### System and Short Resets

During a System Reset, the Z8F640x family device is held in Reset for 514 cycles of the Watch-Dog Timer oscillator followed by 16 cycles of the system clock (crystal oscillator). A Short Reset differs from a System Reset only in the number of Watch-Dog Timer oscillator cycles required to exit Reset. A Short Reset requires only 66 Watch-Dog Timer oscillator cycles. Unless specifically stated otherwise, System Reset and Short Reset are referred to collectively as Reset.

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

Upon 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 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.

## Reset Sources

Table 8 lists the reset sources and type of Reset as a function of the Z8F640x family device operating mode. The text following provides more detailed information on the individual Reset sources. Please note that Power-On Reset / Voltage Brown-Out events always have priority over all other possible reset sources to insure a full system reset occurs.

**Table 8. Reset Sources and Resulting Reset Type**

| Operating Mode | Reset Source | Reset Type |
|---|---|---|
| Normal or Halt modes | Power-On Reset / Voltage Brown-Out | System Reset |
| | Watch-Dog Timer time-out when configured for Reset | Short Reset |
| | $\overline{\text{RESET}}$ pin assertion | Short Reset |
| | On-Chip Debugger initiated Reset (OCDCTL[1] set to 1) | System Reset except the On-Chip Debugger is unaffected by the reset |
| Stop mode | Power-On Reset / Voltage Brown-Out | System Reset |
| | $\overline{\text{RESET}}$ pin assertion | System Reset |
| | DBG pin driven Low | System Reset |

## Power-On Reset

The Z8F640x family products contain an internal Power-On Reset (POR) circuit. The POR circuit monitors the supply voltage and holds the device in the Reset state until the supply voltage reaches a safe operating level. After the supply voltage exceeds the POR voltage threshold ($V_{POR}$), the POR Counter is enabled and counts 514 cycles of the Watch-Dog Timer oscillator. After the POR counter times out, the XTAL Counter is enabled to count a total of 16 system clock pulses. The Z8F640x family device is held in the Reset state until both the POR Counter and XTAL counter have timed out. After the device exits the Power-On Reset state, the eZ8 CPU fetches the Reset vector. Following Power-On Reset, the POR status bit in the Watch-Dog Timer Control (WDTCTL) register is set to 1.

Figure 62 illustrates Power-On Reset operation. Refer to the **Electrical Characteristics** chapter for the POR threshold voltage ($V_{POR}$).

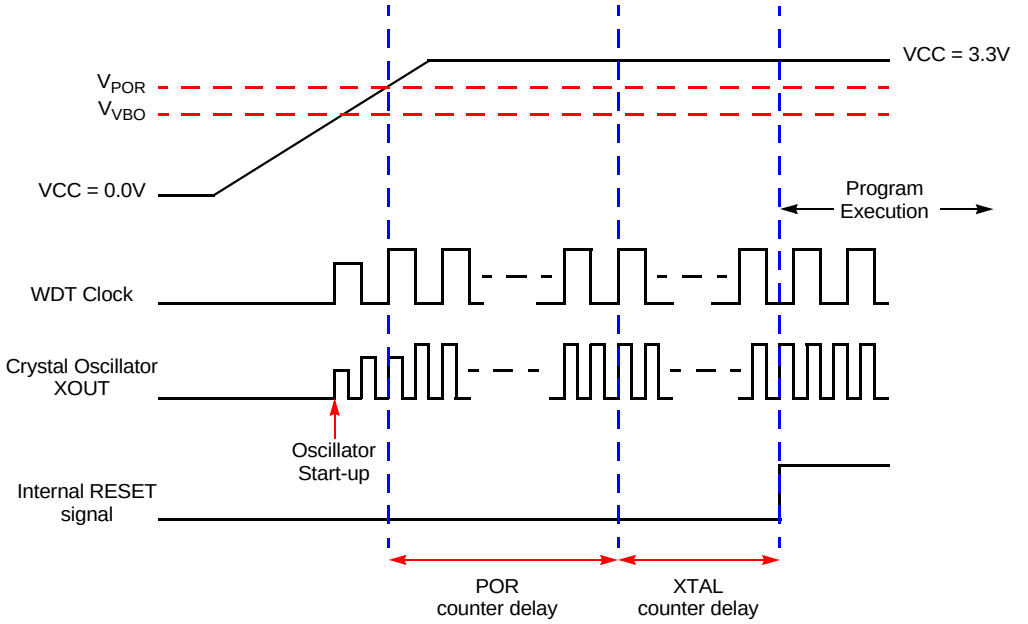



**Figure 62. Power-On Reset Operation (not to scale)**

## Voltage Brown-Out Reset

The devices in the Z8F640x family provide low Voltage Brown-Out (VBO) protection. The VBO circuit senses when the supply voltage drops to an unsafe level (below the VBO

threshold voltage, $V_{VBO}$) and forces the device into the Reset state. While the supply voltage remains below the Power-On Reset voltage threshold ($V_{POR}$), the VBO block holds the Z8F640x family device in the Reset state.

After the supply voltage again exceeds the Power-On Reset voltage threshold, the Z8F640x family device progresses through a full System Reset sequence, as described in the Power-On Reset section. Following Power-On Reset, the POR status bit in the Watch-Dog Timer Control (WDTCTL) register is set to 1. Figure 63 illustrates Voltage Brown-Out operation. Refer to the **Electrical Characteristics** chapter for the VBO and POR threshold voltages ($V_{VBO}$ and $V_{POR}$).

Stop mode disables the Voltage Brown-Out detector.

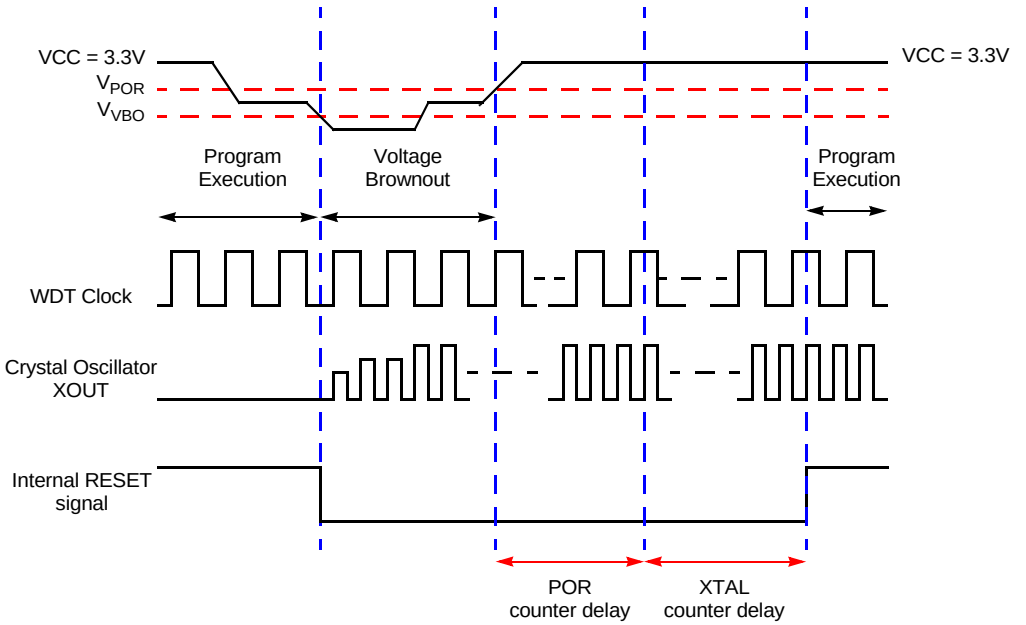



**Figure 63. Voltage Brown-Out Reset Operation (not to scale)**

## Watch-Dog Timer Reset

If the device is in normal or Halt mode, the Watch-Dog Timer can initiate a System Reset at time-out if the WDT_RES Option Bit is set to 1. This is the default (unprogrammed) setting of the WDT_RES Option Bit. The WDT status bit in the WDT Control register is set to signify that the reset was initiated by the Watch-Dog Timer.

### External Pin Reset

The $\overline{\text{RESET}}$ pin has a Schmitt-triggered input and an internal pull-up. Once the $\overline{\text{RESET}}$ pin is asserted, the device progresses through the Short Reset sequence. While the $\overline{\text{RESET}}$ input pin is asserted Low, the Z8F640x family device continues to be held in the Reset state. If the $\overline{\text{RESET}}$ pin is held Low beyond the Short Reset time-out, the device exits the Reset state immediately following $\overline{\text{RESET}}$ pin deassertion. Following a Short Reset initiated by the external $\overline{\text{RESET}}$ pin, the EXT status bit in the Watch-Dog Timer Control (WDTCTL) register is set to 1.

## Stop Mode Recovery

Stop mode is entered by execution of a STOP instruction by the eZ8 CPU. Refer to the **Low-Power Modes** chapter for detailed Stop mode information. During Stop Mode Recovery, the Z8F640x family device is held in reset for 514 cycles of the Watch-Dog Timer oscillator followed by 16 cycles of the system clock (crystal oscillator). Stop Mode Recovery does not affect any values in the Register File, including the Stack Pointer, Register Pointer, Flags and general-purpose RAM.

The eZ8 CPU fetches the Reset vector at Program Memory addresses 0002H and 0003H and loads that value into the Program Counter. Program execution begins at the Reset vector address. Following Stop Mode Recovery, the STOP bit in the Watch-Dog Timer Control Register is set to 1. Table 9 lists the Stop Mode Recovery sources and resulting actions. The text following provides more detailed information on each of the Stop Mode Recovery sources.

**Table 9. Stop Mode Recovery Sources and Resulting Action**

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

### Stop Mode Recovery Using Watch-Dog Timer Time-Out

If the Watch-Dog Timer times out during Stop mode, the Z8F640x family device undergoes a STOP Mode Recovery sequence. In the Watch-Dog Timer Control register, the WDT and STOP bits are set to 1. If the Watch-Dog Timer is configured to generate an interrupt upon time-out and the device is configured to respond to interrupts, the Z8F640x family device services the Watch-Dog Timer interrupt request following the normal Stop Mode Recovery sequence.

### Stop Mode Recovery Using a GPIO Port Pin Transition

Each of the GPIO Port pins may be configured as a Stop Mode Recovery input source. On any GPIO pin enabled as a Stop Mode Recover source, a change in the input pin value (from High to Low or from Low to High) initiates Stop Mode Recovery. In the Watch-Dog Timer Control register, the STOP bit is set to 1.

**Caution:** In Stop mode, the GPIO Port Input Data registers (P*x*IN) are disabled. The Port Input Data registers record the Port transition only if the signal stays on the Port pin through the end of the STOP Mode Recovery delay. Thus, 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).

# Low-Power Modes

## Overview

The Z8F640x family products contain power-saving features. The highest level of power reduction is provided by Stop mode. The next level of power reduction is provided by the Halt mode.

## Stop Mode

Execution of the eZ8 CPU's STOP instruction places the Z8F640x family device into Stop mode. In Stop mode, the operating characteristics are:

- Primary crystal oscillator is stopped
- System clock is stopped
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- Watch-Dog Timer's internal RC oscillator continues to operate
- If enabled, the Watch-Dog Timer continues to operate
- All other on-chip peripherals are idle

To minimize current in Stop mode, all GPIO pins that are configured as digital inputs must be driven to one of the supply rails ($V_{CC}$ or GND). The Z8F640x family device can be brought out of Stop mode using Stop Mode Recovery. For more information on STOP Mode Recovery refer to the **Reset and Stop Mode Recovery** chapter.

## Halt Mode

Execution of the eZ8 CPU's HALT instruction places the Z8F640x family device into Halt mode. In Halt mode, the operating characteristics are:

- Primary crystal oscillator is enabled and continues to operate
- System clock is enabled and continues to operate
- eZ8 CPU is idle
- Program counter (PC) stops incrementing

- Watch-Dog Timer's internal RC oscillator continues to operate
- If enabled, the Watch-Dog Timer 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
- Watch-Dog 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 ($V_{CC}$ or GND).

# General-Purpose I/O

## Overview

The Z8F640x family products support a maximum of seven 8-bit ports (Ports A-G) and one 4-bit port (Port H) for general-purpose input/output (I/O) operations. Each port contains control and data registers. The GPIO control registers are used to determine data direction, open-drain, output drive current and alternate pin functions. Each port pin is individually programmable.

## GPIO Port Availability By Device

Not all Z8F640x family products support all 8 ports (A-H). Table 10 lists the port pins available with each device and package type.

**Table 10. Port Availability by Device and Package Type**

| Device | Packages | Port A | Port B | Port C | Port D | Port E | Port F | Port G | Port H |
|---|---|---|---|---|---|---|---|---|---|
| Z8F1601 | 40-pin | [7:0] | [7:0] | [6:0] | [6:3, 1:0] | - | - | - | - |
| Z8F1601 | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | | | | |
| Z8F1602 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8F2401 | 40-pin | [7:0] | [7:0] | [6:0] | [6:3, 1:0] | - | - | - | - |
| Z8F2401 | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | - | - | - | - |
| Z8F2402 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8F3201 | 40-pin | [7:0] | [7:0] | [6:0] | [6:3, 1:0] | - | - | - | - |
| Z8F3201 | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | - | - | - | - |
| Z8F3202 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8F4801 | 40-pin | [7:0] | [7:0] | [6:0] | [6:3, 1:0] | - | - | - | - |
| Z8F4801 | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | - | - | - | - |
| Z8F4802 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8F4803 | 80-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [3:0] |
| Z8F6401 | 40-pin | [7:0] | [7:0] | [6:0] | [6:3, 1:0] | - | - | - | - |

**Table 10. Port Availability by Device and Package Type (Continued)**

| Device | Packages | Port A | Port B | Port C | Port D | Port E | Port F | Port G | Port H |
|---|---|---|---|---|---|---|---|---|---|
| Z8F6401 | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | - | - | - | - |
| Z8F6402 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8F6403 | 80-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [3:0] |

## Architecture

Figure 64 illustrates a simplified block diagram of a GPIO port pin. In this figure, the ability to accommodate alternate functions and variable port current drive strength are not illustrated.

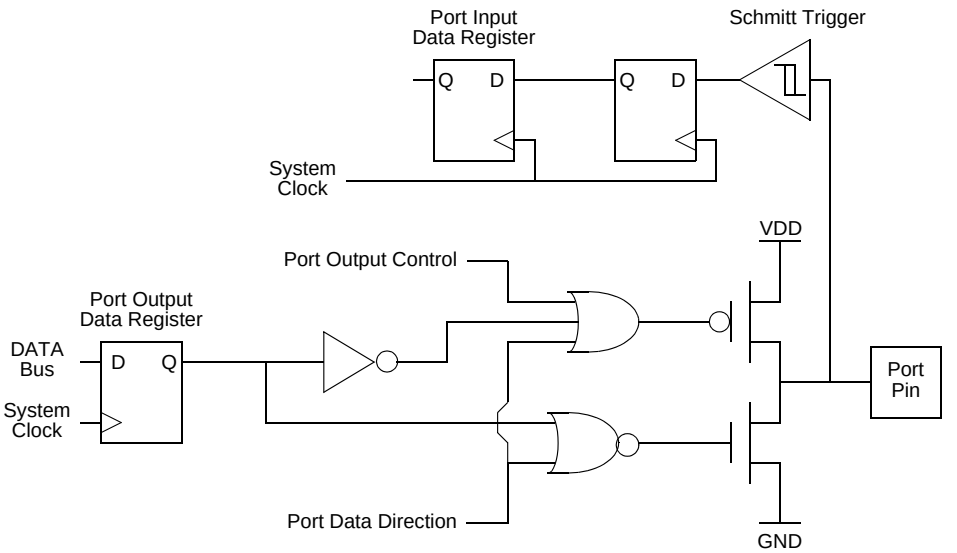



**Figure 64. GPIO Port Pin Block Diagram**

## GPIO Alternate Functions

Many of the GPIO port pins can be used as both general-purpose I/O and to provide access to on-chip peripheral functions such as the timers and serial communication devices. The Port A-H Alternate Function sub-registers configure these pins for either general-purpose I/O or alternate function operation. When a pin is configured for alternate function, control

of the port pin direction (input/output) is passed from the Port A-H Data Direction registers to the alternate function assigned to this pin. Table 11 lists the alternate functions associated with each port pin.

**Table 11. Port Alternate Function Mapping**

| Port | Pin | Mnemonic | Alternate Function Description |
|---|---|---|---|
| **Port A** | PA0 | T0IN | Timer 0 Input |
| | PA1 | T0OUT | Timer 0 Output |
| | PA2 | N/A | No alternate function |
| | PA3 | $\overline{\text{CTS0}}$ | UART 0 Clear to Send |
| | PA4 | RXD0 / IRRX0 | UART 0 / IrDA 0 Receive Data |
| | PA5 | TXD0 / IRTX0 | UART 0 / IrDA 0 Transmit Data |
| | PA6 | SCL | $I^2C$ Clock (automatically open-drain) |
| | PA7 | SDA | $I^2C$ Data (automatically open-drain) |
| **Port B** | PB0 | ANA0 | ADC Analog Input 0 |
| | PB1 | ANA1 | ADC Analog Input 1 |
| | PB2 | ANA2 | ADC Analog Input 2 |
| | PB3 | ANA3 | ADC Analog Input 3 |
| | PB4 | ANA4 | ADC Analog Input 4 |
| | PB5 | ANA5 | ADC Analog Input 5 |
| | PB6 | ANA6 | ADC Analog Input 6 |
| | PB7 | ANA7 | ADC Analog Input 7 |
| **Port C** | PC0 | T1IN | Timer 1 Input |
| | PC1 | T1OUT | Timer 1 Output |
| | PC2 | $\overline{\text{SS}}$ | SPI Slave Select |
| | PC3 | SCK | SPI Serial Clock |
| | PC4 | MOSI | SPI Master Out Slave In |
| | PC5 | MISO | SPI Master In Slave Out |
| | PC6 | T2IN | Timer 2 In |
| | PC7 | T2OUT | Timer 2 Out (not available in 40-pin packages) |

**Table 11. Port Alternate Function Mapping (Continued)**

| Port | Pin | Mnemonic | Alternate Function Description |
|---|---|---|---|
| **Port D** | PD0 | T3IN | Timer 3 In (not available in 40- and 44-pin packages) |
| | PD1 | T3OUT | Timer 3 Out (not available in 40- and 44-pin packages) |
| | PD2 | N/A | No alternate function |
| | PD3 | N/A | No alternate function |
| | PD4 | RXD1 / IRRX1 | UART 1 / IrDA 1 Receive Data |
| | PD5 | TXD1 / IRTX1 | UART 1 / IrDA 1 Transmit Data |
| | PD6 | $\overline{\text{CTS1}}$ | UART 1 Clear to Send |
| | PD7 | RCOUT | Watch-Dog Timer RC Oscillator Output |
| **Port E** | PE[7:0] | N/A | No alternate functions |
| **Port F** | PF[7:0] | N/A | No alternate functions |
| **Port G** | PG[7:0] | N/A | No alternate functions |
| **Port H** | PH0 | ANA8 | ADC Analog Input 8 |
| | PH1 | ANA9 | ADC Analog Input 9 |
| | PH2 | ANA10 | ADC Analog Input 10 |
| | PH3 | ANA11 | ADC Analog Input 11 |

## GPIO Interrupts

Many of the GPIO port pins can be used as interrupt sources. Some port pins may be configured to generate an interrupt request on either the rising edge or falling edge of the pin input signal. Other port pin interrupts generate an interrupt when any edge occurs (both rising and falling). Refer to the **Interrupt Controller** chapter for more information on interrupts using the GPIO pins.

## GPIO Control Register Definitions

Four registers for each Port provide access to GPIO control, input data, and output data. Table 12 lists these Port registers. Use the Port A-H Address and Control registers together to provide access to sub-registers for Port configuration and control.

**Table 12. GPIO Port Registers and Sub-Registers**

| Port Register Mnemonic | Port Register Name |
|---|---|
| P*x*ADDR | Port A-H Address Register (Selects sub-registers) |
| P*x*CTL | Port A-H Control Register (Provides access to sub-registers) |
| P*x*IN | Port A-H Input Data Register |
| P*x*OUT | Port A-H Output Data Register |
| **Port Sub-Register 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 |

## Port A-H Address Registers

The Port A-H Address registers select the GPIO Port functionality accessible through the Port A-H Control registers. The Port A-H Address and Control registers combine to provide access to all GPIO Port control (Table 13).

**Table 13. Port A-H GPIO Address Registers (P*x*ADDR)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PADDR[7:0] | | | | | | | |
| RESET | 00H | | | | | | | |
| R/W | R/W | | | | | | | |
| ADDR | FD0H, FD4H, FD8H, FDCH, FE0H, FE4H, FE8H, FECH | | | | | | | |

PADDR[7:0]—Port Address
The Port Address selects one of the sub-registers accessible through the Port Control register.

| PADDR[7:0] | Port Control sub-register accessible using the Port A-H 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-FFH | No function. |

## Port A-H Control Registers

The Port A-H Control registers set the GPIO port operation. The value in the corresponding Port A-H Address register determines the control sub-registers accessible using the Port A-H Control register (Table 14).

**Table 14. Port A-H Control Registers (P*x*CTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PCTL | | | | | | | |
| RESET | 00H | | | | | | | |
| R/W | R/W | | | | | | | |
| ADDR | FD1H, FD5H, FD9H, FDDH, FE1H, FE5H, FE9H, FEDH | | | | | | | |

PCTL[7:0]—Port Control
The Port Control register provides access to all sub-registers that configure the GPIO Port operation.

### Port A-H Data Direction Sub-Registers

The Port A-H Data Direction sub-register is accessed through the Port A-H Control register by writing 01H to the Port A-H Address register (Table 15).

**Table 15. Port A-H Data Direction Sub-Registers**

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

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

### Port A-H Alternate Function Sub-Registers

The Port A-H Alternate Function sub-register (Table 16) is accessed through the Port A-H Control register by writing 02H to the Port A-H Address register. The Port A-H Alternate Function sub-registers select the alternate functions for the selected pins. Refer to the **GPIO Alternate Functions** section to determine the alternate function associated with each port pin.

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

**Table 16. Port A-H Alternate Function Sub-Registers**

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

AF[7:0]—Port Alternate Function enabled
0 = The port pin is in normal mode and the DDx bit in the Port A-H Data Direction sub-register determines the direction of the pin.
1 = The alternate function is selected. Port pin operation is controlled by the alternate function.

### Port A-H Output Control Sub-Registers

The Port A-H Output Control sub-register (Table 17) is accessed through the Port A-H Control register by writing 03H to the Port A-H Address register. Setting the bits in the Port A-H Output Control sub-registers to 1 configures the specified port pins for open-drain operation. These sub-registers affect the pins directly and, as a result, alternate functions are also affected.

**Table 17. Port A-H Output Control Sub-Registers**

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

POC[7:0]—Port Output Control
These bits function independently of the alternate function bit and disables the drains if set to 1.
0 = The drains are enabled for any output mode.
1 = The drain of the associated pin is disabled (open-drain mode).

### Port A-H High Drive Enable Sub-Registers

The Port A-H High Drive Enable sub-register (Table 18) is accessed through the Port A-H Control register by writing 04H to the Port A-H Address register. Setting the bits in the Port A-H High Drive Enable sub-registers to 1 configures the specified port pins for high current output drive operation. The Port A-H High Drive Enable sub-register affects the pins directly and, as a result, alternate functions are also affected.

**Table 18. Port A-H High Drive Enable Sub-Registers**

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

PHDE[7:0]—Port High Drive Enabled
0 = The Port pin is configured for standard output current drive.
1 = The Port pin is configured for high output current drive.

### Port A-H Stop Mode Recovery Source Enable Sub-Registers

The Port A-H STOP Mode Recovery Source Enable sub-register (Table 19) is accessed through the Port A-H Control register by writing 05H to the Port A-H Address register. Setting the bits in the Port A-H STOP Mode Recovery Source Enable sub-registers to 1 configures the specified Port pins as a STOP Mode Recovery source. During STOP Mode, any logic transition on a Port pin enabled as a STOP Mode Recovery source initiates STOP Mode Recovery.

**Table 19. Port A-H STOP Mode Recovery Source Enable Sub-Registers**

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

PSMRE[7:0]—Port STOP Mode Recovery Source Enabled
0 = The Port pin is not configured as a STOP Mode Recovery source. Transitions on this pin during Stop mode do not initiate STOP Mode Recovery.

1 = The Port pin is configured as a STOP Mode Recovery source. Any logic transition on this pin during Stop mode initiates STOP Mode Recovery.

## Port A-H Input Data Registers

Reading from the Port A-H Input Data registers (Table 20) returns the sampled values from the corresponding port pins. The Port A-H Input Data registers are Read-only.

**Table 20. Port A-H Input Data Registers (PxIN)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | X | X | X | X | X | X | X |
| R/W | R | R | R | R | R | R | R | R |
| ADDR | FD2H, FD6H, FDAH, FDEH, FE2H, FE6H, FEAH, FEEH | | | | | | | |

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

### Port A-H Output Data Register

The Port A-H Output Data register (Table 21) writes output data to the pins.

**Table 21. Port A-H 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 |
| ADDR | FD3H, FD7H, FDBH, FDFH, FE3H, FE7H, FEBH, FEFH | | | | | | | |

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

# Interrupt Controller

## Overview

The interrupt controller on the Z8F640x family device prioritizes the interrupt requests from the on-chip peripherals and the GPIO port pins. The features of the interrupt controller on the Z8F640x family device include the following:

- 24 unique interrupt vectors:
  - 12 GPIO port pin interrupt sources
  - 12 on-chip peripheral interrupt sources
- Flexible GPIO interrupts
  - 8 selectable rising and falling edge GPIO interrupts
  - 4 dual-edge interrupts
- 3 levels of individually programmable interrupt priority
- Watch-Dog Timer 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 control has no effect on operation. Refer to the *eZ8 CPU User Manual* for more information regarding interrupt servicing by the eZ8 CPU. The *eZ8 CPU User Manual* is available for download at www.zilog.com.

## Interrupt Vector Listing

Table 22 lists all of the interrupts available on the Z8F640x family device in order of priority. The interrupt vector is stored with the most significant byte (MSB) at the even Program Memory address and the least significant byte (LSB) at the following odd Program Memory address.

**Table 22. Interrupt Vectors in Order of Priority**

| Priority | Program Memory Vector Address | Interrupt Source | Interrupt Assertion Type |
|---|---|---|---|
| Highest | 0002h | Reset (not an interrupt) | Not applicable |
| | 0004h | Watch-Dog Timer | Continuous assertion |
| | 0006h | Illegal Instruction Trap (not an interrupt) | Not applicable |
| | 0008h | Timer 2 | Single assertion (pulse) |
| | 000Ah | Timer 1 | Single assertion (pulse) |
| | 000Ch | Timer 0 | Single assertion (pulse) |
| | 000Eh | UART 0 receiver | Continuous assertion |
| | 0010h | UART 0 transmitter | Continuous assertion |
| | 0012h | $I^2C$ | Continuous assertion |
| | 0014h | SPI | Continuous assertion |
| | 0016h | ADC | Single assertion (pulse) |
| | 0018h | Port A7 or Port D7, rising or falling input edge | Single assertion (pulse) |
| | 001Ah | Port A6 or Port D6, rising or falling input edge | Single assertion (pulse) |
| | 001Ch | Port A5 or Port D5, rising or falling input edge | Single assertion (pulse) |
| | 001Eh | Port A4 or Port D4, rising or falling input edge | Single assertion (pulse) |
| | 0020h | Port A3 or Port D3, rising or falling input edge | Single assertion (pulse) |
| | 0022h | Port A2 or Port D2, rising or falling input edge | Single assertion (pulse) |
| | 0024h | Port A1 or Port D1, rising or falling input edge | Single assertion (pulse) |
| | 0026h | Port A0 or Port D0, rising or falling input edge | Single assertion (pulse) |
| | 0028h | Timer 3 *(not available in 40/44-pin packages)* | Single assertion (pulse) |
| | 002Ah | UART 1 receiver | Continuous assertion |
| | 002Ch | UART 1 transmitter | Continuous assertion |
| | 002Eh | DMA | Single assertion (pulse) |
| | 0030h | Port C3, both input edges | Single assertion (pulse) |
| | 0032h | Port C2, both input edges | Single assertion (pulse) |
| | 0034h | Port C1, both input edges | Single assertion (pulse) |
| Lowest | 0036h | Port C0, both input edges | Single assertion (pulse) |

## Architecture

Figure 65 illustrates a block diagram of the interrupt controller.

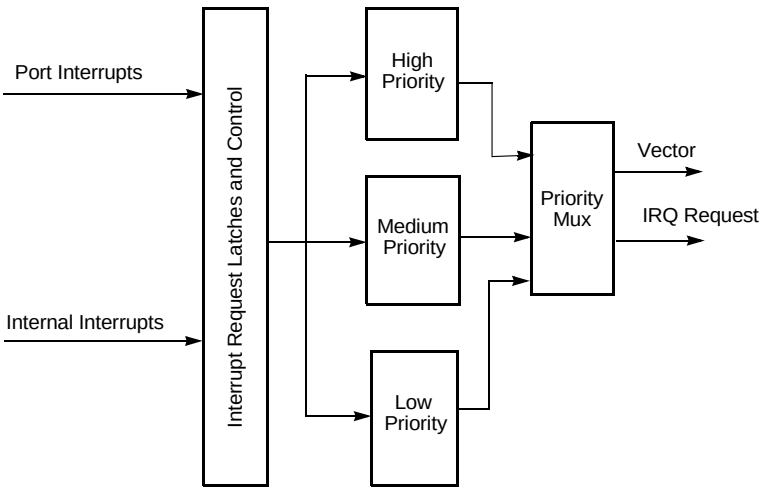



**Figure 65. Interrupt Controller Block Diagram**

## Operation

### Master Interrupt Enable

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

Interrupts are globally enabled by any of the following actions:

- Execution of an EI (Enable Interrupt) instruction
- Execution of an IRET (Return from Interrupt) instruction
- Writing a 1 to the IRQE bit in the Interrupt Control register

Interrupts are globally disabled by any of the following actions:

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

- Execution of a Trap instruction
- Illegal instruction trap

## Interrupt Vectors and Priority

The Z8F640x family device interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority, and Level 1 is the lowest priority. If all of the interrupts were enabled with identical interrupt priority (all as Level 2 interrupts, for example), then interrupt priority would be assigned from highest to lowest as specified in Table 22. Level 3 interrupts always have higher priority than Level 2 interrupts which, in turn, always have higher priority than Level 1 interrupts. Within each interrupt priority level (Level 1, Level 2, or Level 3), priority is assigned as specified in Table 22.

Reset, Watch-Dog Timer interrupt (if enabled), and Illegal Instruction Trap always have highest (Level 3) priority.

## Interrupt Assertion Types

Two types of interrupt assertion - single assertion (pulse) and continuous assertion - are used within the Z8F640x family device. The type of interrupt assertion for each interrupt source is listed in Table 22.

### Single Assertion (Pulse) Interrupt Sources

Some 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.

### Continuous Assertion Interrupt Sources

Other interrupt sources continuously assert their interrupt requests until cleared at the source. For these continuous assertion interrupt sources, interrupt acknowledgement by the eZ8 CPU does not clear the corresponding bit in the Interrupt Request register. Writing a 0 to the corresponding bit in the Interrupt Request register only clears the interrupt for a single clock cycle. Since the source is continuously asserting the interrupt request, the interrupt request bit is set to 1 again during the next clock cycle.

The only way to clear continuous assertion interrupts is at the source of the interrupt (for example, in the UART or SPI peripherals). The source of the interrupt must be cleared first. After the interrupt is cleared at the source, the corresponding bit in the Interrupt Request register must also be cleared to 0. Both the interrupt source and the IRQ register must be cleared.

## Interrupt Control Register Definitions

For all interrupts other than the Watch-Dog Timer interrupt, the interrupt control registers enable individual interrupts, set interrupt priorities, and indicate interrupt requests.

### Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) register (Table 23) 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 23. 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 |
| ADDR | FC0H | | | | | | | |

T2I—Timer 2 Interrupt Request
0 = No interrupt request is pending for Timer 2.
1 = An interrupt request from Timer 2 is awaiting service.

T1I—Timer 1 Interrupt Request
0 = No interrupt request is pending for Timer 1.
1 = An interrupt request from Timer 1 is awaiting service.

T0I—Timer 0 Interrupt Request
0 = No interrupt request is pending for Timer 0.
1 = An interrupt request from Timer 0 is awaiting service.

U0RXI—UART 0 Receiver Interrupt Request
0 = No interrupt request is pending for the UART 0 receiver.
1 = An interrupt request from the UART 0 receiver is awaiting service.

U0TXI—UART 0 Transmitter Interrupt Request
0 = No interrupt request is pending for the UART 0 transmitter.
1 = An interrupt request from the UART 0 transmitter is awaiting service.

$I^2CI$— $I^2C$ Interrupt Request
0 = No interrupt request is pending for the $I^2C$.
1 = An interrupt request from the $I^2C$ is awaiting service.

SPII—SPI Interrupt Request
0 = No interrupt request is pending for the SPI.
1 = An interrupt request from the SPI is awaiting service.

ADCI—ADC Interrupt Request
0 = No interrupt request is pending for the Analog-to-Digital Converter.
1 = An interrupt request from the Analog-to-Digital Converter is awaiting service.

## Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) register (Table 24) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ1 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 1 register to determine if any interrupt requests are pending.

**Table 24. Interrupt Request 1 Register (IRQ1)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PAD7I | PAD6I | PAD5I | PAD4I | PAD3I | PAD2I | PAD1I | PAD0I |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FC3H | | | | | | | |

PAD*x*I—Port A or Port D Pin *x* Interrupt Request
0 = No interrupt request is pending for GPIO Port A or Port D pin *x*.
1 = An interrupt request from GPIO Port A or Port D pin *x* is awaiting service.

where *x* indicates the specific GPIO Port pin number (0 through 7). For each pin, only 1 of either Port A or Port D can be enabled for interrupts at any one time. Port selection (A or D) is determined by the values in the Interrupt Port Select Register.

## Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) register (Table 25) 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 1 register to determine if any interrupt requests are pending.

**Table 25. Interrupt Request 2 Register (IRQ2)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | T3I | U1RXI | U1TXI | DMAI | PC3I | PC2I | PC1I | PC0I |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FC6H | | | | | | | |

T3I—Timer 3 Interrupt Request
0 = No interrupt request is pending for Timer 3.
1 = An interrupt request from Timer 3 is awaiting service.

U1RXI—UART 1 Receive Interrupt Request
0 = No interrupt request is pending for the UART1 receiver.
1 = An interrupt request from UART1 receiver is awaiting service.

U1TXI—UART 1 Transmit Interrupt Request
0 = No interrupt request is pending for the UART 1 transmitter.
1 = An interrupt request from the UART 1 transmitter is awaiting service.

DMAI—DMA Interrupt Request
0 = No interrupt request is pending for the DMA.
1 = An interrupt request from the DMA is awaiting service.

PC*x*I—Port C Pin *x* Interrupt Request
0 = No interrupt request is pending for GPIO Port C pin *x*.
1 = An interrupt request from GPIO Port C pin *x* is awaiting service.

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

## IRQ0 Enable High and Low Bit Registers

The IRQ0 Enable High and Low Bit registers (Tables 27 and 28) form a priority encoded enabling for interrupts in the Interrupt Request 0 register. Priority is generated by setting bits in each register. Table 26 describes the priority control for IRQ0.

**Table 26. IRQ0 Enable and Priority Encoding**

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

where *x* indicates the register bits from 0 through 7.

**Table 27. 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 |
| ADDR | FC1H | | | | | | | |

T2ENH—Timer 2 Interrupt Request Enable High Bit
T1ENH—Timer 1 Interrupt Request Enable High Bit
T0ENH—Timer 0 Interrupt Request Enable High Bit
U0RENH—UART 0 Receive Interrupt Request Enable High Bit
U0TENH—UART 0 Transmit Interrupt Request Enable High Bit
I2CENH—$I^2C$ Interrupt Request Enable High Bit
SPIENH—SPI Interrupt Request Enable High Bit
ADCENH—ADC Interrupt Request Enable High Bit

**Table 28. 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 |
| ADDR | FC2H | | | | | | | |

T2ENL—Timer 2 Interrupt Request Enable Low Bit
T1ENL—Timer 1 Interrupt Request Enable Low Bit
T0ENL—Timer 0 Interrupt Request Enable Low Bit
U0RENL—UART 0 Receive Interrupt Request Enable Low Bit
U0TENL—UART 0 Transmit Interrupt Request Enable Low Bit
I2CENL—$I^2C$ Interrupt Request Enable Low Bit
SPIENL—SPI Interrupt Request Enable Low Bit
ADCENL—ADC Interrupt Request Enable Low Bit

## IRQ1 Enable High and Low Bit Registers

The IRQ1 Enable High and Low Bit registers (Tables 30 and 31) form a priority encoded enabling for interrupts in the Interrupt Request 1 register. Priority is generated by setting bits in each register. Table 29 describes the priority control for IRQ1.

**Table 29. IRQ1 Enable and Priority Encoding**

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

where *x* indicates the register bits from 0 through 7.

**Table 30. IRQ1 Enable High Bit Register (IRQ1ENH)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PAD7ENH | PAD6ENH | PAD5ENH | PAD4ENH | PAD3ENH | PAD2ENH | PAD1ENH | PAD0ENH |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FC4H | | | | | | | |

PAD*x*ENH—Port A or Port D Bit[*x*] Interrupt Request Enable High Bit
Refer to the Interrupt Port Select register for selection of either Port A or Port D as the interrupt source.

**Table 31. IRQ1 Enable Low Bit Register (IRQ1ENL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PAD7ENL | PAD6ENL | PAD5ENL | PAD4ENL | PAD3ENL | PAD2ENL | PAD1ENL | PAD0ENL |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FC5H | | | | | | | |

PAD*x*ENL—Port A or Port D Bit[*x*] Interrupt Request Enable Low Bit
Refer to the Interrupt Port Select register for selection of either Port A or Port D as the interrupt source.

## IRQ2 Enable High and Low Bit Registers

The IRQ2 Enable High and Low Bit registers (Tables 33 and 34) form a priority encoded enabling for interrupts in the Interrupt Request 2 register. Priority is generated by setting bits in each register. Table 32 describes the priority control for IRQ2.

**Table 32. IRQ2 Enable and Priority Encoding**

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

where *x* indicates the register bits from 0 through 7.

**Table 33. IRQ2 Enable High Bit Register (IRQ2ENH)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | T3ENH | U1RENH | U1TENH | DMAENH | C3ENH | C2ENH | C1ENH | C0ENH |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FC7H | | | | | | | |

T3ENH—Timer 3 Interrupt Request Enable High Bit
U1RENH—UART 1 Receive Interrupt Request Enable High Bit
U1TENH—UART 1 Transmit Interrupt Request Enable High Bit
DMAENH—DMA Interrupt Request Enable High Bit
C3ENH—Port C3 Interrupt Request Enable High Bit
C2ENH—Port C2 Interrupt Request Enable High Bit
C1ENH—Port C1 Interrupt Request Enable High Bit
C0ENH—Port C0 Interrupt Request Enable High Bit

**Table 34. IRQ2 Enable Low Bit Register (IRQ2ENL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | T3ENL | U1RENL | U1TENL | DMAENL | C3ENL | C2ENL | C1ENL | C0ENL |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FC8H | | | | | | | |

T3ENL—Timer 3 Interrupt Request Enable Low Bit
U1RENL—UART 1 Receive Interrupt Request Enable Low Bit
U1TENL—UART 1 Transmit Interrupt Request Enable Low Bit
DMAENL—DMA Interrupt Request Enable Low Bit
C3ENL—Port C3 Interrupt Request Enable Low Bit
C2ENL—Port C2 Interrupt Request Enable Low Bit
C1ENL—Port C1 Interrupt Request Enable Low Bit
C0ENL—Port C0 Interrupt Request Enable Low Bit

## Interrupt Edge Select Register

The Interrupt Edge Select (IRQES) register (Table 35) determines whether an interrupt is generated for the rising edge or falling edge on the selected GPIO Port input pin. The

Interrupt Port Select register selects between Port A and Port D for the individual interrupts.

**Table 35. Interrupt Edge Select Register (IRQES)**

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

IES*x*—Interrupt Edge Select *x*
where *x* indicates the specific GPIO Port pin number (0 through 7). The pulse width should be greater than 1 system clock to guarantee capture of the edge triggered interrupt.
0 = An interrupt request is generated on the falling edge of the PA*x*/PD*x* input.
1 = An interrupt request is generated on the rising edge of the PA*x*/PD*x* input.

## Interrupt Port Select Register

The Port Select (IRQPS) register (Table 36) determines the port pin that generates the PAx/PDx interrupts. This register allows either Port A or Port D pins to be used as interrupts. The Interrupt Edge Select register controls the active interrupt edge.

**Table 36. Interrupt Port Select Register (IRQPS)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PAD7S | PAD6S | PAD5S | PAD4S | PAD3S | PAD2S | PAD1S | PAD0S |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FCEH | | | | | | | |

PAD*x*S—PA*x*/PD*x* Selection
0 = PA*x* is used for the interrupt for PA*x*/PD*x* interrupt request.
1 = PD*x* is used for the interrupt for PA*x*/PD*x* interrupt request.
where *x* indicates the specific GPIO Port pin number (0 through 7).

## Interrupt Control Register

The Interrupt Control (IRQCTL) register (Table 37) contains the master enable bit for all interrupts.

**Table 37. Interrupt Control Register (IRQCTL)**

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

IRQE—Interrupt Request Enable
This bit is set to 1 by execution of an EI (Enable Interrupts) or IRET (Interrupt Return) instruction, or by a direct register write of a 1 to this bit. It is reset to 0 by executing a DI instruction, eZ8 CPU acknowledgement of an interrupt request, or Reset.
0 = Interrupts are disabled.
1 = Interrupts are enabled.

Reserved
These bits must be 0.

# Timers

## Overview

The Z8F640x family products contain three to four 16-bit reloadable timers that can be used for timing, event counting, or generation of pulse-width modulated (PWM) signals. The timers' features include:

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

In addition to the timers described in this chapter, the Baud Rate Generators for any unused UART, SPI, or $I^2C$ peripherals may also be used to provide basic timing functionality. Refer to the respective serial communication peripheral chapters for information on using the Baud Rate Generators as timers. Timer 3 is unavailable in the 40- and 44-pin packages.

## Architecture

Figure 66 illustrates the architecture of the timers.

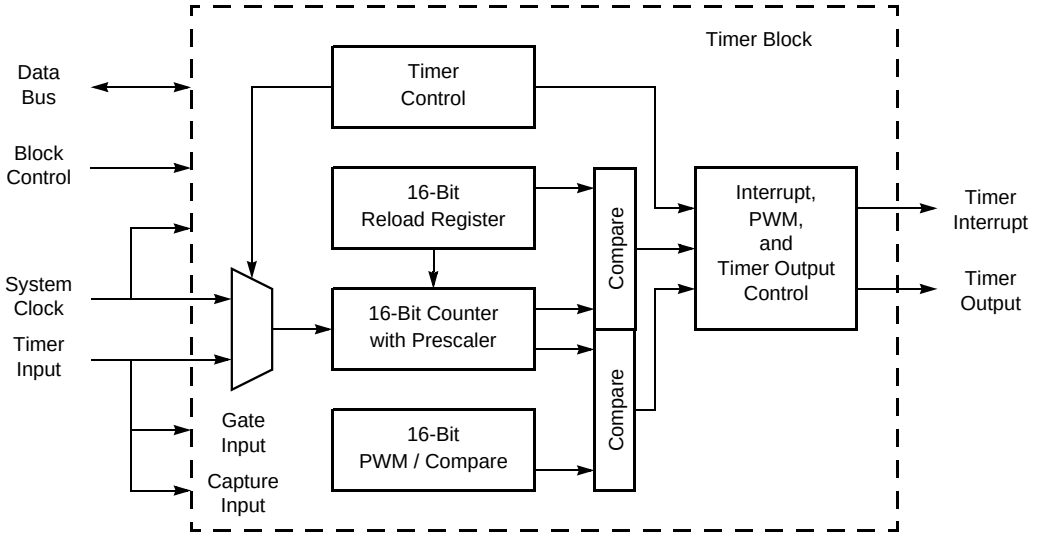



**Figure 66. Timer Block Diagram**

## Operation

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

### Timer Operating Modes

The timers can be configured to operate in the following modes:

#### One-Shot Mode

In One-Shot mode, the timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the Reload value, the timer generates an interrupt and the count value in the Timer High and Low Byte registers is reset to 0001H. Then, the timer is automatically disabled and stops counting.

Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state for one system clock cycle (from Low to High or from High to Low) upon timer Reload. If it is desired to have the Timer Output make a permanent state change upon One-Shot time-

out, first set the TPOL bit in the Timer Control Register to the start value before beginning One-Shot mode. Then, after starting the timer, set TPOL to the opposite bit value.

The steps for configuring a timer for One-Shot mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for One-Shot mode.
   – Set the prescale value.
   – If using the Timer Output alternate function, set the initial output level (High or Low).
2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control register to enable the timer and initiate counting.

In One-Shot mode, the system clock always provides the timer input. The timer period is given by the following equation:

$$\textbf{One-Shot Mode Time-Out Period (s)} = \frac{(\textbf{Reload Value} - \textbf{Start Value}) \times \textbf{Prescale}}{\textbf{System Clock Frequency (Hz)}}$$

### Continuous Mode

In Continuous mode, the timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the Reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) upon timer Reload.

The steps for configuring a timer for Continuous mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for Continuous mode.
   – Set the prescale value.

– If using the Timer Output alternate function, set the initial output level (High or Low).

2. Write to the Timer High and Low Byte registers to set the starting count value (usually 0001H). This only affects the first pass in Continuous mode. After the first timer Reload in Continuous mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control register to enable the timer and initiate counting.

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

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

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

### Counter Mode

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

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

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

The steps for configuring a timer for Counter mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for Counter mode.

– Select either the rising edge or falling edge of the Timer Input signal for the count. This also sets the initial logic level (High or Low) for the Timer Output alternate function. However, the Timer Output function does not have to be enabled.

2. Write to the Timer High and Low Byte registers to set the starting count value. This only affects the first pass in Counter mode. After the first timer Reload in Counter mode, counting always begins at the reset value of 0001H. Generally, in Counter mode the Timer High and Low Byte registers must be written with the value 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control register to enable the timer.

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

**Counter Mode Timer Input Transitions = Current Count Value – Start Value**

### PWM Mode

In PWM mode, the timer outputs a Pulse-Width Modulator (PWM) output signal through a GPIO Port pin. The timer input is the system clock. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM High and Low Byte registers. When the timer count value matches the PWM value, the Timer Output toggles. The timer continues counting until it reaches the Reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the Reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes.

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

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

The steps for configuring a timer for PWM mode and initiating the PWM operation are as follows:

1. Write to the Timer Control register to:

- Disable the timer
- Configure the timer for PWM mode.
- Set the prescale value.
- Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output alternate function.

2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H). This only affects the first pass in PWM mode. After the first timer reset in PWM mode, counting always begins at the reset value of 0001H.
3. Write to the PWM High and Low Byte registers to set the PWM value.
4. Write to the Timer Reload High and Low Byte registers to set the Reload value (PWM period). The Reload value must be greater than the PWM value.
5. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control register to enable the timer and initiate counting.

The PWM period is given by the following equation:

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

If an initial starting value other than 0001H is loaded into the Timer High and Low Byte registers, the One-Shot mode equation 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 given by:

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

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

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

### Capture Mode

In Capture mode, the current timer count value is recorded when the desired external Timer Input transition occurs. The Capture count value is written to the Timer PWM High and Low Byte Registers. The timer input is the system clock. The TPOL bit in the Timer Control register determines if the Capture occurs on a rising edge or a falling edge of the

Timer Input signal. When the Capture event occurs, an interrupt is generated and the timer continues counting.

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

The steps for configuring a timer for Capture mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   - Disable the timer
   - Configure the timer for Capture mode.
   - Set the prescale value.
   - Set the Capture edge (rising or falling) for the Timer Input.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. Clear the Timer PWM High and Low Byte registers to 0000H. This allows user software to determine if interrupts were generated by either a capture event or a reload. If the PWM High and Low Byte registers still contain 0000H after the interrupt, then the interrupt was generated by a Reload.
5. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin for the Timer Input alternate function.
7. Write to the Timer Control register to enable the timer and initiate counting.

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

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

### Compare Mode

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

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

The steps for configuring a timer for Compare mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for Compare mode.
   – Set the prescale value.
   – Set the initial logic level (High or Low) for the Timer Output alternate function, if desired.
2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control register to enable the timer and initiate counting.

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

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

### Gated Mode

In Gated mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the TPOL bit in the Timer Control register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is deasserted or a timer reload occurs. To determine if a Timer Input signal deassertion generated the interrupt, read the associated GPIO input value and compare to the value stored in the TPOL bit.

The timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. When reaching the Reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes (assuming the Timer Input signal is still asserted). Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) at timer reset.

The steps for configuring a timer for Gated mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer

– Configure the timer for Gated mode.
– Set the prescale value.

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

### Capture/Compare Mode

In Capture/Compare mode, the timer begins counting on the *first* external Timer Input transition. The desired transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control Register. The timer input is the system clock.

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

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

The steps for configuring a timer for Capture/Compare mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for Capture/Compare mode.
   – Set the prescale value.
   – Set the Capture edge (rising or falling) for the Timer Input.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.

5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control register to enable the timer.
7. Counting begins on the first appropriate transition of the Timer Input signal. No interrupt is generated by this first edge.

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

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

### Reading the Timer Count Values

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

### Timer Output Signal Operation

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

## Timer Control Register Definitions

Timers 0–2 are available in all packages. Timer 3 is available only in the 64-, 68- and 80-pin packages.

### Timer 0-3 High and Low Byte Registers

The Timer 0-3 High and Low Byte (TxH and TxL) registers (Tables 38 and 39) contain the current 16-bit timer count value. When the timer is enabled, a read from TxH causes the value in TxL to be stored in a temporary holding register. A read from TMRL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TMRL reads the register directly.

Writing to the Timer High and Low Byte registers while the timer is enabled is not recommended. There are no temporary holding registers available for write operations, so simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are

written during counting, the 8-bit written value is placed in the counter (High or Low Byte) at the next clock edge. The counter continues counting from the new value.

**Table 38. Timer 0-3 High Byte Register (TxH)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TH | | | | | | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F00H, F08H, F10H, F18H | | | | | | | |

**Table 39>. Timer 0-3 Low Byte Register (TxL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TL | | | | | | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F01H, F09H, F11H, F19H | | | | | | | |

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

## Timer Reload High and Low Byte Registers

The Timer 0-3 Reload High and Low Byte (TxRH and TxRL) registers (Tables 40 and 41) store a 16-bit reload value, {TRH[7:0], TRL[7:0]}. Values written to the Timer Reload High Byte register are stored in a temporary holding register. When a write to the Timer Reload Low Byte register occurs, the temporary holding register value is written to the Timer High Byte register. This operation allows simultaneous updates of the 16-bit Timer Reload value.

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

In single-byte DMA transactions to the Timer Reload High Byte register, the temporary holding register is bypassed and the value is written directly to the register. If the DMA is

set to 2-byte transfers, the temporary holding register for the Timer Reload High Byte is not bypassed.

**Table 40. Timer 0-3 Reload High Byte Register (TxRH)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TRH | | | | | | | |
| RESET | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F02H, F0AH, F12H, F1AH | | | | | | | |

**Table 41. Timer 0-3 Reload Low Byte Register (TxRL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TRL | | | | | | | |
| RESET | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F03H, F0BH, F13H, F1BH | | | | | | | |

TRH and TRL—Timer Reload Register High and Low
These two bytes form the 16-bit Reload value, {TRH[7:0], TRL[7:0]}. This value is used to set the maximum count value which initiates a timer reload to 0001H. In Compare mode, these two byte form the 16-bit Compare value.

## Timer 0-3 PWM High and Low Byte Registers

The Timer 0-3 PWM High and Low Byte (TxPWMH and TxPWML) registers (Tables 42 and 43) are used for Pulse-Width Modulator (PWM) operations. These registers also store the Capture values for the Capture and Capture/Compare modes.

**Table 42. Timer 0-3 PWM High Byte Register (TxPWMH)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PWMH | | | | | | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F04H, F0CH, F14H, F1CH | | | | | | | |

**Table 43. Timer 0-3 PWM Low Byte Register (TxPWML)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PWML | | | | | | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F05H, F0DH, F15H, F1DH | | | | | | | |

PWMH and PWML—Pulse-Width Modulator High and Low Bytes
These two bytes, {PWMH[7:0], PWML[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the PWM output changes state. The PWM output value is set by the TPOL bit in the Timer Control Register (TxCTL) register.

The TxPWMH and TxPWML registers also store the 16-bit captured timer value when operating in Capture or Capture/Compare modes.

## Timer 0-3 Control Registers

The Timer 0-3 Control (TxCTL) registers enable/disable the timers, set the prescaler value, and determine the timer operating mode.

**Table 44. Timer 0-3 Control Register (TxCTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TEN | TPOL | PRES | | | TMODE | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F07H, F0FH, F17H, F1FH | | | | | | | |

TEN—Timer Enable
0 = Timer is disabled.
1 = Timer enabled to count.

TPOL—Timer Input/Output Polarity
Operation of this bit is a function of the current operating mode of the timer.

**One-Shot mode**
When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer Reload.

**Continuous mode**
When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer Reload.

**Counter mode**
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.

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

**Capture mode**
0 = Count is captured on the rising edge of the Timer Input signal.
1 = Count is captured on the falling edge of the Timer Input signal.

**Compare mode**
When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer Reload.

**Gated mode**
0 = Timer counts when the Timer Input signal is High (1) and interrupts are generated on the falling edge of the Timer Input.
1 = Timer counts when the Timer Input signal is Low (0) and interrupts are generated on the rising edge of the Timer Input.

**Capture/Compare mode**
0 = Counting is started on the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal.
1 = Counting is started on the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal.

PRES—Prescale value.
The timer input clock is divided by $2^{PRES}$, where PRES can be set from 0 to 7. The prescaler is reset each time the Timer is disabled. This insures proper clock division each time the Timer is restarted.
000 = Divide by 1
001 = Divide by 2
010 = Divide by 4
011 = Divide by 8
100 = Divide by 16
101 = Divide by 32
110 = Divide by 64
111 = Divide by 128

TMODE—Timer mode
000 = One-Shot mode
001 = Continuous mode
010 = Counter mode
011 = PWM mode
100 = Capture mode
101 = Compare mode
110 = Gated mode
111 = Capture/Compare mode

# *Watch-Dog Timer*

## Overview

The Watch-Dog Timer (WDT) helps protect against corrupt or unreliable software, power faults, and other system-level problems which may place the Z8 Encore!® into unsuitable operating states. The Watch-Dog Timer includes the following features:

- On-chip RC oscillator
- A selectable time-out response: Short Reset or interrupt
- 24-bit programmable time-out value

## Operation

The Watch-Dog Timer (WDT) is a retriggerable one-shot timer that resets or interrupts the Z8F640x family device when the WDT reaches its terminal count. The Watch-Dog Timer uses its own dedicated on-chip RC oscillator as its clock source. The Watch-Dog Timer has only two modes of operation—on and off. Once enabled, it always counts and must be refreshed to prevent a time-out. An enable can be performed by executing the WDT instruction or by setting the WDT_AO Option Bit. The WDT_AO bit enables the Watch-Dog Timer to operate all the time, even if a WDT instruction has not been executed.

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

$$\text{WDT Time-out Period (ms)} = \frac{\text{WDT Reload Value}}{50}$$

where the WDT reload value is the decimal value of the 24-bit value given by {WDTU[7:0], WDTH[7:0], WDTL[7:0]} and the typical Watch-Dog Timer RC oscillator frequency is 50kHz. The Watch-Dog Timer cannot be refreshed once it reaches 000002H. The WDT Reload Value must not be set to values below 000004H. Table 45 provides

information on approximate time-out delays for the minimum and maximum WDT reload values.

**Table 45. Watch-Dog Timer Approximate Time-Out Delays**

| WDT Reload Value | WDT Reload Value | Approximate Time-Out Delay (with 50kHz typical WDT oscillator frequency) | |
|---|---|---|---|
| (Hex) | (Decimal) | Typical | Description |
| 000004 | 4 | 80µs | Minimum time-out delay |
| FFFFFF | 16,777,215 | 335.5s | Maximum time-out delay |

## Watch-Dog Timer Refresh

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

When the Z8F640x family device is operating in Debug Mode (via the On-Chip Debugger), the Watch-Dog Timer is continuously refreshed to prevent spurious Watch-Dog Timer time-outs.

## Watch-Dog Timer Time-Out Response

The Watch-Dog Timer times out when the counter reaches 000000H. A time-out of the Watch-Dog Timer generates either an interrupt or a Short Reset. The WDT_RES Option Bit determines the time-out response of the Watch-Dog Timer. Refer to the **Option Bits** chapter for information regarding programming of the WDT_RES Option Bit.

### WDT Interrupt in Normal Operation

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

### WDT Interrupt in Stop Mode

If configured to generate an interrupt when a time-out occurs and the Z8F640x family device is in STOP mode, the Watch-Dog Timer automatically initiates a STOP Mode Recovery and generates an interrupt request. Both the WDT status bit and the STOP bit in the Watch-Dog Timer Control register are set to 1 following WDT time-out in STOP

mode. Refer to the **Reset and Stop Mode Recovery** chapter for more information on STOP Mode Recovery.

If interrupts are enabled, following completion of the Stop Mode Recovery the eZ8 CPU responds to the interrupt request by fetching the Watch-Dog Timer interrupt vector and executing code from the vector address.

### WDT Reset in Normal Operation

If configured to generate a Reset when a time-out occurs, the Watch-Dog Timer forces the Z8F640x family device into the Short Reset state. The WDT status bit in the Watch-Dog Timer Control register is set to 1. Refer to the **Reset and Stop Mode Recovery** chapter for more information on Short Reset.

### WDT Reset in Stop Mode

If configured to generate a Reset when a time-out occurs and the Z8F640x family device is in STOP mode, the Watch-Dog Timer initiates a Stop Mode Recovery. Both the WDT status bit and the STOP bit in the Watch-Dog Timer Control register are set to 1 following WDT time-out in STOP mode. Refer to the **Reset and Stop Mode Recovery** chapter for more information.

## Watch-Dog Timer Reload Unlock Sequence

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

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

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

## Watch-Dog Timer Control Register Definitions

### Watch-Dog Timer Control Register

The Watch-Dog Timer Control (WDTCTL) register, detailed in Table 46, is a Read-Only register that indicates the source of the most recent Reset event, indicates a Stop Mode Recovery event, and indicates a Watch-Dog Timer time-out. Reading this register resets the upper four bits to 0.

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

**Table 46. Watch-Dog Timer Control Register (WDTCTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | POR | STOP | WDT | EXT | Reserved | | | |
| RESET | X | X | X | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| ADDR | FF0 | | | | | | | |

POR—Power-On Reset Indicator
If this bit is set to 1, a Power-On Reset event occurred. This bit is reset to 0 if a WDT time-out or Stop Mode Recovery occurs. This bit is also reset to 0 when the register is read.

STOP—STOP Mode Recovery Indicator
If this bit is set to 1, a STOP Mode Recovery occurred. If the STOP and WDT bits are both set to 1, the STOP Mode Recovery occurred due to a WDT time-out. If the STOP bit is 1 and the WDT bit is 0, the STOP Mode Recovery was not caused by a WDT time-out. This bit is reset by a Power-On Reset or a WDT time-out that occurred while not in STOP mode. Reading this register also resets this bit.

WDT—Watch-Dog Timer Time-Out Indicator
If this bit is set to 1, a WDT time-out occurred. A Power-On Reset resets this pin. A Stop Mode Recovery from a change in an input pin also resets this bit. Reading this register resets this bit.

EXT—External Reset Indicator
If this bit is set to 1, a Reset initiated by the external $\overline{\text{RESET}}$ pin occurred. A Power-On Reset or a Stop Mode Recovery from a change in an input pin resets this bit. Reading this register resets this bit.

Reserved
These bits are reserved and must be 0.

## Watch-Dog Timer Reload Upper, High and Low Byte Registers

The Watch-Dog Timer Reload Upper, High and Low Byte (WDTU, WDTH, WDTL) registers (Tables 47 through 49) form the 24-bit reload value that is loaded into the Watch-Dog Timer when a WDT instruction executes. The 24-bit reload value is {WDTU[7:0], WDTH[7:0], WDTL[7:0]. Writing to these registers sets the desired Reload Value. Reading from these registers returns the current Watch-Dog Timer count value.

**Caution:** The 24-bit WDT Reload Value must not be set to a value less than 000004H or unpredictable behavior may result.

**Table 47. Watch-Dog Timer Reload Upper Byte Register (WDTU)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | WDTU | | | | | | | |
| RESET | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* |
| ADDR | FF1H | | | | | | | |

R/W* - Read returns the current WDT count value. Write sets the desired Reload Value.

WDTU—WDT Reload Upper Byte

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

**Table 48. Watch-Dog Timer Reload High Byte Register (WDTH)**

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

R/W* - Read returns the current WDT count value. Write sets the desired Reload Value.

WDTH—WDT Reload High Byte

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

**Table 49. Watch-Dog Timer Reload Low Byte Register (WDTL)**

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

WDTL—WDT Reload Low

Least significant byte (LSB), Bits[7:0], of the 24-bit WDT reload value.

# *UART*

## Overview

The Universal Asynchronous Receiver/Transmitter (UART) is a full-duplex communication channel capable of handling asynchronous data transfers. The Z8F640x family device contains two fully independent UARTs. The UART uses a single 8-bit data mode with selectable parity. Features of the UART include:

- 8-bit asynchronous data transfer
- Selectable even- and odd-parity generation and checking
- Option of one or two Stop bits
- Separate transmit and receive interrupts
- Framing, parity, overrun and break detection
- Separate transmit and receive enables
- Selectable 9-bit multiprocessor (9-bit) mode
- 16-bit Baud Rate Generator (BRG)

## Architecture

The UART consists of three primary functional blocks: transmitter, receiver, and baud rate generator. The UART's transmitter and receiver function independently, but employ the same baud rate and data format. Figure 67 illustrates the UART architecture.

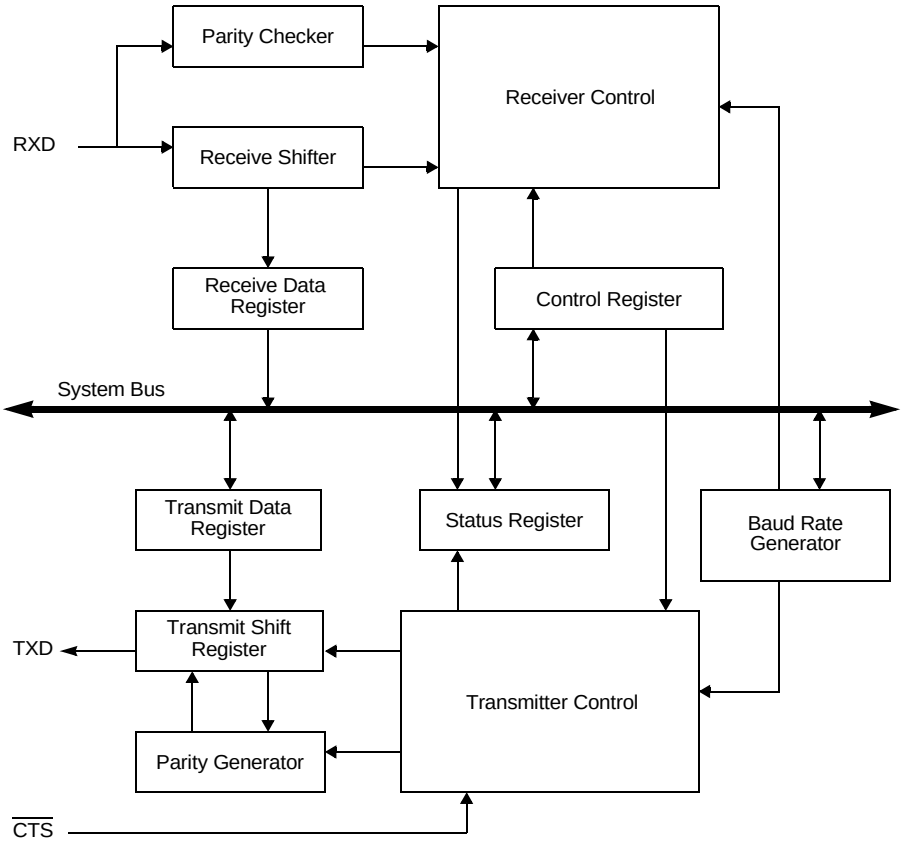



**Figure 67. UART Block Diagram**

## Operation

### Data Format

The UART always transmits and receives data in an 8-bit data format, least-significant bit first. An even or odd parity bit 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 68 and 69 illustrates the asynchronous data format employed by the UART without parity and with parity, respectively.

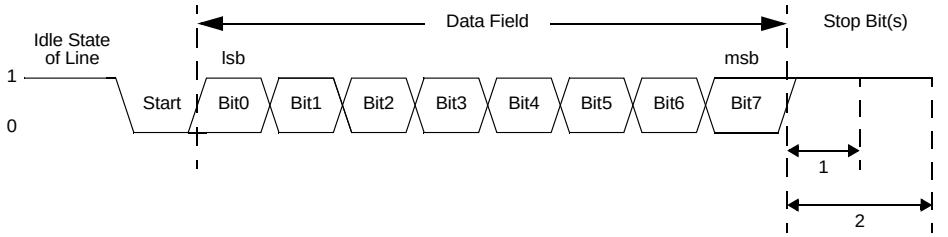



**Figure 68. UART Asynchronous Data Format without Parity**

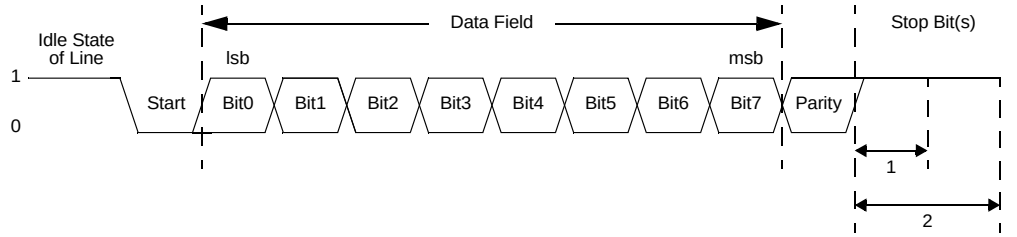



**Figure 69. UART Asynchronous Data Format with Parity**

## Transmitting Data using the Polled Method

Follow these steps to transmit data using the polled method of operation:

1. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Write to the UART Control 1 register to enable Multiprocessor (9-bit) mode functions, if desired.
4. Write to the UART Control 0 register to:
   - Set the transmit enable bit (TEN) to enable the UART for data transmission
   - Enable parity, if desired, and select either even or odd parity.
   - Set or clear the CTSE bit to enable or disable control from the receiver using the $\overline{\text{CTS}}$ pin.

5. Check the TDRE bit in the UART Status 0 register to determine if the Transmit Data register is empty (indicated by a 1). If empty, continue to Step 6. If the Transmit Data register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data register becomes available to receive new data.
6. Write the data byte to the UART Transmit Data register. The transmitter automatically transfers the data to the Transmit Shift register and transmit the data.
7. To transmit additional bits, return to Step 5.

### Transmitting Data using the Interrupt-Driven Method

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

1. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Execute a DI instruction to disable interrupts.
4. Write to the Interrupt control registers to enable the UART Transmitter interrupt and set the desired priority.
5. Write to the UART Control 1 register to enable Multiprocessor (9-bit) mode functions, if desired.
6. Write to the UART Control 0 register to:
   – Set the transmit enable bit (TEN) to enable the UART for data transmission
   – Enable parity, if desired, and select either even or odd parity.
   – Set or clear the CTSE bit to enable or disable control from the receiver via the $\overline{\text{CTS}}$ pin.
7. Execute an EI instruction to enable interrupts.

The UART is now configured for interrupt-driven data transmission. When the UART Transmit interrupt is detected, the associated interrupt service routine (ISR) should perform the following:

8. Write the data byte to the UART Transmit Data register. The transmitter will automatically transfer the data to the Transmit Shift register and transmit the data.
9. Clear the UART Transmit interrupt bit in the applicable Interrupt Request register.
10. Execute the IRET instruction to return from the interrupt-service routine and wait for the Transmit Data register to again become empty.

## Receiving Data using the Polled Method

Follow these steps to configure the UART for polled data reception:

1. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Write to the UART Control 1 register to enable Multiprocessor (9-bit) mode functions, if desired.
4. Write to the UART Control 0 register to:
   – Set the receive enable bit (REN) to enable the UART for data reception
   – Enable parity, if desired, and select either even or odd parity.
5. Check the RDA bit in the UART Status 0 register to determine if the Receive Data register contains a valid data byte (indicated by a 1). If RDA is set to 1 to indicate available data, continue to Step 6. If the Receive Data register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.
6. Read data from the UART Receive Data register. If operating in Multiprocessor (9-bit) mode, first read the Multiprocessor Receive flag (MPRX) to determine if the data was directed to this UART before reading the data.
7. Return to Step 6 to receive additional data.

## Receiving Data using the Interrupt-Driven Method

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

1. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Execute a DI instruction to disable interrupts.
4. Write to the Interrupt control registers to enable the UART Receiver interrupt and set the desired priority.
5. Clear the UART Receiver interrupt in the applicable Interrupt Request register.
6. Write to the UART Control 1 register to enable Multiprocessor (9-bit) mode functions, if desired.

7. Write to the UART Control 0 register to:
   – Set the receive enable bit (REN) to enable the UART for data reception
   – Enable parity, if desired, and select either even or odd parity.
8. Execute an EI instruction to enable interrupts.

The UART is now configured for interrupt-driven data reception. When the UART Receiver interrupt is detected, the associated interrupt service routine (ISR) should perform the following:

9. Check the UART Status 0 register to determine the source of the interrupt - error, break, or received data.
10. If the interrupt was due to data available, read the data from the UART Receive Data register. If operating in Multiprocessor (9-bit) mode, first read the Multiprocessor Receive flag (MPRX) to determine if the data was directed to this UART before reading the data.
11. Clear the UART Receiver interrupt in the applicable Interrupt Request register.
12. Execute the IRET instruction to return from the interrupt-service routine and await more data.

## Receiving Data using the Direct Memory Access Controller (DMA)

The DMA and UART can coordinate automatic data transfer from the UART Receive Data register to general-purpose Register File RAM. This reduces the eZ8 CPU processing overhead required to support UART data reception. The UART Receiver interrupt must then only notify the eZ8 CPU of error conditions. Follow these steps to configure the UART and DMA for automatic data handling:

1. Write to the DMA control registers to configure the DMA to transfer data from the UART Receive Data register to general-purpose Register File RAM.
2. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.
3. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
4. Write to the Interrupt control registers to enable the UART Receiver interrupt and set the desired priority.
5. Write to the UART Control 1 register to:
   – Enable Multiprocessor (9-bit) mode functions, if desired.
   – Disable the UART interrupt for received data by clearing $\overline{\text{RDAIRQ}}$ to 0.

6. Write to the UART Control 0 register to:
   – Set the receive enable bit (REN) to enable the UART for data reception
   – Enable parity, if desired, and select either even or odd parity.

The UART and DMA are now configured for data reception and automatic data transfer to the Register File. When a valid data byte is received by the UART the following occurs:

7. The UART notifies the DMA Controller that a data byte is available in the UART Receive Data register.
8. The DMA Controller requests control of the system bus from the eZ8 CPU.
9. The eZ8 CPU acknowledges the bus request.
10. The DMA Controller transfers the data from the UART Receive Data register to another location in RAM and then return bus control back to the eZ8 CPU.

The UART and DMA can continue to transfer incoming data bytes without eZ8 CPU intervention. When a UART error is detected, the UART Receiver interrupt is generated. The associated interrupt service routine (ISR) should perform the following:

11. Check the UART Status 0 register to determine the source of the UART error or break condition and then respond appropriately.

## Multiprocessor (9-bit) mode

The UART has a Multiprocessor mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In Multiprocessor (9-bit) mode (also referred to as 9-Bit mode), the multiprocessor bit (MP) is transmitted immediately following the 8-bits of data and immediately preceding the STOP bit(s) as illustrated in Figure 70. The character format is:

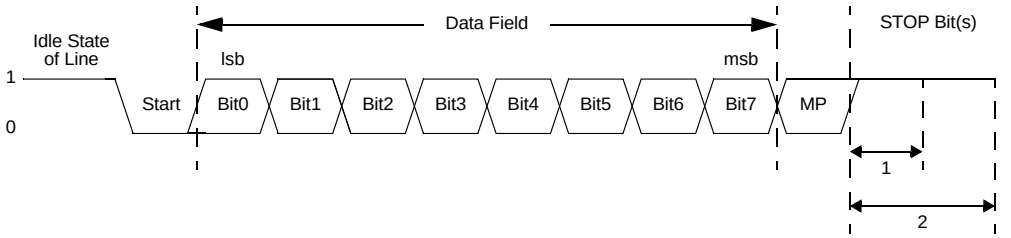



**Figure 70. UART Asynchronous Multiprocessor (9-bit) Mode Data Format**

In Multiprocessor (9-bit) mode, parity is not an option as the Parity bit location (9th bit) becomes the Multiprocessor control bit. The UART Control 1 and Status 1 registers provide multiprocessor (9-bit) mode control and status information.

## UART Interrupts

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

### Transmitter Interrupts

The transmitter generates an interrupt anytime the Transmit Data Register Empty bit (TDRE) is set to 1. This indicates that the transmitter is ready to accept new data for transmission. Writing to the UART Transmit Data register clears the UART Transmit interrupt.

### Receiver Interrupts

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

- A data byte has been received and is available in the UART Receive Data register. This interrupt can be disabled independent of the other receiver interrupt sources.
- A break is received.
- An overrun is detected.
- A data framing error is detected.

### Baud Rate Generator Interrupts

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

## UART Baud Rate Generator

The 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 UART*x* Baud Rate High and Low Byte registers combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the UART. The UART data rate is calculated using the following equation:

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

When the UART is disabled, the Baud Rate Generator can function as a basic 16-bit timer with interrupt on time-out. To configure the Baud Rate Generator as a timer with interrupt on time-out, complete the following procedure:

1. Disable the UART by clearing the REN and TEN bits in the UART Control 0 register to 0.
2. Load the desired 16-bit count value into the UART Baud Rate High and Low Byte registers.

3. Enable the Baud Rate Generator timer function and associated interrupt by setting the BIRQ bit in the UART*x* Control 1 register to 1.

## UART Control Register Definitions

The UART control registers support both the UARTs and the associated Infrared Encoder/Decoders. For more information on the infrared operation, refer to the **Infrared Encoder/Decoder** chapter on page 95.

### UART*x* Transmit Data Register

Data bytes written to the UART*x* Transmit Data register (Table 50) are shifted out on the TXD*x* pin. The Write-only UART*x* Transmit Data register shares a Register File address with the Read-only UART*x* Receive Data register.

**Table 50. UART*x* Transmit Data Register (U*x*TXD)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TXD | | | | | | | |
| RESET | X | X | X | X | X | X | X | X |
| R/W | W | W | W | W | W | W | W | W |
| ADDR | F40H and F48H | | | | | | | |

TXD—Transmit Data
UART transmitter data byte to be shifted out through the TXD*x* pin.

## UART*x* Receive Data Register

Data bytes received through the RXD*x* pin are stored in the UART*x* Receive Data register (Table 51). The Read-only UART*x* Receive Data register shares a Register File address with the Write-only UART*x* Transmit Data register.

**Table 51. UART*x* Receive Data Register (U*x*RXD)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | RXD | | | | | | | |
| RESET | X | X | X | X | X | X | X | X |
| R/W | R | R | R | R | R | R | R | R |
| ADDR | F40H and F48H | | | | | | | |

RXD—Receive Data
UART receiver data byte from the RXD*x* pin

## UART*x* Status 0 and Status 1 Registers

The UART*x* Status 0 and Status 1 registers (Table 52 and 53) identify the current UART operating configuration and status.

**Table 52. UART*x* Status 0 Register (U*x*STAT0)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | RDA | PE | OE | FE | BRKD | TDRE | TXE | CTS |
| RESET | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X |
| R/W | R | R | R | R | R | R | R | R |
| ADDR | F41H and F49H | | | | | | | |

RDA—Receive Data Available
This bit indicates that the UART Receive Data register has received data. Reading the UART Receive Data register clears this bit.
0 = The UART Receive Data register is empty.
1 = There is a byte in the UART Receive Data register.

PE—Parity Error
This bit indicates that a parity error has occurred. Reading the UART Receive Data register clears this bit.

0 = No parity error has occurred.
1 = A parity error has occurred.

OE—Overrun Error
This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the UART Receive Data register has not been read. If the RDA bit is reset to 0, then reading the UART Receive Data register clears this bit.
0 = No overrun error occurred.
1 = An overrun error occurred.

FE—Framing Error
This bit indicates that a framing error (no Stop bit following data reception) was detected. Reading the UART Receive Data register clears this bit.
0 = No framing error occurred.
1 = A framing error occurred.

BRKD—Break Detect
This bit indicates that a break occurred. If the data bits, parity/multiprocessor bit, and Stop bit(s) are all zeros then this bit is set to 1. Reading the UART Receive Data register clears this bit.
0 = No break occurred.
1 = A break occurred.

TDRE—Transmitter Data Register Empty
This bit indicates that the UART Transmit Data register is empty and ready for additional data. Writing to the UART Transmit Data register resets this bit.
0 = Do not write to the UART Transmit Data register.
1 = The UART Transmit Data register is ready to receive an additional byte to be transmitted.

TXE—Transmitter Empty
This bit indicates that the transmit shift register is empty and character transmission is finished.
0 = Data is currently transmitting.
1 = Transmission is complete.

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

**Table 53. UART*x* Status 1 Register (U*x*STAT1)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | Reserved | | | | | | | MPRX |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| ADDR | F44H and F4CH | | | | | | | |

Reserved
These bits are reserved and must be 0.

MPRX—Multiprocessor Receive
This status bit is for the receiver and reflects the actual status of the last multiprocessor bit received. Reading from the UART Data register resets this bit to 0.

## UART*x* Control 0 and Control 1 Registers

The UART*x* Control 0 and Control 1 registers (Tables 54 and 55) configure the properties of the UART's transmit and receive operations. The UART Control registers must ben be written while the UART is enabled.

**Table 54. UART*x* Control 0 Register (U*x*CTL0)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TEN | REN | CTSE | PEN | PSEL | SBRK | STOP | LBEN |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F42H and F4AH | | | | | | | |

TEN—Transmit Enable
This bit enables or disables the transmitter. The enable is also controlled by the $\overline{\text{CTS}}$ signal and the CTSE bit. If the $\overline{\text{CTS}}$ signal is low and the CTSE bit is 1, the transmitter is enabled.
0 = Transmitter disabled.
1 = Transmitter enabled.

REN—Receive Enable
This bit enables or disables the receiver.
0 = Receiver disabled.
1 = Receiver enabled.

CTSE—CTS Enable
0 = The CTS signal has no effect on the transmitter.
1 = The UART recognizes the CTS signal as an enable control from the transmitter.

PEN—Parity Enable
This bit enables or disables parity. Even or odd is determined by the PSEL bit.
0 = Parity is disabled.
1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit.

PSEL—Parity Select
0 = Even parity is transmitted and expected on all received data.
1 = Odd parity is transmitted and expected on all received data.

SBRK—Send Break
This bit pauses or breaks data transmission. Sending a break interrupts any transmission in progress, so insure that the transmitter has finished sending data before setting this bit.
0 = No break is sent.
1 = The output of the transmitter is zero.

STOP—Stop Bit Select
0 = The transmitter sends one stop bit.
1 = The transmitter sends two stop bits.

LBEN—Loop Back Enable
0 = Normal operation.
1 = All transmitted data is looped back to the receiver.

**Table 55. UART*x* Control 1 Register (U*x*CTL1)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | BIRQ | MPM | MPE | MPBT | Reserved | | RDAIRQ | IREN |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F43H and F4BH | | | | | | | |

BIRQ—Baud Rate Generator Interrupt Request
This bit sets an interrupt request when the Baud Rate Generator times out and is only set if a UART is not enabled. The is bit produces no effect when the UART is enabled.
0 = Interrupts behave as set by UART control.
1 = The Baud Rate Generator generates a receive interrupt when it counts down to zero.

MPM—Multiprocessor (9-bit) mode Select
This bit is used to enable Multiprocessor (9-bit) mode.

0 = Disable Multiprocessor mode.
1 = Enable Multiprocessor mode.

MPE—Multiprocessor Enable
0 = The UART processes all received data bytes.
1 = The UART processes only data bytes in which the multiprocessor data bit (9th bit) is set to 1.

MPBT—Multiprocessor Bit Transmitter
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).

Reserved
These bits are reserved and must be 0.

$\overline{\text{RDAIRQ}}$—Receive Data Interrupt $\overline{\text{Enable}}$
0 = Received data and receiver errors generates an interrupt request to the Interrupt Controller.
1 = Received data does not generate an interrupt request to the Interrupt Controller. Only receiver errors generate an interrupt request. The associated DMA will still be notified that received data is available.

IREN—Infrared Encoder/Decoder Enable
0 = Infrared Encoder/Decoder is disabled. UART operates normally operation.
1 = Infrared Encoder/Decoder is enabled. The UART transmits and receives data through the Infrared Encoder/Decoder.

## UART*x* Baud Rate High and Low Byte Registers

The UART*x* Baud Rate High and Low Byte registers (Tables 56 and 57) combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the UART.

**Table 56. UART*x* Baud Rate High Byte Register (U*x*BRH)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | BRH | | | | | | | |
| RESET | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F46H and F4EH | | | | | | | |

**Table 57. UART*x* Baud Rate Low Byte Register (U*x*BRL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | BRL | | | | | | | |
| RESET | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/w |
| ADDR | F47H and F4FH | | | | | | | |

The UART data rate is calculated using the following equation:

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

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

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

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

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

For reliable communication, the UART baud rate error must never exceed 5 percent. Table 58 provides information on data rate errors for popular baud rates and commonly used crystal oscillator frequencies.

**Table 58. UART Baud Rates**

**20.0 MHz System Clock**

| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
|---|---|---|---|
| 1250.0 | 1 | 1250.0 | 0.00 |
| 625.0 | 2 | 625.0 | 0.00 |
| 250.0 | 5 | 250.0 | 0.00 |
| 115.2 | 11 | 113.6 | -1.36 |
| 57.6 | 22 | 56.8 | -1.36 |
| 38.4 | 33 | 37.9 | -1.36 |
| 19.2 | 65 | 19.2 | 0.16 |
| 9.60 | 130 | 9.62 | 0.16 |
| 4.80 | 260 | 4.81 | 0.16 |
| 2.40 | 521 | 2.40 | -0.03 |
| 1.20 | 1042 | 1.20 | -0.03 |
| 0.60 | 2083 | 0.60 | 0.02 |
| 0.30 | 4167 | 0.30 | -0.01 |

**18.432 MHz System Clock**

| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
|---|---|---|---|
| 1250.0 | 1 | 1152.0 | -7.84% |
| 625.0 | 2 | 576.0 | -7.84% |
| 250.0 | 5 | 230.4 | -7.84% |
| 115.2 | 10 | 115.2 | 0.00 |
| 57.6 | 20 | 57.6 | 0.00 |
| 38.4 | 30 | 38.4 | 0.00 |
| 19.2 | 60 | 19.2 | 0.00 |
| 9.60 | 120 | 9.60 | 0.00 |
| 4.80 | 240 | 4.80 | 0.00 |
| 2.40 | 480 | 2.40 | 0.00 |
| 1.20 | 960 | 1.20 | 0.00 |
| 0.60 | 1920 | 0.60 | 0.00 |
| 0.30 | 3840 | 0.30 | 0.00 |

**16.667 MHz System Clock**

| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
|---|---|---|---|
| 1250.0 | 1 | 1041.69 | -16.67 |
| 625.0 | 2 | 520.8 | -16.67 |
| 250.0 | 4 | 260.4 | 4.17 |
| 115.2 | 9 | 115.7 | 0.47 |
| 57.6 | 18 | 57.87 | 0.47 |
| 38.4 | 27 | 38.6 | 0.47 |
| 19.2 | 54 | 19.3 | 0.47 |
| 9.60 | 109 | 9.56 | -0.45 |
| 4.80 | 217 | 4.80 | -0.83 |
| 2.40 | 434 | 2.40 | 0.01 |
| 1.20 | 868 | 1.20 | 0.01 |
| 0.60 | 1736 | 0.60 | 0.01 |
| 0.30 | 3472 | 0.30 | 0.01 |

**11.0592 MHz System Clock**

| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
|---|---|---|---|
| 1250.0 | N/A | N/A | N/A |
| 625.0 | 1 | 691.2 | 10.59 |
| 250.0 | 3 | 230.4 | -7.84 |
| 115.2 | 6 | 115.2 | 0.00 |
| 57.6 | 12 | 57.6 | 0.00 |
| 38.4 | 18 | 38.4 | 0.00 |
| 19.2 | 36 | 19.2 | 0.00 |
| 9.60 | 72 | 9.60 | 0.00 |
| 4.80 | 144 | 4.80 | 0.00 |
| 2.40 | 288 | 2.40 | 0.00 |
| 1.20 | 576 | 1.20 | 0.00 |
| 0.60 | 1152 | 0.60 | 0.00 |
| 0.30 | 2304 | 0.30 | 0.00 |

**Table 58. UART Baud Rates (Continued)**

| 10.0 MHz System Clock | | | | 5.5296 MHz System Clock | | | |
|---|---|---|---|---|---|---|---|
| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) | Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
| 1250.0 | N/A | N/A | N/A | 1250.0 | N/A | N/A | N/A |
| 625.0 | 1 | 625.0 | 0.00 | 625.0 | N/A | N/A | N/A |
| 250.0 | 3 | 208.33 | -16.67 | 250.0 | 1 | 345.6 | 38.24 |
| 115.2 | 5 | 125.0 | 8.51 | 115.2 | 3 | 115.2 | 0.00 |
| 57.6 | 11 | 56.8 | -1.36 | 57.6 | 6 | 57.6 | 0.00 |
| 38.4 | 16 | 39.1 | 1.73 | 38.4 | 9 | 38.4 | 0.00 |
| 19.2 | 33 | 18.9 | 0.16 | 19.2 | 18 | 19.2 | 0.00 |
| 9.60 | 65 | 9.62 | 0.16 | 9.60 | 36 | 9.60 | 0.00 |
| 4.80 | 130 | 4.81 | 0.16 | 4.80 | 72 | 4.80 | 0.00 |
| 2.40 | 260 | 2.40 | -0.03 | 2.40 | 144 | 2.40 | 0.00 |
| 1.20 | 521 | 1.20 | -0.03 | 1.20 | 288 | 1.20 | 0.00 |
| 0.60 | 1042 | 0.60 | -0.03 | 0.60 | 576 | 0.60 | 0.00 |
| 0.30 | 2083 | 0.30 | 0.02 | 0.30 | 1152 | 0.30 | 0.00 |

| 3.579545 MHz System Clock | | | | 1.8432 MHz System Clock | | | |
|---|---|---|---|---|---|---|---|
| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) | Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
| 1250.0 | N/A | N/A | N/A | 1250.0 | N/A | N/A | N/A |
| 625.0 | N/A | N/A | N/A | 625.0 | N/A | N/A | N/A |
| 250.0 | 1 | 223.72 | -10.51 | 250.0 | N/A | N/A | N/A |
| 115.2 | 2 | 111.9 | -2.90 | 115.2 | 1 | 115.2 | 0.00 |
| 57.6 | 4 | 55.9 | -2.90 | 57.6 | 2 | 57.6 | 0.00 |
| 38.4 | 6 | 37.3 | -2.90 | 38.4 | 3 | 38.4 | 0.00 |
| 19.2 | 12 | 18.6 | -2.90 | 19.2 | 6 | 19.2 | 0.00 |
| 9.60 | 23 | 9.73 | 1.32 | 9.60 | 12 | 9.60 | 0.00 |
| 4.80 | 47 | 4.76 | -0.83 | 4.80 | 24 | 4.80 | 0.00 |
| 2.40 | 93 | 2.41 | 0.23 | 2.40 | 48 | 2.40 | 0.00 |
| 1.20 | 186 | 1.20 | 0.23 | 1.20 | 96 | 1.20 | 0.00 |
| 0.60 | 373 | 0.60 | -0.04 | 0.60 | 192 | 0.60 | 0.00 |
| 0.30 | 746 | 0.30 | -0.04 | 0.30 | 384 | 0.30 | 0.00 |

# *Infrared Encoder/Decoder*

## Overview

The Z8F640x family products contain two fully-functional, high-performance UART to Infrared Encoder/Decoders (Endecs). Each Infrared Endec is integrated with an on-chip UART to allow easy communication between the Z8F640x family device and IrDA Physical Layer Specification Version 1.3-compliant infrared transceivers. Infrared communication provides secure, reliable, low-cost, point-to-point communication between PCs, PDAs, cell phones, printers and other infrared enabled devices.

## Architecture

Figure 71 illustrates the architecture of the Infrared Endec.

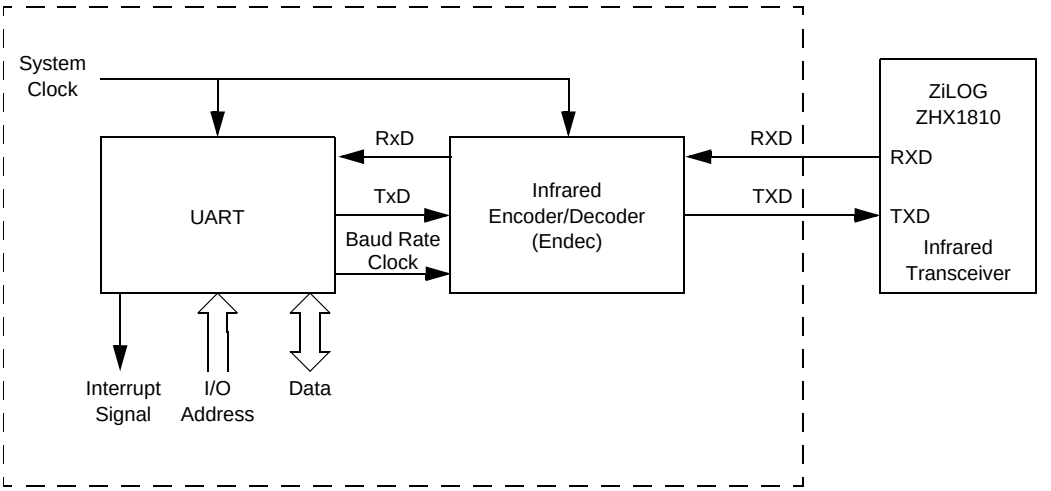



**Figure 71. Infrared Data Communication System Block Diagram**

## Operation

When the Infrared Endec is enabled, the transmit data from the associated on-chip UART is encoded as digital signals in accordance with the IrDA standard and output to the infrared transceiver via the TXD pin. Likewise, data received from the infrared transceiver is passed to the Infrared Endec via the RXD pin, decoded by the Infrared Endec, and then passed to the UART. Communication is half-duplex, which means simultaneous data transmission and reception is not allowed.

The baud rate is set by the UART's Baud Rate Generator and supports IrDA standard baud rates from 9600 baud to 115.2 kbaud. Higher baud rates are possible, but do not meet IrDA specifications. The UART must be enabled to use the Infrared Endec. The Infrared Endec data rate is calculated using the following equation:

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

### Transmitting IrDA Data

The data to be transmitted using the infrared transceiver is first sent to the UART. The UART's transmit signal (TXD) and baud rate clock are used by the IrDA to generate the modulation signal (IR_TXD) that drives the infrared transceiver. Each UART/Infrared data bit is 16-clocks wide. If the data to be transmitted is 1, the IR_TXD signal remains low for the full 16-clock period. If the data to be transmitted is 0, a 3-clock high pulse is output following a 7-clock low period. After the 3-clock high pulse, a 6-clock low pulse is output to complete the full 16-clock data period. Figure 72 illustrates IrDA data transmission. When the Infrared Endec is enabled, the UART's TXD signal is internal to the Z8F640x family device while the IR_TXD signal is output through the TXD pin.

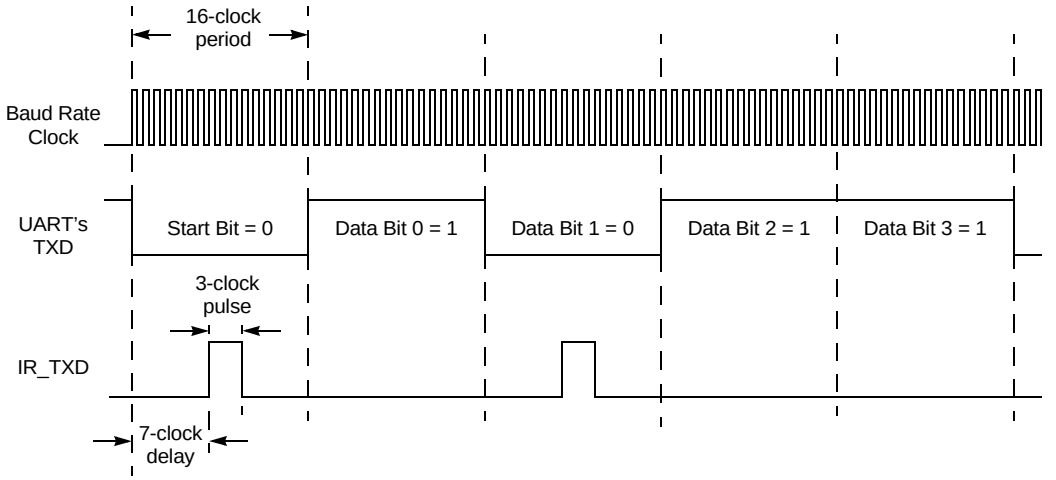



**Figure 72. Infrared Data Transmission**

## Receiving IrDA Data

Data received from the infrared transceiver via 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 73 illustrates data reception. When the Infrared Endec is enabled, the UART's RXD signal is internal to the Z8F640x family device while the IR_RXD signal is received through the RXD pin.

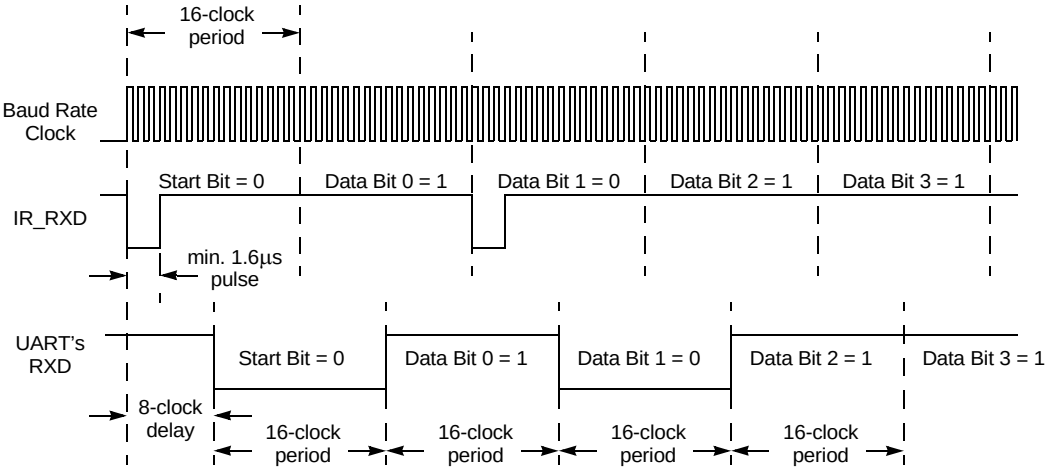



**Figure 73. Infrared Data Reception**

### Jitter

Because of the inherent sampling of the received IR_RXD signal by the bit rate clock, some jitter can be expected on the first bit in any sequence of data. All subsequent bits in the received data stream are a fixed 16-clock periods wide.

## 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 86.

**Caution:** To prevent spurious signals during IrDA data transmission, set the IREN bit in the UART*x* Control 1 register to 1 to enable the Infrared Encoder/Decoder *before* enabling the GPIO Port alternate function for the corresponding pin.

# *Serial Peripheral Interface*

## Overview

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

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

## Architecture

The SPI may be configured as either a Master (in single or multi-master systems) or a Slave as illustrated in Figures 74 through 76.

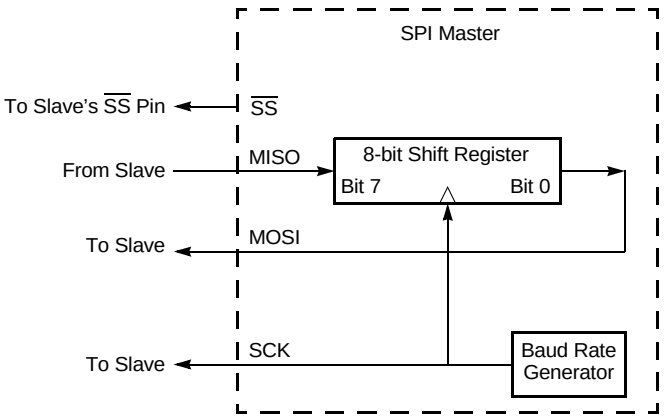



**Figure 74. SPI Configured as a Master in a Single Master, Single Slave System**

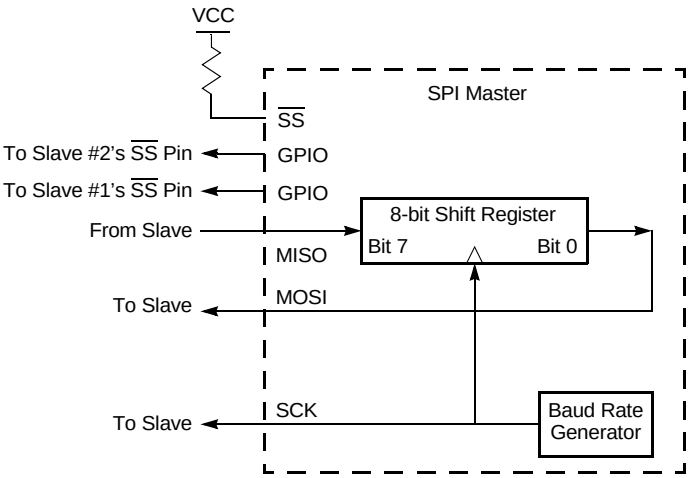



**Figure 75. SPI Configured as a Master in a Single Master, Multiple Slave System**

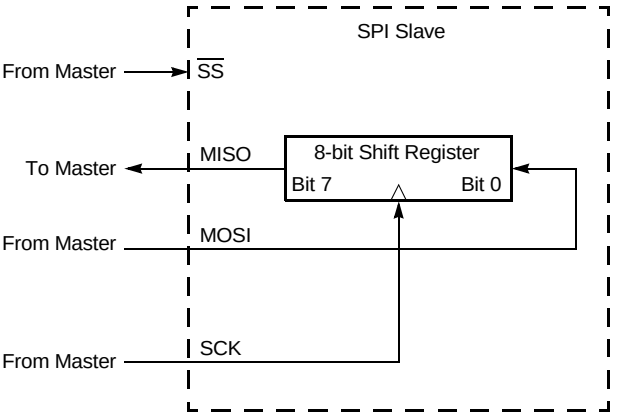



**Figure 76. SPI Configured as a Slave**

## Operation

The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (serial clock, transmit, receive and Slave select). The SPI block consists of trans-

mitter and receiver sections, a Baud Rate (clock) Generator and a control unit. The transmitter and receiver sections use the same clock.

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

## SPI Signals

The four basic SPI signals are:

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

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

### Master-In, Slave-Out

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

### Master-Out, Slave-In

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

### Serial Clock

The Serial Clock (SCK) is used to synchronize data movement both in and out of the device through its MOSI and MISO pins. In Master mode, the SPI's Baud Rate Generator creates the serial clock. The Master drives the serial clock out its own SCK pin to the Slave's SCK pin. When the SPI is configured as a Slave, the SCK pin is an input and the clock signal from the Master synchronizes the data transfer between the Master and Slave devices. Slave devices ignore the SCK signal, unless the $\overline{SS}$ pin is asserted.

The Master and Slave are each capable of exchanging a byte of data during a sequence of eight clock cycles. In both Master and Slave SPI devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. Edge polarity is determined by the SPI phase and polarity control.

### Slave Select

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

When the SPI on the Z8F640x family device is configured as the only Master in an SPI system, the $\overline{SS}$ pin can be set as either an input or an output. For communication between the Z8F640x family device SPI Master and external Slave devices, the $\overline{SS}$ signal, as an output, can assert the $\overline{SS}$ input pin on one of the Slave devices. Other GPIO output pins can also be employed to select external SPI Slave devices.

When the SPI on the Z8F640x family device is configured as one Master in a multi-master SPI system, the $\overline{SS}$ pin on the should be set as an input. The $\overline{SS}$ input signal on the Master must be High. If the $\overline{SS}$ signal goes Low (indicating another Master is driving the SPI bus), a Mode Fault error flag is set in the SPI Status register.

## SPI Clock Phase and Polarity Control

The SPI supports four combinations of serial clock phase and polarity using two bits in the SPI Control register. The clock polarity bit, CLKPOL, selects an active high or active low clock and has no effect on the transfer format. Table 59 lists the SPI Clock Phase and Polarity Operation parameters. The clock phase bit, PHASE, selects one of two fundamentally different transfer formats. For proper data transmission, the clock phase and polarity must be identical for the SPI Master and the SPI Slave. The Master always places data on the MOSI line a half-cycle before the clock edge (SCK signal), in order for the Slave to latch the data.

**Table 59. SPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation**

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

### Transfer Format PHASE Equals Zero

Figure 77 illustrates the timing diagram for an SPI transfer in which PHASE is cleared to 0. The two SCK waveforms show polarity with CLKPOL reset to 0 and with CLKPOL set to one. The diagram may be interpreted as either a Master or Slave timing diagram since the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the Master and the Slave.

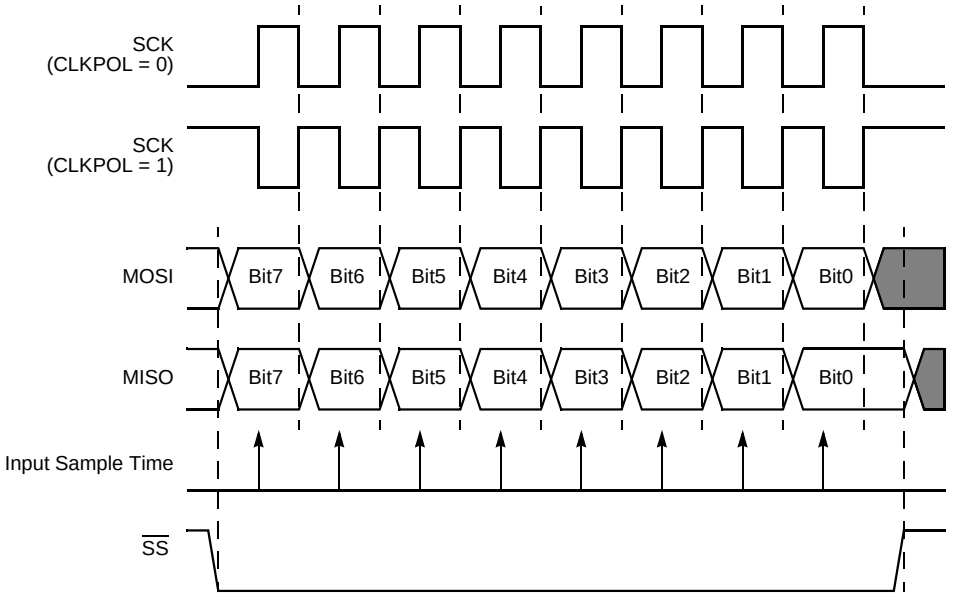



**Figure 77. SPI Timing When PHASE is 0**

### Transfer Format PHASE Equals One

Figure 78 illustrates the timing diagram for an SPI transfer in which PHASE is one. Two waveforms are depicted for SCK, one for CLKPOL reset to 0 and another for CLKPOL set to 1.

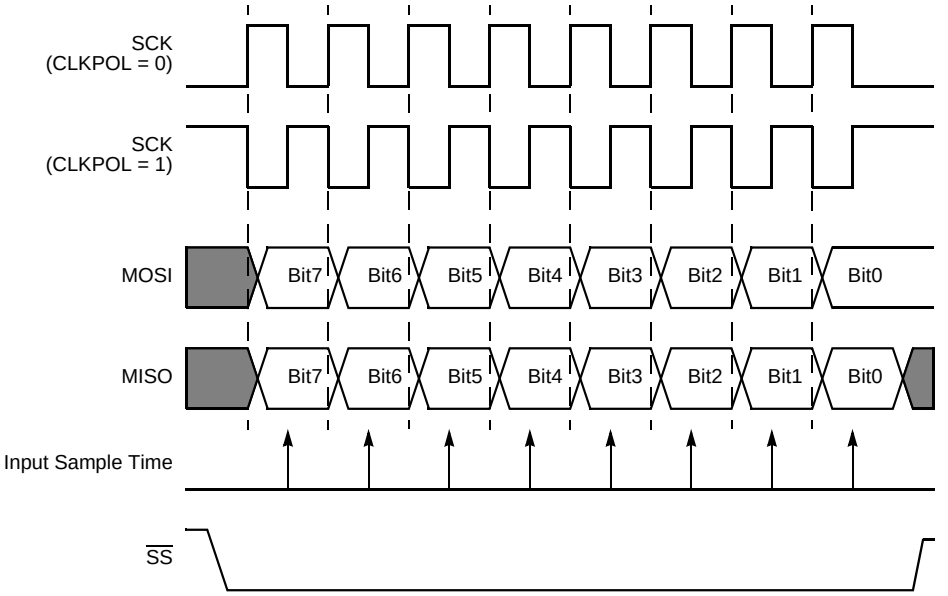



**Figure 78. SPI Timing When PHASE is 1**

## Multi-Master Operation

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

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

## Error Detection

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

### Overrun (Write Collision)

An overrun error (write collision) indicates a write to the SPI Data register was attempted while a data transfer is in progress. An overrun sets the OVR bit in the SPI Status register to 1. Writing a 1 to OVR clears this error flag.

### Mode Fault (Multi-Master Collision)

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

## SPI Interrupts

When SPI interrupts are enabled, the SPI generates an interrupt after data transmission. The SPI in Master mode generates an interrupt after a character has been sent. A character can be defined to be 1 through 8 bits by the NUMBITS field in the SPI Mode register. The SPI in Slave mode generates an interrupt when the $\overline{SS}$ signal deasserts to indicate completion of the data transfer. Writing a 1 to the IRQ bit in the SPI Status Register clears the pending interrupt request. If the SPI is disabled, an SPI interrupt can be generated by a Baud Rate Generator time-out.

## SPI Baud Rate Generator

In SPI 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 SPI Baud Rate High and Low Byte registers combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator. The reload value must be greater than or equal to 0002H for SPI operation (maximum baud rate is system clock frequency divided by 4). The SPI baud rate is calculated using the following equation:

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

When the SPI is disabled, the Baud Rate Generator can function as a basic 16-bit timer with interrupt on time-out. To configure the Baud Rate Generator as a timer with interrupt on time-out, complete the following procedure:

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

## SPI Control Register Definitions

### SPI Data Register

The SPI Data register stores both the outgoing (transmit) data and the incoming (received) data. Reads from the SPI Data register always return the current contents of the 8-bit shift register.

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

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

**Table 60. SPI Data Register (SPIDATA)**

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

DATA—Data
Transmit and/or receive data.

## SPI Control Register

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

**Table 61. SPI Control Register (SPICTL)**

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

IRQE—Interrupt Request Enable
0 = SPI interrupts are disabled. No interrupt requests are sent to the Interrupt Controller.
1 = SPI interrupts are enabled. Interrupt requests are sent to the Interrupt Controller.

STR—Start an SPI Interrupt Request
0 = No effect.
1 = Setting this bit to 1 also sets the IRQ bit in the SPI Status register to 1. Setting this bit forces the SPI to send an interrupt request to the Interrupt Control. This bit can be used by software for a function similar to transmit buffer empty in a UART.

BIRQ—BRG Timer Interrupt Request
If the SPI is enabled, this bit has no effect. If the SPI is disabled:
0 = The Baud Rate Generator timer function is disabled.
1 = The Baud Rate Generator timer function and time-out interrupt are enabled.

PHASE—Phase Select
Sets the phase relationship of the data to the clock. Refer to the **SPI Clock Phase and Polarity Control** section for more information on operation of the PHASE bit.

CLKPOL—Clock Polarity
0 = SCK idles Low (0).
1 = SCK idle High (1).

WOR—Wire-OR (Open-Drain) Mode Enabled
0 = SPI signal pins not configured for open-drain.
1 = All four SPI signal pins (SCK, $\overline{SS}$, MISO, MOSI) configured for open-drain function. This setting is typically used for multi-master and/or multi-slave configurations.

MMEN—SPI Master Mode Enable
0 = SPI configured in Slave mode.
1 = SPI configured in Master mode.

SPIEN—SPI Enable
0 = SPI disabled.
1 = SPI enabled.

## SPI Status Register

The SPI Status register indicates the current state of the SPI.

**Table 62. SPI Status Register (SPISTAT)**

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

IRQ—Interrupt Request
0 = No SPI interrupt request pending.
1 = SPI interrupt request is pending.

OVR—Overrun
0 = An overrun error has not occurred.
1 = An overrun error has been detected.

COL—Collision
0 = A multi-master collision (mode fault) has not occurred.
1 = A multi-master collision (mode fault) has been detected.

Reserved
These bits are reserved and must be 0.

TXST—Transmit Status
0 = No data transmission currently in progress.
1 = Data transmission currently in progress.

SLAS—Slave Select
If SPI enabled as a Slave,
0 = $\overline{SS}$ input pin is asserted (Low)
1 = $\overline{SS}$ input is not asserted (High).
If SPI enabled as a Master, this bit is not applicable.

## SPI Mode Register

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

**Table 63. SPI Mode Register (SPIMODE)**

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

Reserved
These bits are reserved and must be 0.

NUMBITS[2:0]—Number of Data Bits Per Character to Transfer
This field contains the number of bits to shift for each character transfer. Refer to the SPI Data Register description for information on valid bit positions when the character length is less than 8-bits.

000 = 8 bits
001 = 1 bit
010 = 2 bits
011 = 3 bits
100 = 4 bits
101 = 5 bits
110 = 6 bits
111 = 7 bits.

SSIO—Slave Select I/O
0 = $\overline{SS}$ pin configured as an input.
1 = $\overline{SS}$ pin configured as an output (Master mode only).

SSV—Slave Select Value
If SSIO = 1 and SPI configured as a Master:
0 = $\overline{SS}$ pin driven Low (0).
1 = $\overline{SS}$ pin driven High (1).
This bit has no effect if SSIO = 0 or SPI configured as a Slave.

## SPI Baud Rate High and Low Byte Registers

The SPI Baud Rate High and Low Byte registers combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator. The reload value must be greater than or equal to 0002H for proper SPI operation (maximum baud rate is system clock frequency divided by 4). The SPI baud rate is calculated using the following equation:

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

**Table 64. SPI Baud Rate High Byte Register (SPIBRH)**

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

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

**Table 65. SPI Baud Rate Low Byte Register (SPIBRL)**

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

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

# I²C Controller

## Overview

The I²C Controller makes the Z8F640x family device bus-compatible with the I²C™ protocol. The I²C Controller consists of two bidirectional bus lines—a serial data signal (SDA) and a serial clock signal (SCL). Features of the I²C Controller include:

- Transmit and Receive Operation in Master mode
- Maximum data rate of 400kbit/sec
- 7- and 10-bit Addressing Modes for Slaves
- Unrestricted Number of Data Bytes Transmitted per Transfer

The I²C Controller in the Z8F640x family device does not operate in Slave mode.

## Operation

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

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

### SDA and SCL Signals

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

The master (I²C) is responsible for driving the SCL clock signal, although the clock signal can become skewed by a slow slave device. During the high period of the clock, the slave pulls the SCL signal Low to suspend the transaction. When the slave has released the line, the I²C Controller continues the transaction. All data is transferred in bytes and there is no

limit to the amount of data transferred in one operation. When transmitting data or acknowledging read data from the slave, the SDA signal changes in the middle of the low period of SCL and is sampled in the middle of the high period of SCL.

### I²C Interrupts

the I²C Controller contains three sources of interrupts—Transmit, Receive and Not Acknowledge (NAK) interrupts. NAK interrupts occur when a Not Acknowledge is received from the slave or sent by the I²C Controller and the Start or Stop bit is set. This source sets bit 0 and can only be cleared by setting the Start or Stop bit. When this interrupt occurs, the I²C Controller waits until it is cleared before performing any action. In an interrupt service routine, this interrupt must be the first thing polled. Receive interrupts occur when a byte of data has been received by the I²C master. This interrupt is cleared by reading from the I²C Data register. If no action is taken, the I²C Controller waits until this interrupt is cleared before performing any other action.

For Transmit interrupts to occur, the TXI bit must be 1 in the I²C Control register. Transmit interrupts occur under the following conditions when the transmit data register is empty:

- The I²C Controller is idle (not performing an operation).
- The START bit is set and there is no valid data in the I²C Shift or I²C Data register to shift out.
- The first bit of the byte of an address is shifting out and the RD bit of the I²C Status register is deasserted.
- The first bit of a 10-bit address shifts out.
- The first bit of write data shifted out.

**Note:** Writing to the I²C Data register always clears a Transmit interrupt.

### Start and Stop Conditions

The master (I²C) drives all Start and Stop signals and initiates all transactions. To start a transaction, the I²C Controller generates a START condition by pulling the SDA signal low while SCL is high. Then a high-to-low transition occurs on the SDA signal while the clock is High. To complete a transaction, the I²C Controller generates a Stop condition by creating a low-to-high transition of the SDA signal in the middle of the high period of the SCL signal.When the SCL signal is High, the master generates a Start bit by pulling a High SDA signal Low and generates a Stop bit by releasing the SDA signal. The Start and Stop signals are found in the I²C Control register and must be written by software when the Z8F640x family device must begin or end a transaction.

### Writing a Transaction with a 7-Bit Address

1. The I²C Controller shifts the I²C Shift register out onto SDA signal.

2. The I2C Controller waits for the slave to send an Acknowledge (by pulling the SDA signal Low). If the slave pulls the SDA signal High (Not-Acknowledge), the I2C Controller sends a Stop signal.
3. If the slave needs to service an interrupt, it pulls the SCL signal Low, which halts I2C operation.
4. If there is no other data in the I2C Data register or the STOP bit in the I2C Control register is set by software, then the Stop signal is sent.

Figure 79 illustrates the data transfer format for a 7-bit addressed slave. Shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

**Figure 79. 7-Bit Addressed Slave Data Transfer Format**

The data transfer format for a transmit operation on a 7-bit addressed slave is as follows:

1. Software asserts the IEN bit in the I2C Control register.
2. Software asserts the TXI bit of the I2C Control register to enable Transmit interrupts.
3. The I2C interrupt asserts, because the I2C Data register is empty
4. Software responds to the TDRE bit by writing a 7-bit slave address followed by a 0 (write) to the I2C Data register.
5. Software asserts the START bit of the I2C Control register.
6. The I2C Controller sends the START condition to the I2C slave.
7. The I2C Controller loads the I2C Shift register with the contents of the I2C Data register.
8. After one bit of address has been shifted out by the SDA signal, the Transmit interrupt is asserted.
9. Software responds by writing the contents of the data into the I2C Data register.
10. The I2C Controller shifts the rest of the address and write bit out by the SDA signal.
11. 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.
12. The I2C Controller loads the contents of the I2C Shift register with the contents of the I2C Data register.
13. The I2C Controller shifts the data out of via the SDA signal. After the first bit is sent, the Transmit interrupt is asserted.

14. Software responds by setting the STOP bit of the I2C Control register.
15. If no new data is to be sent or address is to be sent, software responds by clearing the TXI bit of the I2C Control register.
16. The I2C Controller completes transmission of the data on the SDA signal.
17. The I2C Controller sends the STOP condition to the I2C bus.

## Writing a Transaction with a 10-Bit Address

1. The I2C Controller shifts the I2C Shift register out onto SDA signal.
2. The I2C Controller waits for the slave to send an Acknowledge (by pulling the SDA signal Low). If the slave pulls the SDA signal High (Not-Acknowledge), the I2C Controller sends a Stop signal.
3. If the slave needs to service an interrupt, it pulls the SCL signal low, which halts I2C operation.
4. If there is no other data in the I2C Data register or the STOP bit in the I2C Control register is set by software, then the Stop signal is sent.

The data transfer format for a 10-bit addressed slave is illustrated in the figure below. Shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

**Figure 80. 10-Bit Addressed Slave Data Transfer Format**

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

The data transfer format for a transmit operation on a 10-bit addressed slave is as follows:

1. Software asserts the IEN bit in the I2C Control register.
2. Software asserts the TXI bit of the I2C Control register to enable Transmit interrupts.
3. The I2C interrupt asserts because the I2C Data register is empty.
4. Software responds to the TDRE bit by writing the first slave address byte. The least-significant bit must be 0 for the write operation.
5. Software asserts the START bit of the I2C Control register.
6. The I2C Controller sends the START condition to the I2C slave.

7. The I2C Controller loads the I2C Shift register with the contents of the I2C Data register.
8. After one bit of address is shifted out by the SDA signal, the Transmit interrupt is asserted.
9. Software responds by writing the second byte of address into the contents of the I2C Data register.
10. The I2C Controller shifts the rest of the first byte of address and write bit out by 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.
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 by the SDA signal. After the first bit has been sent, the Transmit interrupt is asserted.
14. Software responds by writing the data to be written out to the I2C Control register.
15. The I2C Controller shifts out the rest of the second byte of slave address by 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.
17. The I2C Controller shifts the data out by the SDA signal. After the first bit is sent, the Transmit interrupt is asserted.
18. Software responds by asserting the STOP bit of the I2C Control register.
19. The I2C Controller completes transmission of the data on the SDA signal.
20. The I2C Controller sends the STOP condition to the I2C bus.

## Reading a Transaction with a 7-Bit Address

Figure 81 illustrates the data transfer format for a receive operation on a 7-bit addressed slave. The shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

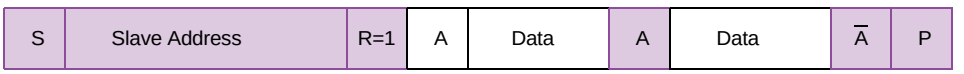



**Figure 81. Receive Data Transfer Format for a 7-Bit Addressed Slave**

The data transfer format for a receive operation on a 7-bit addressed slave is as follows:

1. Software writes the $I^2C$ Data register with a 7-bit slave address followed by a 1 (read).
2. Software asserts the START bit of the $I^2C$ Control register.
3. Software asserts the NAK bit of the $I^2C$ Control register so that after the first byte of data has been read by the $I^2C$ Controller, a Not Acknowledge is sent to the $I^2C$ slave.
4. The $I^2C$ Controller sends the START condition.
5. The $I^2C$ Controller sends the address and read bit by the SDA signal.
6. The $I^2C$ slave sends an Acknowledge by pulling the SDA signal Low during the next high period of SCL.
7. The $I^2C$ Controller reads the first byte of data from the $I^2C$ slave.
8. The $I^2C$ Controller asserts the Receive interrupt.
9. Software responds by reading the $I^2C$ Data register.
10. The $I^2C$ Controller sends a NAK to the $I^2C$ slave.
11. A NAK interrupt is generated by the $I^2C$ Controller.
12. Software responds by setting the STOP bit of the $I^2C$ Control register.
13. A STOP condition is sent to the $I^2C$ slave.

## Reading a Transaction with a 10-Bit Address

Figure 82 illustrates the receive format for a 10-bit addressed slave. The shaded regions indicate data transferred from the $I^2C$ Controller to slaves and unshaded regions indicate data transferred from the slaves to the $I^2C$ Controller.

| S | Slave Address 1st 7 bits | W=0 | A | Slave address 2nd Byte | A | S | Slave Address 1st 7 bits | R=1 | A | Data | A | Data | $\overline{A}$ | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**Figure 82. Receive Data Format for a 10-Bit Addressed Slave**

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

The data transfer format for a receive operation on a 10-bit addressed slave is as follows:

1. Software writes an address 11110B followed by the two address bits and a 0 (write).
2. Software asserts the START bit of the $I^2C$ Control register.
3. The $I^2C$ Controller sends the Start condition.

4. The I2C Controller loads the I2C Shift register with the contents of the I2C Data register.
5. After the first bit has been shifted out, a Transmit interrupt is asserted.
6. Software responds by writing eight bits of address to the I2C Data register.
7. The I2C Controller completes shifting of the two address bits and a 0 (write).
8. The I2C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
9. The I2C Controller loads the I2C Shift register with the contents of the I2C Data register.
10. The I2C Controller shifts out the next eight bits of address. After the first bits are shifted, the I2C Controller generates a Transmit interrupt.
11. Software responds by setting the START bit of the I2C Control register to generate a repeated START.
12. Software responds by writing 11110B followed by the 2-bit slave address and a 1 (read).
13. Software responds by setting the NAK bit of the I2C Control register, so that a Not Acknowledge is sent after the first byte of data has been read. If you want to read only one byte, software responds by setting the NAK bit of the I2C Control register.
14. After the I2C Controller shifts out the address bits mentioned in step 9, the I2C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
15. The I2C Controller sends the repeated START condition.
16. The I2C Controller loads the I2C Shift register with the contents of the I2C Data register.
17. The I2C Controller sends 11110B followed by the 2-bit slave read and a 1 (read).
18. The I2C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
19. The I2C slave sends a byte of data.
20. A Receive interrupt is generated.
21. Software responds by reading the I2C Data register.
22. Software responds by setting the STOP bit of the I2C Control register.
23. A NAK condition is sent to the I2C slave.
24. A STOP condition is sent to the I2C slave.

## I²C Control Register Definitions

### I²C Data Register

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

**Table 66. I²C Data Register (I2CDATA)**

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

### I²C Status Register

The Read-only I²C Status register indicates the status of the I²C Controller.

**Table 67. I²C Status Register (I2CSTAT)**

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

TDRE—Transmit Data Register Empty
When the I²C Controller is enabled, this bit is 1 when the I²C Data register is empty. When active, 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 and the interrupt are cleared by writing to the I²CD register.

RDRF—Receive Data Register Full
This bit is set active high when the I²C Controller is enabled and the I²C Controller has

received a byte of data. When active, this bit causes the I2C Controller to generate an interrupt. This bit is cleared by reading the I2C Data register.

ACK—Acknowledge
This bit indicates the status of the Acknowledge for the last byte transmitted or received. When set, this bit indicates that an Acknowledge was received for the last byte transmitted or received.

10B—10-Bit Address
This bit indicates whether a 10- or 7-bit address is being transmitted. After the START bit is set, if the five most-significant bits of the address are 11110B, this bit is set. When set, it is reset once the first byte of the address has been sent.

RD—Read
This bit indicates the direction of transfer of the data. It is active high during a read. The status of this bit is determined by the least-significant bit of the I2C Shift register after the START bit is set.

TAS—Transmit Address State
This bit is active high while the address is being shifted out of the I2C Shift register.

DSS—Data Shift State
This bit is active high while data is being transmitted to or from the I2C Shift register.

NCKI—NACK Interrupt
This bit is set high when a Not Acknowledge condition is received or sent and neither the START nor the STOP bit is active. When set, this bit generates an interrupt that can only be cleared by setting the START or STOP bit, allowing the user to specify whether he wants to perform a STOP or a repeated START.

## I2C Control Register

The I2C Control register enables the I2C operation.

**Table 68. 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/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | F52H | | | | | | | |

IEN—I2C Enable
This bit enables the I2C transmitter and receiver.

START—Send Start Condition
This bit sends the Start condition. Once asserted, it is cleared by the I2C Controller after it sends the START condition or by deasserting the IEN bit. After this bit is set, the Start condition is sent if there is data in the I2C Data or I2C Shift register. If there is no data in one of these registers, the I2C Controller waits until data is loaded. If this bit is set while the I2C Controller is shifting out data, it generates a START condition after the byte shifts and the acknowledge phase completed. If the STOP bit is also set, it also waits until the STOP condition is sent before the START condition. If this bit is 1, it cannot be cleared to 0 by writing to the register.This bit clears when the I2C is disabled.

STOP—Send Stop Condition
This bit causes the I2C Controller to issue a Stop condition after the byte in the I2C Shift register has completed transmission or after a byte has been received in a receive operation. Once 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.This bit clears when the I2C is disabled.

BIRQ—Baud Rate Generator Interrupt Request
This bit causes an interrupt to occur every time the baud rate generator counts down to zero. This bit allows the I2C Controller to be used as an additional counter when it is not being used elsewhere. This bit must only be set when the I2C Controller is disabled.

TXI—Enable TDRE interrupts
This bit enables interrupts when the I2C Data register is empty on the I2C Controller.

NAK—Send NAK
This bit sends a Not Acknowledge condition after the next byte of data has been read from the I2C slave. Once asserted, it is deasserted after a Not Acknowledge is sent or the IEN bit is deasserted.

FLUSH—Flush Data
Setting this bit to 1 clears the I2C Data register and sets the TDRE bit to 1. This bit allows flushing of the I2C Data register when an NAK is received after the data has been sent to the I2C Data register. Reading this bit always returns 0.

FILTEN—I2C Signal Filter Enable
Setting this bit to 1 enables low-pass digital filters on the SDA and SCL input signals. These filters reject any input pulse with periods less than a full system clock cycle. The filters introduce a 3-system clock cycle latency on the inputs.

## I²C Baud Rate High and Low Byte Registers

The I²C Baud Rate High and Low Byte registers 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:

$$\text{I2C Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{4 \times \text{BRG[15:0]}}$$

.

**Table 69. I²C Baud Rate High Byte Register (I2CBRH)**

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

BRH = I²C Baud Rate High Byte
Most significant byte, BRG[15:8], of the I²C Baud Rate Generator's reload value.

**Table 70. I²C Baud Rate Low Byte Register (I2CBRL)**

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

BRL = I²C Baud Rate Low Byte
Least significant byte, BRG[7:0], of the I²C Baud Rate Generator's reload value.

# *Direct Memory Access Controller*

## Overview

The Z8F640x family device's Direct Memory Access (DMA) Controller provides three independent Direct Memory Access channels. Two of the channels (DMA0 and DMA1) transfer data between the on-chip peripherals and the Register File. The third channel (DMA_ADC) controls the Analog-to-Digital Converter (ADC) operation and transfers the Single-Shot mode ADC output data to the Register File.

## Operation

### DMA0 and DMA1 Operation

DMA0 and DMA1, referred to collectively as DMA*x*, transfer data either from the on-chip peripheral control registers to the Register File, or from the Register File to the on-chip peripheral control registers. The sequence of operations in a DMA*x* data transfer is:

1. DMA*x* trigger source requests a DMA data transfer.
2. DMA*x* requests control of the system bus (address and data) from the eZ8 CPU.
3. After the eZ8 CPU acknowledges the bus request, DMA*x* transfers either a single byte or a two-byte word (depending upon configuration) and then returns system bus control back to the eZ8 CPU.
4. If Current Address equals End Address:
   - DMA*x* reloads the original Start Address
   - If configured to generate an interrupt, DMA*x* sends an interrupt request to the Interrupt Controller
   - If configured for single-pass operation, DMA*x* resets the DEN bit in the DMA*x* Control register to 0 and the DMA is disabled.

   If Current Address does not equal End Address, the Current Address increments by 1 (single-byte transfer) or 2 (two-byte word transfer).

## Configuring DMA0 and DMA1 for Data Transfer

Follow these steps to configure and enable DMA0 or DMA1:

1. Write to the DMA*x* I/O Address register to set the Register File address identifying the on-chip peripheral control register. The upper nibble of the 12-bit address for on-chip peripheral control registers is always FH. The full address is {FH, DMA*x*_IO[7:0]}
2. Determine the 12-bit Start and End Register File addresses. The 12-bit Start Address is given by {DMA*x*_H[3:0], DMA_START[7:0]}. The 12-bit End Address is given by {DMA*x*_H[7:4], DMA_END[7:0]}.
3. Write the Start and End Register File address high nibbles to the DMA*x* End/Start Address High Nibble register.
4. Write the lower byte of the Start Address to the DMA*x* Start/Current Address register.
5. Write the lower byte of the End Address to the DMA*x* End Address register.
6. Write to the DMA*x* Control register to complete the following:
   - Select loop or single-pass mode operation
   - Select the data transfer direction (either from the Register File RAM to the on-chip peripheral control register; or from the on-chip peripheral control register to the Register File RAM)
   - Enable the DMA*x* interrupt request, if desired
   - Select Word or Byte mode
   - Select the DMA*x* request trigger
   - Enable the DMA*x* channel

## DMA_ADC Operation

DMA_ADC transfers data from the ADC to the Register File. The sequence of operations in a DMA_ADC data transfer is:

1. ADC completes conversion on the current ADC input channel and signals the DMA controller that two-bytes of ADC data are ready for transfer.
2. DMA_ADC requests control of the system bus (address and data) from the eZ8 CPU.
3. After the eZ8 CPU acknowledges the bus request, DMA_ADC transfers the two-byte ADC output value to the Register File and then returns system bus control back to the eZ8 CPU.
4. If the current ADC Analog Input is the highest numbered input to be converted:
   - DMA_ADC resets the ADC Analog Input number to 0 and initiates data conversion on ADC Analog Input 0.
   - If configured to generate an interrupt, DMA_ADC sends an interrupt request to the Interrupt Controller

If the current ADC Analog Input is not the highest numbered input to be converted, DMA_ADC initiates data conversion in the next higher numbered ADC Analog Input.

### Configuring DMA_ADC for Data Transfer

Follow these steps to configure and enable DMA_ADC:

1. Write the DMA_ADC Address register with the 7 most-significant bits of the Register File address for data transfers.
2. Write to the DMA_ADC Control register to complete the following:
   – Enable the DMA_ADC interrupt request, if desired
   – Select the number of ADC Analog Inputs to convert
   – Enable the DMA_ADC channel

**Caution:** When using the DMA_ADC to perform conversions on multiple ADC inputs and the ADC_IN field in the DMA_ADC Control Register is greater than 000b, the Analog-to-Digital Converter must be configured for Single-Shot mode.

Continuous mode operation of the ADC can **only** be used in conjunction with DMA_ADC if the ADC_IN field in the DMA_ADC Control Register is reset to 000b to enable conversion on ADC Analog Input 0 only.

## DMA Control Register Definitions

### DMA*x* Control Register

The DMA*x* Control register is used to enable and select the mode of operation for DMA*x*.

**Table 71. DMA*x* Control Register (DMA*x*CTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DEN | DLE | DDIR | IRQEN | WSEL | RSS | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FB0H, FB8H | | | | | | | |

DEN—DMA*x* Enable
0 = DMA*x* is disabled and data transfer requests are disregarded.

1 = DMA*x* is enabled and initiates a data transfer upon receipt of a request from the trigger source.

DLE—DMA*x* Loop Enable
0 = DMA*x* reloads the original Start Address and is then disabled after the End Address data is transferred.
1 = DMA*x*, after the End Address data is transferred, reloads the original Start Address and continues operating.

DDIR—DMA*x* Data Transfer Direction
0 = Register File → on-chip peripheral control register.
1 = on-chip peripheral control register → Register File.

IRQEN—DMA*x* Interrupt Enable
0 = DMA*x* does not generate any interrupts.
1 = DMA*x* generates an interrupt when the End Address data is transferred.

WSEL—Word Select
0 = DMA*x* transfers a single byte per request.
1 = DMA*x* transfers a two-byte word per request. The address for the on-chip peripheral control register must be an even address.

RSS—Request Trigger Source Select
The Request Trigger Source Select field determines the peripheral that can initiate a DMA request transfer. The corresponding interrupts do not need to be enabled within the Interrupt Controller to initiate a DMA transfer. However, if the Request Trigger Source can enable or disable the interrupt request sent to the Interrupt Controller, the interrupt request must be enabled within the Request Trigger Source block.
000 = Timer 0.
001 = Timer 1.
010 = Timer 2.
011 = Timer 3.
100 = DMA0 Control register: UART0 Received Data register contains valid data. DMA1 Control register: UART0 Transmit Data register empty.
101 = DMA0 Control register: UART1 Received Data register contains valid data. DMA1 Control register: UART1 Transmit Data register empty.
110 = DMA0 Control register: $I^2C$ Receiver Interrupt. DMA1 Control register: $I^2C$ Transmitter Interrupt register empty.
111 = Reserved.

## DMA*x* I/O Address Register

The DMA*x* I/O Address register contains the low byte of the on-chip peripheral address for data transfer. The full 12-bit Register File address is given by {FH, DMA*x*_IO[7:0]}.

When the DMA is configured for two-byte word transfers, the DMA*x* I/O Address register must contain an even numbered address.

**Table 72. DMA*x* I/O Address Register (DMA*x*IO)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DMA_IO | | | | | | | |
| RESET | X | X | X | X | X | X | X | X |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FB1H, FB9H | | | | | | | |

DMA_IO—DMA on-chip peripheral control register address
This byte sets the low byte of the on-chip peripheral control register address on Register File Page FH (addresses F00H to FFFH).

## DMA*x* Address High Nibble Register

The DMA*x* Address High register specifies the upper four bits of address for the Start/Current and End Addresses of DMA*x*.

**Table 73. DMA*x* Address High Nibble Register (DMA*x*H)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DMA_END_H | | | | DMA_START_H | | | |
| RESET | X | X | X | X | X | X | X | X |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FB2H, FHAH | | | | | | | |

DMA_END_H—DMA*x* End Address High Nibble
These bits, used with the DMA*x* End Address Low register, form a 12-bit End Address. The full 12-bit address is given by {DMA_END_H[3:0], DMA_END[7:0]}.

DMA_START_H—DMA*x* Start/Current Address High Nibble
These bits, used with the DMA*x* Start/Current Address Low register, form a 12-bit Start/Current Address. The full 12-bit address is given by {DMA_START_H[3:0], DMA_START[7:0]}.

### DMA*x* Start/Current Address Low Byte Register

The DMA*x* Start/Current Address Low register, in conjunction with the DMA*x* Address High Nibble register, forms a 12-bit Start/Current Address. Writes to this register set the Start Address for DMA operations. Each time the DMA completes a data transfer, the 12-bit Start/Current Address increments by either 1 (single-byte transfer) or 2 (two-byte word transfer). Reads from this register return the low byte of the Current Address to be used for the next DMA data transfer.

**Table 74. DMA*x* Start/Current Address Low Byte Register (DMA*x*START)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DMA_START | | | | | | | |
| RESET | X | X | X | X | X | X | X | X |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FB3H, FHBH | | | | | | | |

DMA_START—DMA*x* Start/Current Address Low
These bits, with the four lower bits of the DMA*x*_H register, form the 12-bit Start/Current address. The full 12-bit address is given by {DMA_START_H[3:0], DMA_START[7:0]}.

## DMA*x* End Address Low Byte Register

The DMA*x* End Address Low Byte register, in conjunction with the DMA*x*_H register, forms a 12-bit End Address.

**Table 75. DMA*x* End Address Low Byte Register (DMA*x*END)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DMA_END | | | | | | | |
| RESET | X | X | X | X | X | X | X | X |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FB4H, FBCH | | | | | | | |

DMA_END—DMA*x* End Address Low
These bits, with the four upper bits of the DMA*x*_H register, form a 12-bit address. This address is the ending location of the DMA*x* transfer. The full 12-bit address is given by {DMA_END_H[3:0], DMA_END[7:0]}.

## DMA_ADC Address Register

The DMA_ADC Address register points to a block of the Register File to store ADC conversion values as illustrated in Table 76. This register contains the seven most-significant bits of the 12-bit Register File addresses. The five least-significant bits are calculated from the ADC Analog Input number (5-bit base address is equal to twice the ADC Analog Input number). The 10-bit ADC conversion data is stored as two bytes with the most significant byte of the ADC data stored at the even numbered Register File address.

Table 76 provides an example of the Register File addresses if the DMA_ADC Address register contains the value 72H.

**Table 76. DMA_ADC Register File Address Example**

| ADC Analog Input | Register File Address (Hex)[1] |
|---|---|
| 0 | 720H-721H |
| 1 | 722H-723H |
| 2 | 724H-725H |
| 3 | 726H-727H |
| 4 | 728H-729H |
| 5 | 72AH-72BH |
| 6 | 72CH-72DH |
| 7 | 72EH-72FH |
| 8 | 730H-731H |
| 9 | 732H-733H |
| 10 | 734H-735H |
| 11 | 736H-737H |

[1] DMAA_ADDR set to 72H.

**Table 77. DMA_ADC Address Register (DMAA_ADDR)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DMAA_ADDR | | | | | | | Reserved |
| RESET | X | X | X | X | X | X | X | X |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FBDH | | | | | | | |

DMAA_ADDR—DMA_ADC Address
These bits specify the seven most-significant bits of the 12-bit Register File addresses used for storing the ADC output data. The ADC Analog Input Number defines the five least-significant bits of the Register File address. Full 12-bit address is {DMAA_ADDR[7:1], 4-bit ADC Analog Input Number, 0}.

Reserved
This bit is reserved and must be 0.

## DMA_ADC Control Register

The DMA_ADC Control register enables and sets options (DMA enable and interrupt enable) for ADC operation.

**Table 78. DMA_ADC Control Register (DMAACTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DAEN | IRQEN | Reserved | | ADC_IN | | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | FBEH | | | | | | | |

DAEN—DMA_ADC Enable
0 = DMA_ADC is disabled and the ADC Analog Input Number (ADC_IN) is reset to 0.
1 = DMA_ADC is enabled.

IRQEN—Interrupt Enable
0 = DMA_ADC does not generate any interrupts.
1 = DMA_ADC generates an interrupt after transferring data from the last ADC Analog Input specified by the ADC_IN field.

Reserved
These bits are reserved and must be 0.

ADC_IN—ADC Analog Input Number
These bits set the number of ADC Analog Inputs to be used in the continuous update (data conversion followed by DMA data transfer). The conversion always begins with ADC Analog Input 0 and then progresses sequentially through the other selected ADC Analog Inputs.
0000 = ADC Analog Input 0 updated.
0001 = ADC Analog Inputs 0-1 updated.
0010 = ADC Analog Inputs 0-2 updated.
0011 = ADC Analog Inputs 0-3 updated.
0100 = ADC Analog Inputs 0-4 updated.
0101 = ADC Analog Inputs 0-5 updated.
0110 = ADC Analog Inputs 0-6 updated.
0111 = ADC Analog Inputs 0-7 updated.
1000 = ADC Analog Inputs 0-8 updated.
1001 = ADC Analog Inputs 0-9 updated.
1010 = ADC Analog Inputs 0-10 updated.
1011 = ADC Analog Inputs 0-11 updated.
1100-1111 = Reserved.

## DMA Status Register

The DMA Status register indicates the DMA channel that generated the interrupt and the ADC Analog Input that is currently undergoing conversion. Reads from this register reset the Interrupt Request Indicator bits (IRQA, IRQ1, and IRQ0) to 0. Therefore, software interrupt service routines that read this register must process all three interrupt sources from the DMA.

**Table 79. DMA_ADC Status Register (DMAA_STAT)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | CADC[3:0] | | | | Reserved | IRQA | IRQ1 | IRQ0 |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| ADDR | FBFH | | | | | | | |

CADC[3:0]—Current ADC Analog Input
This field identifies the Analog Input that the ADC is currently converting.

Reserved
This bit is reserved and must be 0.

IRQA—DMA_ADC Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA_ADC is not the source of the interrupt from the DMA Controller.
1 = DMA_ADC completed transfer of data from the last ADC Analog Input and generated an interrupt.

IRQ1—DMA1 Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA1 is not the source of the interrupt from the DMA Controller.
1 = DMA1 completed transfer of data to/from the End Address and generated an interrupt.

IRQ0—DMA0 Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA0 is not the source of the interrupt from the DMA Controller.
1 = DMA0 completed transfer of data to/from the End Address and generated an interrupt.

# *Analog-to-Digital Converter*

## Overview

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

- 12 analog input sources are multiplexed with general-purpose I/O ports
- Interrupt upon conversion complete
- Internal voltage reference generator
- Direct Memory Access (DMA) controller can automatically initiate data conversion and transfer of the data from 1 to 12 of the analog inputs.

## Architecture

Figure 83 illustrates the three major functional blocks (converter, analog multiplexer, and voltage reference generator) of the ADC. The ADC converts an analog input signal to its digital representation. The 12-input analog multiplexer selects one of the 12 analog input sources. The ADC requires an input reference voltage for the conversion. The voltage reference for the conversion may be input through the external VREF pin or generated internally by the voltage reference generator.

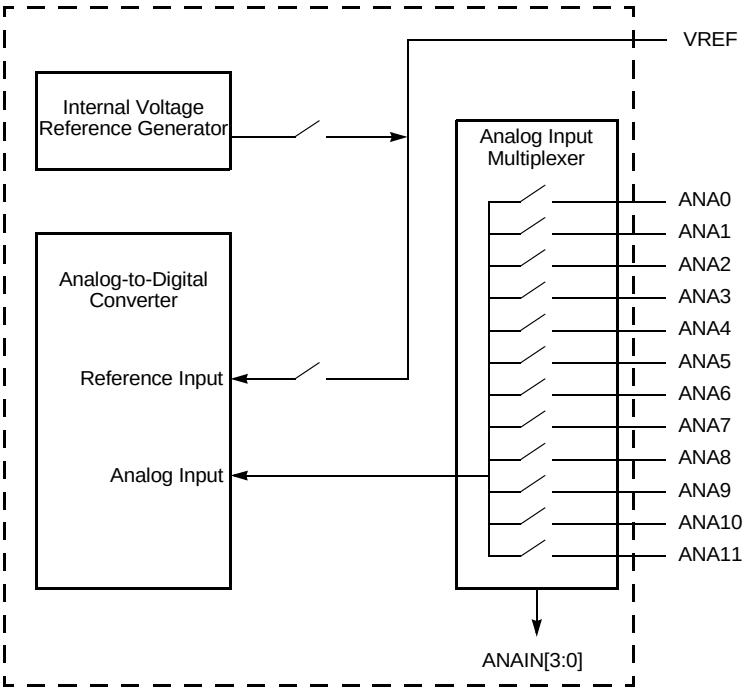



**Figure 83. Analog-to-Digital Converter Block Diagram**

## Operation

### Automatic Power-Down

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

### Single-Shot Conversion

When configured for single-shot conversion, the ADC performs a single analog-to-digital conversion on the selected analog input channel. After completion of the conversion, the ADC shuts down. The steps for setting up the ADC and initiating a single-shot conversion are as follows:

1. Enable the desired analog inputs by configuring the general-purpose I/O pins for alternate function. This configuration disables the digital input and output drivers.
2. Write to the ADC Control register to configure the ADC and begin the conversion. The bit fields in the ADC Control register can be written simultaneously:
   – Write to ANAIN[3:0] to select one of the 12 analog input sources.
   – Clear CONT to 0 to select a single-shot conversion.
   – Write to VREF to enable or disable the internal voltage reference generator.
   – Set CEN to 1 to start the conversion.
3. CEN remains 1 while the conversion is in progress. A single-shot conversion requires 5129 system clock cycles to complete. If a single-shot conversion is requested from an ADC powered-down state, the ADC uses 40 additional clock cycles to power-up before beginning the 5129 cycle conversion.
4. When the conversion is complete, the ADC control logic performs the following operations:
   – 10-bit data result written to {ADCD_H[7:0], ADCD_L[7:6]}.
   – CEN resets to 0 to indicate the conversion is complete.
   – An interrupt request is sent to the Interrupt Controller.
5. If the ADC remains idle for 160 consecutive system clock cycles, it is automatically powered-down.

## Continuous Conversion

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

**Caution:** In Continuous mode, users must be aware that ADC updates are limited by the input signal bandwidth of the ADC and the latency of the ADC and its digital filter. Step changes at the input are not seen at the next output from the ADC. The response of the ADC (in all modes) is limited by the input signal bandwidth and the latency.

The steps for setting up the ADC and initiating continuous conversion are as follows:

1. Enable the desired analog input by configuring the general-purpose I/O pins for alternate function. This disables the digital input and output driver.
2. Write to the ADC Control register to configure the ADC for continuous conversion. The bit fields in the ADC Control register may be written simultaneously:
   – Write to ANAIN[3:0] to select one of the 12 analog input sources.

– Set CONT to 1 to select continuous conversion.
– Write to VREF to enable or disable the internal voltage reference generator.
– Set CEN to 1 to start the conversions.

3. When the first conversion in continuous operation is complete (after 5129 system clock cycles, plus the 40 cycles for power-up, if necessary), the ADC control logic performs the following operations:
   – CEN resets to 0 to indicate the first conversion is complete. CEN remains 0 for all subsequent conversions in continuous operation.
   – An interrupt request is sent to the Interrupt Controller to indicate the ***first*** conversion is complete. An interrupt request is not sent for subsequent conversions in continuous operation.
4. Thereafter, the ADC writes a new 10-bit data result to {ADCD_H[7:0], ADCD_L[7:6]} every 256 system clock cycles.
5. To disable continuous conversion, clear the CONT bit in the ADC Control register to 0.

### DMA Control of the ADC

The Direct Memory Access (DMA) Controller can control operation of the ADC including analog input selection and conversion enable. For more information on the DMA and configuring for ADC operations refer to the **Direct Memory Access Controller** chapter.

## ADC Control Register Definitions

### ADC Control Register

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

**Table 80. ADC Control Register (ADCCTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | CEN | Reserved | $\overline{\text{VREF}}$ | CONT | ANAIN[3:0] | | | |
| RESET | 0 | 0 | 0 | 0 | 0000 | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | | | |
| ADDR | F70H | | | | | | | |

CEN—Conversion Enable
0 = Conversion is complete. Writing a 0 produces no effect. The ADC automatically clears

this bit to 0 when a conversion has been completed.
1 = Begin conversion. Writing a 1 to this bit starts a conversion. If a conversion is already in progress, the conversion restarts. This bit remains 1 until the conversion is complete.

Reserved
This bit is reserved and must be 0.

$\overline{\text{VREF}}$
0 = Internal voltage reference generator enabled. The VREF pin should be left unconnected (or capacitively coupled to analog ground).
1 = Internal voltage reference generator disabled. An external voltage reference must be provided through the VREF pin.

CONT
0 = Single-shot conversion. ADC data is output once at completion of the 5129 system clock cycles.
1 = Continuous conversion. ADC data updated every 256 system clock cycles.

ANAIN—Analog Input Select
These bits select the analog input for conversion. Not all Port pins in this list are available in all packages for the Z8F640x family of products. Refer to the **Signal and Pin Descriptions** chapter for information regarding the Port pins available with each package style. Do not enable unavailable analog inputs.
0000 = ANA0
0001 = ANA1
0010 = ANA2
0011 = ANA3
0100 = ANA4
0101 = ANA5
0110 = ANA6
0111 = ANA7
1000 = ANA8
1001 = ANA9
1010 = ANA10
1011 = ANA11
11XX = Reserved.

## ADC Data High Byte Register

The ADC Data High Byte register contains the upper eight bits of the 10-bit ADC output. During a conversion, this value is invalid. Access to the ADC Data High Byte register is read-only. The full 10-bit ADC result is given by {ADCD_H[7:0], ADCD_L[7:6]}.

**Table 81. ADC Data High Byte Register (ADCD_H)**

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

ADCD_H—ADC Data High Byte
This byte contains the upper eight bits of the 10-bit ADC output. These bits are not valid during a conversion. These bits are undefined after a Reset.

## ADC Data Low Bits Register

The ADC Data Low Bits register contains the lower two bits of the conversion value. During a conversion this value is invalid. Access to the ADC Data Low Bits register is read-only. The full 10-bit ADC result is given by {ADCD_H[7:0], ADCD_L[7:6]}.

**Table 82. ADC Data Low Bits Register (ADCD_L)**

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

ADCD_L—ADC Data Low Bits
These are the least significant two bits of the 10-bit ADC output. During a conversion, this value is invalid. These bits are undefined after a Reset.

Reserved
These bits are reserved and are always undefined.

# *Flash Memory*

## Overview

The Z8F640x family features up to 64KB (65,536 bytes) of non-volatile Flash memory with read/write/erase capability. The Flash Memory can be programmed and erased in-circuit by either user code or through the On-Chip Debugger.

The Flash memory array is arranged in pages with 512 bytes per page. The 512-byte page is the minimum Flash block size that can be erased. Each page is divided into 8 rows of 64 bytes. The Flash memory also contains a High Sector that can be enabled for writes and erase separately from the rest of the Flash array. The first 2 bytes of the Flash Program memory are used as Option Bits. Refer to the **Option Bits** chapter for more information on their operation.

Table 83 describes the Flash memory configuration for each device in the Z8F640x family. Figure 84 illustrates the Flash memory arrangement.

**Table 83. Z8F640x family Flash Memory Configurations**

| Part Number | Flash Size KB (Bytes) | Flash Pages | Program Memory Addresses | Flash High Sector Size KB (Bytes) | High Sector Addresses |
|---|---|---|---|---|---|
| Z8F160x | 16 (16,384) | 32 | 0000H - 3FFFH | 1 (1024) | 3C00H - 3FFFH |
| Z8F240x | 24 (24,576) | 48 | 0000H - 5FFFH | 2 (2048) | 5800H - 5FFFH |
| Z8F320x | 32 (32,768) | 64 | 0000H - 7FFFH | 2 (2048) | 7800H - 7FFFH |
| Z8F480x | 48 (49,152) | 96 | 0000H - BFFFH | 4 (4096) | B000H - BFFFH |
| Z8F640x | 64 (65,536) | 128 | 0000H - FFFFH | 8 (8192) | E000H - FFFFH |

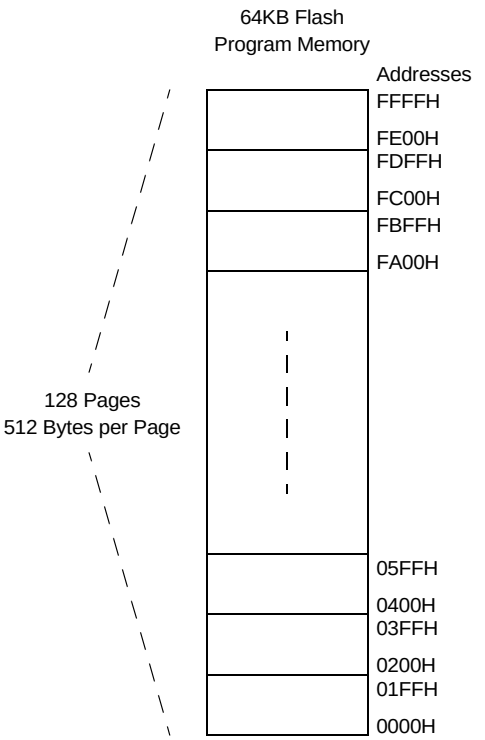



**Figure 84. Flash Memory Arrangement**

## Operation

The Flash Controller programs and erases the Flash memory. The Flash Controller provides the proper Flash controls and timing for byte programming, Page Erase, and Mass Erase of the Flash memory. The Flash Controller contains a protection mechanism, via the Flash Control register (FCTL) to prevent accidental programming or erasure. The Flow Chart in Figure 85 illustrates basic Flash Controller operation. The following subsections provide details on the various operations (Lock, Unlock, Byte Programming, Page Erase, and Mass Erase) listed in Figure 85.

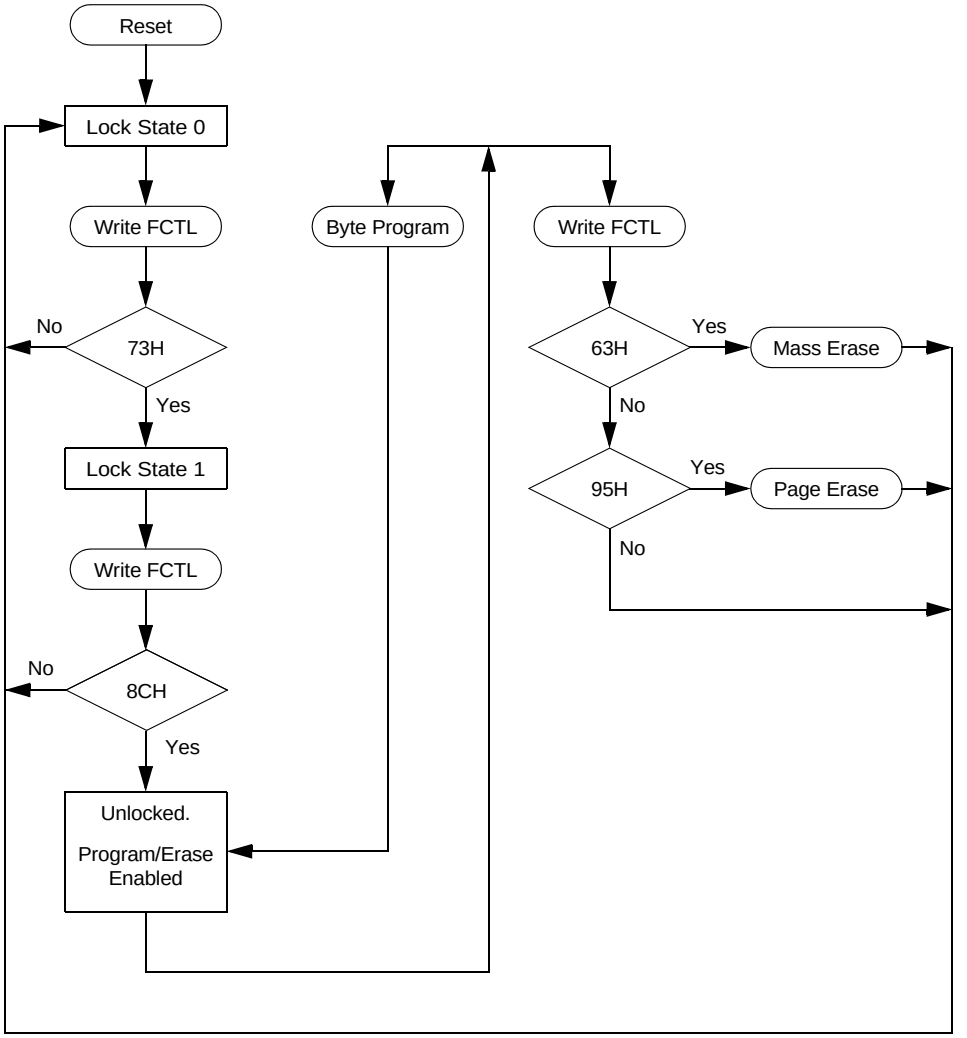



**Figure 85. Flash Controller Operation Flow Chart**

## Flash Operation Timing Using the Flash Frequency Registers

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

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

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

**Caution:** Flash programming and erasure are not supported for system clock frequencies below 32KHz (32768Hz) or above 20MHz. The Flash Frequency High and Low Byte registers must be loaded with the correct value to insure proper operation of the Z8F640x family device.

## Flash Code Protection Against External Access

The user code contained within the Z8F640x family device's Flash memory can be protected against external access via the On-Chip Debugger. Programming the RP Option Bit prevents reading of the user code through the On-Chip Debugger. Refer to the **Option Bits** chapter and the **On-Chip Debugger** chapter for more information.

## Flash Code Protection Against Accidental Program and Erasure

The Z8F640x family device provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by a combination of the Option bits and the locking mechanism of the Flash Controller.

### Flash Code Protection Using the Option Bits

The FHSWP and FWP Option Bits combine to provide three levels of Flash Program Memory protection as listed in Table 84. Refer to the **Option Bits** chapter for more information.

**Table 84. Flash Code Protection Using the Option Bits**

| FHSWP | FWP | Flash Code Protection Description |
|---|---|---|
| 0 | 0 | Programming and erasure 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 | 0 | Programming and Page Erase are enabled for the High Sector of the Flash Program Memory only. The High Sector on the Z8F640x family device contains 1KB to 4KB of Flash with addresses at the top of the available Flash memory. Programming and Page Erase are disabled for the other portions of the Flash Program Memory. Mass erase through user code is disabled. Mass Erase is available through the On-Chip Debugger. |
| 0 or 1 | 1 | Programming, Page Erase, and Mass Erase are enabled for all of Flash Program Memory. |

### Flash Code Protection Using the Flash Controller

At Reset, the Flash Controller locks to prevent accidental program or erasure of the Flash memory. To program or erase the Flash memory, unlock the Flash Controller by making two consecutive writes to the Flash Control register with the values 73H and 8CH, sequentially. After unlocking the Flash Controller, the Flash can be programmed or erased. When the Flash Controller is unlocked, any value written to the Flash Control register locks the Flash Controller. Writing the Mass Erase or Page Erase commands executes the function before locking the Flash Controller.

## Byte Programming

When the Flash Controller is unlocked, all writes to Program Memory program a byte into the Flash. An erased Flash byte contains all 1's (FFH). The programming operation can only be used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from 0 to 1 requires execution of either the Page Erase or Mass Erase commands.

Byte Programming can be accomplished using the On-Chip Debugger's Write Memory command or eZ8 CPU execution of the LDC or LDCI instructions. Refer to the **eZ8 CPU User Manual** for a description of the LDC and LDCI instructions. While the Flash Controller programs the Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. To exit programming mode and lock the Flash, write any value to the Flash Control register, except the Mass Erase or Page Erase commands.

**Caution:** The byte at each address of the Flash memory cannot be programmed (any bits written to 0) more than twice before an erase cycle occurs.

### Page Erase

The Flash memory can be erased one page (512 bytes) at a time. Page Erasing the Flash memory sets all bytes in that page to the value FFH. The Flash Page Select register identifies the page to be erased. With the Flash Controller unlocked, writing the value 95H to the Flash Control register initiates the Page Erase operation. While the Flash Controller executes the Page Erase operation, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. The eZ8 CPU resumes operation after the Page Erase operation completes. If the Page Erase operation is performed through the On-Chip Debugger, poll the Flash Status register to determine when the Page Erase operation is complete. When the Page Erase is complete, the Flash Controller returns to its locked state.

### Mass Erase

The Flash memory can also be Mass Erased using the Flash Controller. Mass Erasing the 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. Typically, the Flash Memory is Mass Erased using the On-Chip Debugger. Via the On-Chip Debugger, poll the Flash Status register to determine when the Mass Erase operation is complete. Although the Flash can be Mass Erased by user program code, when the Mass Erase is complete the user program code is completely erased. When the Mass Erase is complete, the Flash Controller returns to its locked state.

### Flash Controller Bypass

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

Row programing is recommended for gang programming applications and large volume customers who do not require in-circuit initial programming of the Flash memory. Mass Erase and Page Erase operations are also supported when the Flash Controller is bypassed.

Please refer to the document entitled *Third-Party Flash Programming Support for Z8 Encore!*™ for more information on bypassing the Flash Controller. This document is available for download at www.zilog.com.

## Flash Control Register Definitions

### Flash Control Register

The Flash Controller must be unlocked via the Flash Control register before programming or erasing the Flash memory. Writing the sequence 73H 8CH, sequentially, to the Flash Control register unlocks the Flash Controller. When the Flash Controller is unlocked, writing to the Flash Control register can initiate either Page Erase or Mass Erase of the Flash memory. 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 85. Flash Control Register (FCTL)**

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

FCMD—Flash Command
73H = First unlock command.
8CH = Second unlock command.
95H = Page erase command (must be third command in sequence to initiate Page Erase).
63H = Mass erase command (must be third command in sequence to initiate Mass Erase).

## Flash Status Register

The Flash Status register 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 86. Flash Status Register (FSTAT)**

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

Reserved
These bits are reserved and must be 0.

FSTAT—Flash Controller Status
000000 = Flash Controller locked.
000001 = First unlock command received.
000010 = Flash Controller unlocked (second unlock command received).
001xxx = Program operation in progress.
010xxx = Page erase operation in progress.
100xxx = Mass erase operation in progress.

## Flash Page Select Register

The Flash Page Select register is used to select one of the 128 available Flash memory pages to be erased in a Page Erase operation. Each Flash Page contains 512 bytes of Flash memory. During a Page Erase operation, all Flash memory having addresses with the most significant 7-bits given by FPS[6:0] are erased (all bytes written to FFH).

**Table 87. Flash Page Select Register (FPS)**

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

Reserved
This bit is reserved and must be 0.

PAGE—Page Select
This 7-bit field identifies the Flash memory page for Page Erase operation.
Program Memory Address[15:9] = PAGE[6:0]

## Flash Frequency High and Low Byte Registers

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

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

**Caution:** Flash programming and erasure is not supported for system clock frequencies below 32KHz (32768Hz) or above 20MHz. The Flash Frequency High and Low Byte registers must be loaded with the correct value to insure proper operation of the Z8F640x family device.

**Table 88. Flash Frequency High Byte Register (FFREQH)**

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

FFREQH—Flash Frequency High Byte
High byte of the 16-bit Flash Frequency value.

**Table 89. Flash Frequency Low Byte Register (FFREQL)**

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

FFREQL—Flash Frequency Low Byte
Low byte of the 16-bit Flash Frequency value.

# *Option Bits*

## Overview

Option Bits allow user configuration of certain aspects of Z8F640x family device operation. The feature configuration data is stored in the Program Memory and read during Reset. The features available for control via the Option Bits are:

- Watch-Dog Timer time-out response selection–interrupt or Short Reset.
- Watch-Dog 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.

## Operation

### Option Bit Configuration By Reset

Each time the Option Bits are programmed or erased, the Z8F640x family device must be Reset for the change to take place. During any reset operation (System Reset, Short Reset, or Stop Mode Recovery), the Option Bits are automatically read from the Program Memory and written to Option Configuration registers. The Option Configuration registers control operation of the Z8F640x family device. Option Bit control of the Z8F640x family device is established before the device exits Reset and the eZ8 CPU begins code execution. The Option Configuration registers are not part of the Register File and are not accessible for read or write access.

## Option Bit Address Space

The first two bytes of Program Memory at addresses 0000H and 0001H are reserved for the user Option Bits. The byte at Program Memory address 0000H is used to configure user options. The byte at Program Memory address 0001H is reserved for future use and must be left in its unprogrammed state.

### Program Memory Address 0000H

**Table 90. Option Bits At Program Memory Address 0000H**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | WDT_RES | WDT_AO | Reserved | | | RP | FHSWP | FWP |
| RESET | U | U | U | U | U | U | U | U |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | Program Memory 0000H | | | | | | | |

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

WDT_RES—Watch-Dog Timer Reset
0 = Watch-Dog Timer time-out generates an interrupt request. Interrupts must be globally enabled for the eZ8 CPU to acknowledge the interrupt request.
1 = Watch-Dog Timer time-out causes a Short Reset. This setting is the default for unprogrammed (erased) Flash.

WDT_AO—Watch-Dog Timer Always On
0 = Watch-Dog Timer is automatically enabled upon application of system power. Watch-Dog Timer can not be disabled.
1 = Watch-Dog Timer is enabled upon execution of the WDT instruction. Once enabled, the Watch-Dog Timer can only be disabled by a Reset or Stop Mode Recovery. This setting is the default for unprogrammed (erased) Flash.

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

RP—Read Protect
0 = User program code is inaccessible. Limited control features are available through the On-Chip Debugger.
1 = User program code is accessible. All On-Chip Debugger commands are enabled. This setting is the default for unprogrammed (erased) Flash.

FHSWP—Flash High Sector Write Protect
FWP—Flash Write Protect
These two Option Bits combine to provide 3 levels of Program Memory protection:

| FHSWP | FWP | Description |
|---|---|---|
| 0 | 0 | Programming and erasure disabled for all of Program Memory. Programming, Page Erase, and Mass Erase via User Code is disabled. Mass Erase is available through the On-Chip Debugger. |
| 1 | 0 | Programming and Page Erase are enabled for the High Sector of the Program Memory only. The High Sector on the Z8F640x family device contains 1KB to 4KB of Flash with addresses at the top of the available Flash memory. Programming and Page Erase are disabled for the other portions of the Program Memory. Mass erase through user code is disabled. Mass Erase is available through the On-Chip Debugger. |
| 0 or 1 | 1 | Programming, Page Erase, and Mass Erase are enabled for all of Program Memory. |

## Program Memory Address 0001H

**Table 91. Options Bits at Program Memory Address 0001H**

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

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

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

# *On-Chip Debugger*

## Overview

The Z8F640x family devices have 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 and Watchpoints
- Execution of eZ8 CPU instructions.

## Architecture

The On-Chip Debugger consists of four primary functional blocks: transmitter, receiver, auto-baud generator, and debug controller. Figure 86 illustrates the architecture of the On-Chip Debugger

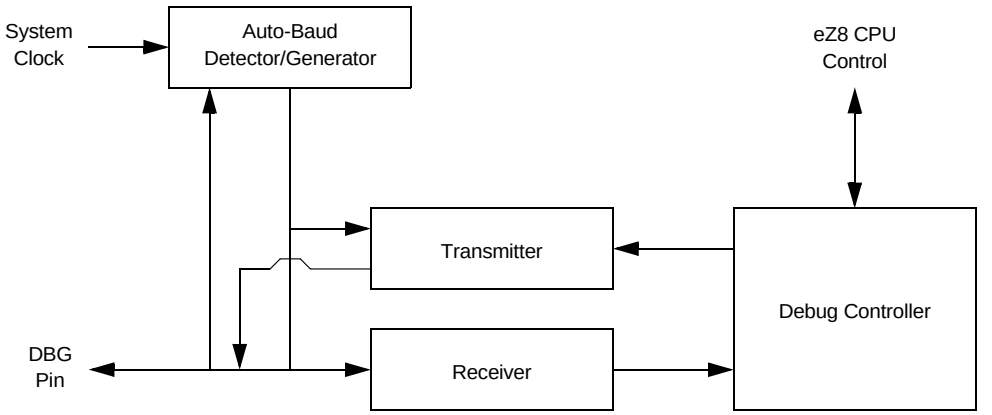



**Figure 86. On-Chip Debugger Block Diagram**

# Operation

## OCD Interface

The On-Chip Debugger uses the DBG pin for communication with an external host. This one-pin interface is a bi-directional open-drain interface that transmits and receives data. Data transmission is half-duplex, in that transmit and receive cannot occur simultaneously. The serial data on the DBG pin is sent using the standard asynchronous data format defined in RS-232. This pin can interface the Z8F640x family device to the serial port of a host PC using minimal external hardware.Two different methods for connecting the DBG pin to an RS-232 interface are depicted in Figures 87 and 88.

**Caution:** For operation of the On-Chip Debugger, ***all*** power pins (VDD and AVDD) must be supplied with power, and ***all*** ground pins (VSS and AVSS) must be properly grounded.
The DBG pin is open-drain and must always be connected to $V_{DD}$ through an external pull-up resistor to ensure proper operation.

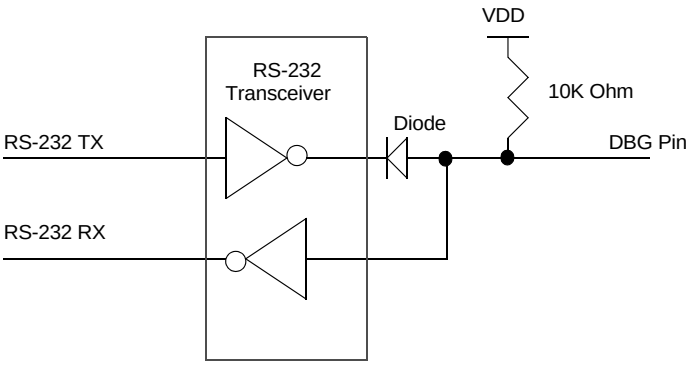



**Figure 87. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (1)**

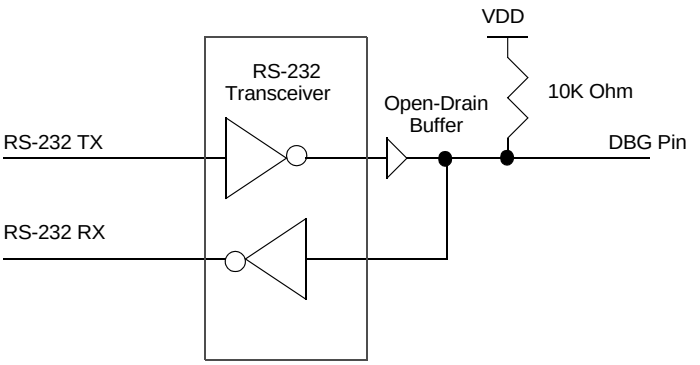



**Figure 88. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (2)**

## Debug Mode

The operating characteristics of the Z8F640x family devices in Debug mode are:

- The eZ8 CPU fetch unit stops, idling the eZ8 CPU, unless directed by the OCD to execute specific instructions.
- The system clock operates unless in Stop mode.
- All enabled on-chip peripherals operate unless in Stop mode.
- Automatically exits Halt mode.
- Constantly refreshes the Watch-Dog Timer, if enabled.

### Entering Debug Mode

The Z8F640x family device enters Debug mode following any of the following operations:

- Writing the DBGMODE bit in the OCD Control Register to 1 using the OCD interface.
- eZ8 CPU execution of a BRK (Breakpoint) instruction (when enabled).
- Break upon a Watchpoint match.
- If the DBG pin is Low when the Z8F640x family device exits Reset, the On-Chip Debugger automatically puts the device into Debug mode.

### Exiting Debug Mode

The device exits Debug mode following any of the following operations:

- Clearing the DBGMODE bit in the OCD Control Register to 0.

- Power-on reset
- Voltage Brownout reset
- Asserting the $\overline{\text{RESET}}$ pin Low to initiate a Reset.
- Driving the DBG pin Low while the Z8F640x family device is in Stop mode initiates a System Reset.

## OCD Data Format

The OCD interface uses the asynchronous data format defined for RS-232. Each character is transmitted as 1 Start bit, 8 data bits (least-significant bit first), and 1.5 Stop bits (Figure 89)

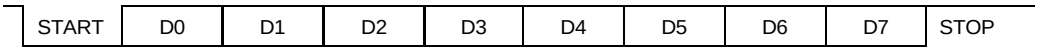



**Figure 89. OCD Data Format**

## OCD Auto-Baud Detector/Generator

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

The Auto-Baud Detector/Generator is clocked by the Z8F640x family device system clock. The minimum baud rate is the system clock frequency divided by 512. For optimal operation, the maximum recommended baud rate is the system clock frequency divided by 8. The theoretical maximum baud rate is the system clock frequency divided by 4. This theoretical maximum is possible for low noise designs with clean signals. Table 92 lists minimum and recommended maximum baud rates for sample crystal frequencies.

**Table 92. OCD Baud-Rate Limits**

| System Clock Frequency (MHz) | Recommended Maximum Baud Rate (kbits/s) | Minimum Baud Rate (kbits/s) |
|---|---|---|
| 20.0 | 2500 | 39.1 |
| 1.0 | 125.0 | 1.96 |
| 0.032768 (32KHz) | 4.096 | 0.064 |

If the OCD receives a Serial Break (nine or more continuous bits Low) the Auto-Baud Detector/Generator resets. The Auto-Baud Detector/Generator can then be reconfigured by sending 80H.

## OCD Serial Errors

The On-Chip Debugger can detect any of the following error conditions on the DBG pin:

- Serial Break (a minimum of nine continuous bits Low)
- Framing Error (received Stop bit is Low)
- Transmit Collision (OCD and host simultaneous transmission detected by the OCD)

When the OCD detects one of these errors, it aborts any command currently in progress, transmits a four character long Serial Break back to the host, and resets the Auto-Baud Detector/Generator. A Framing Error or Transmit Collision may 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 break back to the host only extends the length of the Serial Break if the host releases the Serial Break early.

The host should transmit a Serial Break on the DBG pin when first connecting to the Z8F640x family device or when recovering from an error. A Serial Break from the host resets the Auto-Baud Generator/Detector but does not reset the OCD Control register. A Serial Break leaves the Z8F640x family device in Debug mode if that is the current mode. The OCD is held in Reset until the end of the Serial Break when the DBG pin returns High. Because of the open-drain nature of the DBG pin, the host can send a Serial Break to the OCD even if the OCD is transmitting a character.

## Breakpoints

Execution Breakpoints are generated using the BRK instruction (opcode 00H). When the eZ8 CPU decodes a BRK instruction, it signals the On-Chip Debugger. If Breakpoints are enabled, the OCD enters Debug mode and idles the eZ8 CPU. If Breakpoints are not enabled, the OCD ignores the BRK signal and the BRK instruction operates as an NOP.

### Breakpoints in Flash Memory

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

## Watchpoints

The On-Chip Debugger can set one Watchpoint to cause a Debug Break. The Watchpoint identifies a single Register File address. The Watchpoint can be set to break on reads and/or writes of the selected Register File address. Additionally, the Watchpoint can be configured to break only when a specific data value is read and/or written from the specified reg-

ister. When the Watchpoint event occurs, the Z8F640x family device enters Debug mode and the DBGMODE bit in the OCDCTL register becomes 1.

### Runtime Counter

The On-Chip Debugger contains a 16-bit Runtime Counter. It counts system clock cycles between Breakpoints. The counter starts counting when the On-Chip Debugger leaves Debug mode and stops counting when it enters Debug mode again or when it reaches the maximum count of FFFFH.

## On-Chip Debugger Commands

The host communicates to the On-Chip Debugger by sending OCD commands using the DBG interface. During normal operation of the Z8F640x family device, only a subset of the OCD commands are available. In Debug mode, all OCD commands become available unless the user code and control registers are protected by programming the Read Protect Option Bit (RP). The Read Protect Option Bit prevents the code in memory from being read out of the Z8F640x family device. When this option is enabled, several of the OCD commands are disabled. Table 93 contains a summary of the On-Chip Debugger commands. Each OCD command is described in further detail in the bulleted list following Table 93. Table 93 indicates those commands that operate when the Z8F640x family device is not in Debug mode (normal operation) and those commands that are disabled by programming the Read Protect Option Bit.

**Table 93. On-Chip Debugger Commands**

| Debug Command | Command Byte | Enabled when NOT in Debug mode? | Disabled by Read Protect Option Bit |
|---|---|---|---|
| Read OCD Revision | 00H | Yes | - |
| Reserved | 01H | - | - |
| Read OCD Status Register | 02H | Yes | - |
| Read Runtime Counter | 03H | - | - |
| Write OCD Control Register | 04H | Yes | Cannot clear DBGMODE bit |
| Read OCD Control Register | 05H | Yes | - |
| Write Program Counter | 06H | - | Disabled |
| Read Program Counter | 07H | - | Disabled |
| Write Register | 08H | - | Only writes of the Flash Memory Control registers are allowed. Additionally, only the Mass Erase command is allowed to be written to the Flash Control register. |
| Read Register | 09H | - | Disabled |

**Table 93. On-Chip Debugger Commands**

| Debug Command | Command Byte | Enabled when NOT in Debug mode? | Disabled by Read Protect Option Bit |
|---|---|---|---|
| Write Program Memory | 0AH | - | Disabled |
| Read Program Memory | 0BH | - | Disabled |
| Write Data Memory | 0CH | - | Yes |
| Read Data Memory | 0DH | - | - |
| Read Program Memory CRC | 0EH | - | - |
| Reserved | 0FH | - | - |
| Step Instruction | 10H | - | Disabled |
| Stuff Instruction | 11H | - | Disabled |
| Execute Instruction | 12H | - | Disabled |
| Reserved | 13H - 1FH | - | - |
| Write Watchpoint | 20H | - | Disabled |
| Read Watchpoint | 21H | - | - |
| Reserved | 22H - FFH | - | - |

In the following bulleted list of OCD Commands, data and commands sent from the host to the On-Chip Debugger are identified by '`DBG <-- Command/Data`'. Data sent from the On-Chip Debugger back to the host is identified by '`DBG --> Data`'

- **Read OCD Revision (00H)**—The Read OCD Revision command is used to determine the version of the On-Chip Debugger. If OCD commands are added, removed, or changed, this revision number changes.

```
DBG <-- 00H
DBG --> OCDREV[15:8] (Major revision number)
DBG --> OCDREV[7:0] (Minor revision number)
```

- **Read OCD Status Register (02H)**—The Read OCD Status Register command is used to read the OCDSTAT register.

```
DBG <-- 02H
DBG --> OCDSTAT[7:0]
```

- **Read Runtime Counter (03H)**—The Runtime Counter is used to count Z8 Encore! system clock cycles in between Breakpoints. The 16-bit Runtime Counter counts up from `0000H` and stops at the maximum count of `FFFFH`. The Runtime Counter is overwritten during the Write Memory, Read Memory, Write Register, Read Register, Read Memory CRC, Step Instruction, Stuff Instruction, and Execute Instruction commands.

```
DBG <-- 03H
DBG --> RuntimeCounter[15:8]
DBG --> RuntimeCounter[7:0]
```

- **Write OCD Control Register (04H)**—The Write OCD Control Register command writes the data that follows to the OCDCTL register. When the Read Protect Option Bit is enabled, the DBGMODE bit (OCDCTL[7]) can only be set to 1, it cannot be cleared to 0 and the only method of putting the Z8F640x family device back into normal operating mode is to reset the device.

```
DBG <-- 04H
DBG <-- OCDCTL[7:0]
```

- **Read OCD Control Register (05H)**—The Read OCD Control Register command reads the value of the OCDCTL register.

```
DBG <-- 05H
DBG --> OCDCTL[7:0]
```

- **Write Program Counter (06H)**—The Write Program Counter command writes the data that follows to the eZ8 CPU's Program Counter (PC). If the Z8F640x family 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 Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, this command returns FFFFH.

```
DBG <-- 07H
DBG --> ProgramCounter[15:8]
DBG --> ProgramCounter[7:0]
```

- **Write Register (08H)**—The Write Register command writes data to the Register File. Data can be written 1-256 bytes at a time (256 bytes can be written by setting size to zero). If the Z8F640x family device is not in Debug mode, the address and data values are discarded. If the Read Protect Option Bit is enabled, then only writes to the Flash Control Registers are allowed and all other register write data values are discarded.

```
DBG <-- 08H
DBG <-- {4'h0,Register Address[11:8]}
DBG <-- Register Address[7:0]
DBG <-- Size[7:0]
DBG <-- 1-256 data bytes
```

- **Read Register (09H)**—The Read Register command reads data from the Register File. Data can be read 1-256 bytes at a time (256 bytes can be read by setting size to

zero). If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, this command returns FFH for all the data values.

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

- **Write Program Memory (0AH)**—The Write Program Memory command writes data to Program Memory. This command is equivalent to the LDC and LDCI instructions. Data can be written 1-65536 bytes at a time (65536 bytes can be written by setting size to zero). The on-chip Flash Controller must be written to and unlocked for the programming operation to occur. If the Flash Controller is not unlocked, the data is discarded. If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, the data is discarded.

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

- **Read Program Memory (0BH)**—The Read Program Memory command reads data from Program Memory. This command is equivalent to the LDC and LDCI instructions. Data can be read 1-65536 bytes at a time (65536 bytes can be read by setting size to zero). If the Z8F640x family 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 can be written 1-65536 bytes at a time (65536 bytes can be written by setting size to zero). If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, the data is discarded.

```
DBG <-- 0CH
DBG <-- Data Memory Address[15:8]
DBG <-- Data Memory Address[7:0]
DBG <-- Size[15:8]
DBG <-- Size[7:0]
DBG <-- 1-65536 data bytes
```

- **Read Data Memory (0DH)**—The Read Data Memory command reads from Data Memory. This command is equivalent to the LDE and LDEI instructions. Data can be read 1-65536 bytes at a time (65536 bytes can be read by setting size to zero). If the Z8F640x family device is not in Debug mode, this command returns FFH for the data.

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

- **Read Program Memory CRC (0EH)**—The Read Program Memory CRC command computes and returns the CRC (cyclic redundancy check) of Program Memory using the 16-bit CRC-CCITT polynomial. If the Z8F640x family device is not in Debug mode, this command returns FFFFH for the CRC value. Unlike most other OCD Read commands, there is a delay from issuing of the command until the OCD returns the data. The OCD reads the Program Memory, calculates the CRC value, and returns the result. The delay is a function of the Program Memory size and is approximately equal to the system clock period multiplied by the number of bytes in the Program Memory.

```
DBG <-- 0EH
DBG --> CRC[15:8]
DBG --> CRC[7:0]
```

- **Step Instruction (10H)**—The Step Instruction command steps one assembly instruction at the current Program Counter (PC) location. If the Z8F640x family 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 Z8F640x family device is not in Debug mode or the Read Protect Option Bit is enabled, the OCD ignores this command.

```
DBG <-- 11H
DBG <-- opcode[7:0]
```

- **Execute Instruction (12H)**—The Execute Instruction command allows sending an entire instruction to be executed to the eZ8 CPU. This command can also step over Breakpoints. The number of bytes to send for the instruction depends on the opcode. If the Z8F640x family device is not in Debug mode or the Read Protect Option Bit is enabled, this command reads and discards one byte.

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

- **Write Watchpoint (20H)**—The Write Watchpoint command sets and configures the debug Watchpoint. If the Z8F640x family device is not in Debug mode or the Read Protect Option Bit is enabled, the WPTCTL bits are all set to zero.

```
DBG <-- 20H
DBG <-- WPTCTL[7:0]
DBG <-- WPTADDR[7:0]
DBG <-- WPTDATA[7:0]
```

- **Read Watchpoint (21H)**—The Read Watchpoint command reads the current Watchpoint registers.

```
DBG <-- 21H
DBG --> WPTCTL[7:0]
DBG --> WPTADDR[7:0]
DBG --> WPTDATA[7:0]
```

## On-Chip Debugger Control Register Definitions

### OCD Control Register

The OCD Control register controls the state of the On-Chip Debugger. This register enters or exits Debug mode and enables the BRK instruction. It can also reset the Z8F640x family device.

A “reset and stop” function can be achieved by writing 81H to this register. A “reset and go” function can be achieved by writing 41H to this register. If the Z8F640x family device is in Debug mode, a “run” function can be implemented by writing 40H to this register.

**Table 94. OCD Control Register (OCDCTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DBGMODE | BRKEN | DBGACK | Reserved | | | | RST |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R | R | R | R/W |

DBGMODE—Debug Mode
Setting this bit to 1 causes the Z8F640x family device to enter Debug mode. When in Debug mode, the eZ8 CPU stops fetching new instructions. Clearing this bit causes the eZ8 CPU to start running again. This bit is automatically set when a BRK instruction is decoded and Breakpoints are enabled or when a Watchpoint Debug Break is detected. If the Read Protect Option Bit is enabled, this bit can only be cleared by resetting the Z8F640x family device, it cannot be written to 0.
0 = The Z8F640x family device is operating in normal mode.
1 = The Z8F640x family device is in Debug mode.

BRKEN—Breakpoint Enable
This bit controls the behavior of the BRK instruction (opcode 00H). By default, Breakpoints are disabled and the BRK instruction behaves like a NOP. If this bit is set to 1, when a BRK instruction is decoded, the DBGMODE bit of the OCDCTL register is automatically set to one.
0 = Breakpoints are disabled.
1 = Breakpoints are enabled.

DBGACK—Debug Acknowledge
This bit enables the debug acknowledge feature. If this bit is set to 1, then the OCD sends an Debug Acknowledge character (FFH) to the host when a Breakpoint or Watchpoint occurs.
0 = Debug Acknowledge is disabled.
1 = Debug Acknowledge is enabled.

Reserved
These bits are reserved and must be 0.

RST—Reset
Setting this bit to 1 resets the Z8F640x family device. The device goes through a normal Power-On Reset sequence with the exception that the On-Chip Debugger is not reset. This bit is automatically cleared to 0 when the reset finishes.
0 = No effect.
1 = Reset Z8F640x family device.

## OCD Status Register

The OCD Status register reports status information about the current state of the debugger and the Z8F640x family device.

**Table 95. OCD Status Register (OCDSTAT)**

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

DBG—Debug Status
0 = The Z8F640x family device is operating in normal mode.
1 = The Z8F640x family device is in Debug mode.

HALT—Halt Mode
0 = The Z8F640x family device is not in Halt mode.
1 = The Z8F640x family device is in Halt mode.

RPEN—Read Protect Option Bit Enabled
0 = The Read Protect Option Bit is disabled (1).
0 = The Read Protect Option Bit is enabled (0), disabling many OCD commands.

Reserved
These bits are always 0.

## OCD Watchpoint Control Register

The OCD Watchpoint Control register is used to configure the debug Watchpoint.

**Table 96. OCD Watchpoint Control/Address (WPTCTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | WPW | WPR | WPDM | Reserved | WPTADDR[11:8] | | | |
| 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 |

WPW—Watchpoint Break on Write
This bit cannot be set if the Read Protect Option Bit is enabled.
0 = Watchpoint Break on Register File write is disabled.
1 = Watchpoint Break on Register File write is enabled.

WPR—Watchpoint Break on Read
This bit cannot be set if the Read Protect Option Bit is enabled.
0 = Watchpoint Break on Register File read is disabled.
1 = Watchpoint Break on Register File write is enabled.

WPDM—Watchpoint Data Match
If this bit is set, then the Watchpoint only generates a Debug Break if the data being read or written matches the specified Watchpoint data. Either the WPR and/or WPW bits must also be set for this bit to affect operation. This bit cannot be set if the Read Protect Option Bit is enabled.
0 = Watchpoint Break on read and/or write does not require a data match.
1 = Watchpoint Break on read and/or write requires a data match.

Reserved
This bit is reserved and must be 0.

RADDR[11:8]—Register address
These bits specify the upper 4 bits of the Register File address to match when generating a Watchpoint Debug Break. The full 12-bit Register File address is given by {WPTCTL3:0], WPTADDR[7:0]}.

## OCD Watchpoint Address Register

The OCD Watchpoint Address register specifies the lower 8 bits of the Register File address bus to match when generating Watchpoint Debug Breaks. The full 12-bit Register File address is given by {WPTCTL3:0], WPTADDR[7:0]}.

**Table 97. OCD Watchpoint Address (WPTADDR)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | WPTADDR[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 |

WPTADDR[7:0]—Watchpoint Register File Address
These bits specify the lower eight bits of the register address to match when generating a Watchpoint Debug Break.

## OCD Watchpoint Data Register

The OCD Watchpoint Data register specifies the data to match if Watchpoint data match is enabled.

**Table 98. OCD Watchpoint Data (WPTDATA)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | WPTDATA[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 |

WPTDATA[7:0]—Watchpoint Register File Data
These bits specify the Register File data to match when generating Watchpoint Debug Breaks with the WPDM bit (WPTCTL[5]) is set to 1.

# On-Chip Oscillator

The Z8F640x family devices feature an on-chip oscillator for use with an external 1-20MHz crystal. This oscillator generates the primary system clock for the internal eZ8 CPU and the majority of the on-chip peripherals. Alternatively, the $X_{IN}$ input pin can also accept a CMOS-level clock input signal (32kHz-20MHz). If an external clock generator is used, the $X_{OUT}$ pin must be left unconnected. The Z8F640x family device does ***not*** contain in internal clock divider. The frequency of the signal on the $X_{IN}$ input pin determines the frequency of the system clock. The Z8F640x family device on-chip oscillator does not support external RC networks or ceramic resonators.

## 20MHz Crystal Oscillator Operation

Figure 90 illustrates a recommended configuration for connection with an external 20MHz, fundamental-mode, parallel-resonant crystal. Recommended crystal specifications are provided in Table 99. Resistor $R_1$ limits total power dissipation by the crystal. Printed circuit board layout should 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.

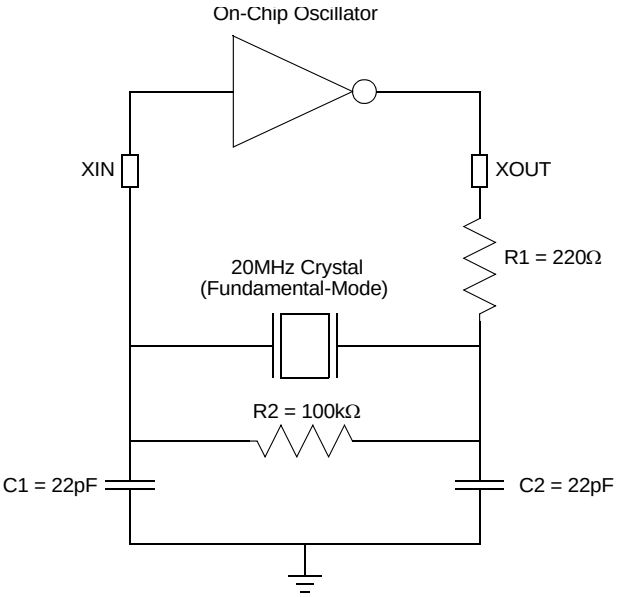



**Figure 90. Recommended Crystal Oscillator Configuration (20MHz operation)**

**Table 99. Recommended Crystal Oscillator Specifications (20MHz Operation)**

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

# Electrical Characteristics

## Absolute Maximum Ratings

Stresses greater than those listed in Table 100 may cause permanent damage to the device. These ratings are stress ratings only. Operation of the device at any condition outside those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. For improved reliability, unused inputs must be tied to one of the supply voltages ($V_{DD}$ or $V_{SS}$).

**Table 100. Absolute Maximum Ratings**

| Parameter | Minimum | Maximum | Units | Notes |
|---|---|---|---|---|
| Ambient temperature under bias | -40 | +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 | |
| **80-Pin QFP Maximum Ratings at -40°C to 70°C** | | | | |
| Total power dissipation | | 550 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 150 | mA | |
| **80-Pin QFP Maximum Ratings at 70°C to 105°C** | | | | |
| Total power dissipation | | 200 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 56 | mA | |
| **68-Pin PLCC 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 | |

Notes:
1. This voltage applies to all pins except the following: $V_{DD}$, $AV_{DD}$, pins supporting analog input (Port B and Port H), $\overline{\text{RESET}}$, and where noted otherwise.

**Table 100. Absolute Maximum Ratings**

| Parameter | Minimum | Maximum | Units | Notes |
|---|---|---|---|---|
| **68-Pin PLCC Maximum Ratings at $70^0$C to $105^0$C** | | | | |
| Total power dissipation | | 500 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 140 | mA | |
| **64-Pin LQFP 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 | |
| **64-Pin LQFP Maximum Ratings at $70^0$C to $105^0$C** | | | | |
| Total power dissipation | | 540 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 150 | mA | |
| **44-Pin PLCC 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 PLCC Maximum Ratings at $70^0$C to $105^0$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 $70^0$C to $105^0$C** | | | | |
| Total power dissipation | | 410 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 114 | 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:
1. This voltage applies to all pins except the following: $V_{DD}$, $AV_{DD}$, pins supporting analog input (Port B and Port H), $\overline{\text{RESET}}$, and where noted otherwise.

## DC Characteristics

Table 101 lists the DC characteristics of the Z8F640x family devices. All voltages are referenced to $V_{SS}$, the primary system ground.

**Table 101. DC Characteristics**

| Symbol | Parameter | $T_A$ = -40°C to 105°C | | | Units | Conditions |
|---|---|---|---|---|---|---|
| | | Minimum | Typical | Maximum | | |
| $V_{DD}$ | Supply Voltage | 3.0 | – | 3.6 | V | |
| $V_{IL1}$ | Low Level Input Voltage | -0.3 | – | 0.3*$V_{DD}$ | V | For all input pins except $\overline{\text{RESET}}$, DBG, and XIN. |
| $V_{IL2}$ | Low Level Input Voltage | -0.3 | – | 0.2*$V_{DD}$ | V | For $\overline{\text{RESET}}$, DBG, and XIN. |
| $V_{IH1}$ | High Level Input Voltage | 0.7*$V_{DD}$ | – | 5.5 | V | Port A, C, D, E, F, and G pins. |
| $V_{IH2}$ | High Level Input Voltage | 0.7*$V_{DD}$ | – | $V_{DD}$+0.3 | V | Port B and H pins. |
| $V_{IH3}$ | High Level Input Voltage | 0.8*$V_{DD}$ | – | $V_{DD}$+0.3 | V | $\overline{\text{RESET}}$, DBG, and XIN pins. |
| $V_{OL1}$ | Low Level Output Voltage | – | – | 0.4 | V | $V_{DD}$ = 3.0V; $I_{OL}$ = 2mA<br>High Output Drive disabled. |
| $V_{OH1}$ | High Level Output Voltage | 2.4 | – | – | V | $V_{DD}$ = 3.0V; $I_{OH}$ = -2mA<br>High Output Drive disabled. |
| $V_{OL2}$ | Low Level Output Voltage | – | – | 0.6 | V | $V_{DD}$ = 3.3V; $I_{OL}$ = 20mA<br>High Output Drive enabled.<br>$T_A$ = -40°C to +70°C |
| $V_{OL3}$ | Low Level Output Voltage | – | – | 0.6 | V | $V_{DD}$ = 3.3V; $I_{OL}$ = 15mA<br>High Output Drive enabled.<br>$T_A$ = 70°C to +105°C |
| $V_{OH2}$ | High Level Output Voltage | 2.4 | – | – | V | $V_{DD}$ = 3.3V; $I_{OH}$ = -20mA<br>High Output Drive enabled.<br>$T_A$ = -40°C to +70°C |
| $V_{OH3}$ | High Level Output Voltage | 2.4 | – | – | V | $V_{DD}$ = 3.3V; $I_{OH}$ = -15mA<br>High Output Drive enabled.<br>$T_A$ = 70°C to +105°C |
| $I_{IL}$ | Input Leakage Current | -5 | – | +5 | µA | $V_{DD}$ = 3.6V;<br>$V_{IN}$ = VDD or VSS[1] |
| $I_{TL}$ | Tri-State Leakage Current | -5 | – | +5 | µA | $V_{DD}$ = 3.6V |
| $C_{PAD}$ | GPIO Port Pad Capacitance | – | 8.0[2] | – | pF | |
| $C_{XIN}$ | XIN Pad Capacitance | – | 8.0[2] | – | pF | |
| $C_{XOUT}$ | XOUT Pad Capacitance | – | 9.5[2] | – | pF | |

**Table 101. DC Characteristics**

| Symbol | Parameter | $T_A$ = -40°C to 105°C | | | Units | Conditions |
|---|---|---|---|---|---|---|
| | | Minimum | Typical | Maximum | | |
| $I_{PU}$ | Weak Pull-up Current | 30 | 100 | 350 | µA | $V_{DD}$ = 3.0 - 3.6V |
| $I_{CCS}$ | Supply Current in Stop Mode | | 600 | | µA | $V_{DD}$ = 3.3V |

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

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

Figure 91 illustrates the typical current consumption while operating at 25ºC, 3.3V, versus the system clock frequency.

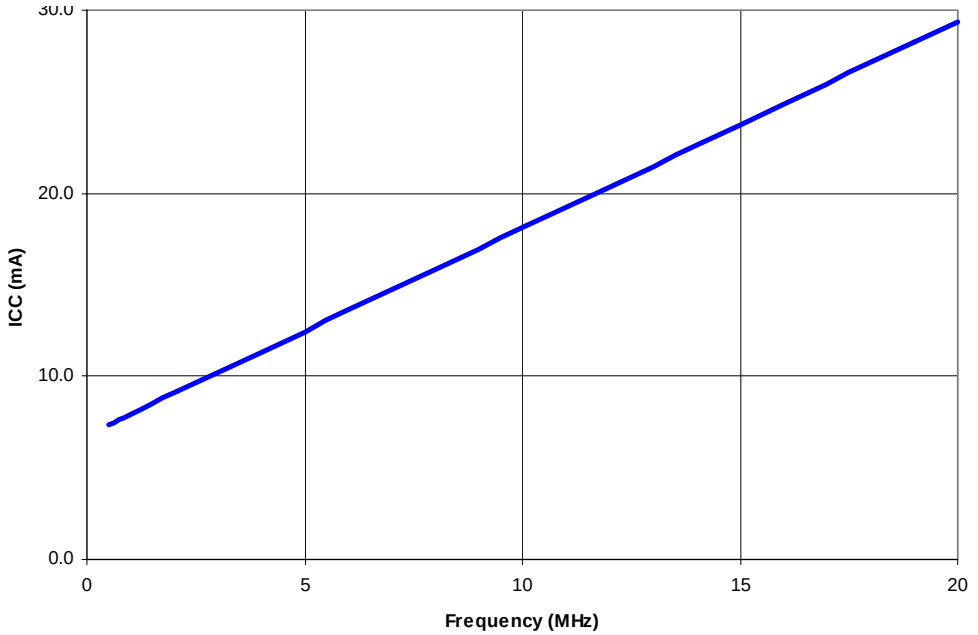



**Figure 91. Nominal ICC Versus System Clock Frequency**

Figure 92 illustrates the typical current consumption in Halt mode while operating at 25ºC, 3.3V, versus the system clock frequency.

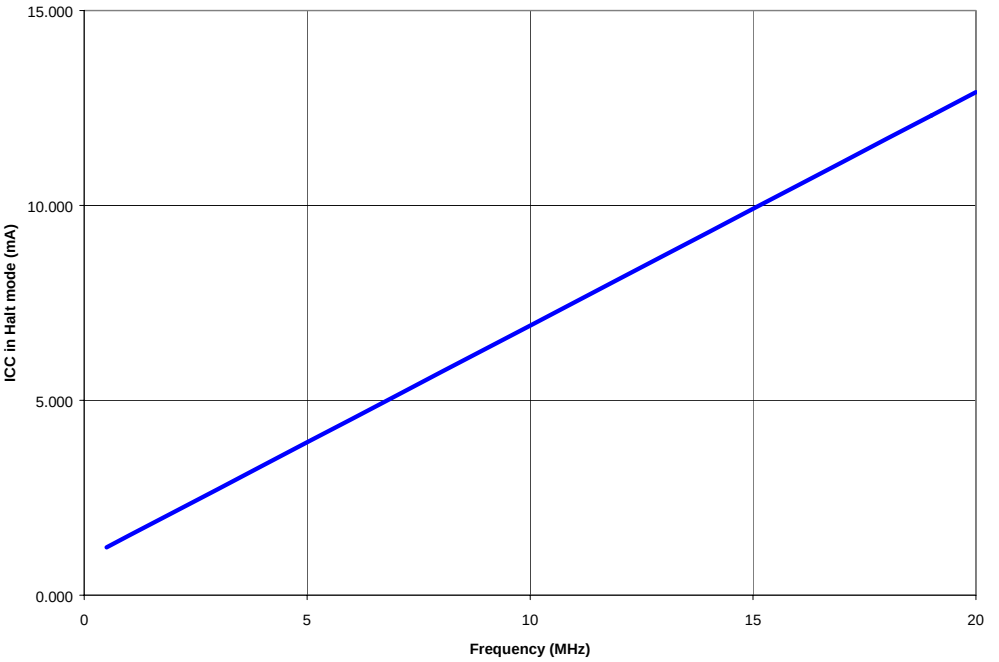



**Figure 92. Nominal Halt Mode ICC Versus System Clock Frequency**

## AC Characteristics

The section provides information on the AC characteristics and timing of the Z8 Encore!™. All AC timing information assumes a standard load of 50pF on all outputs.

**Table 102. AC Characteristics**

| Symbol | Parameter | $V_{DD}$ = 3.0 - 3.6V<br>$T_A$ =-40$^0$C to 105$^0$C | | Units | Conditions |
|---|---|---|---|---|---|
| | | Minimum | Maximum | | |
| $F_{sysclk}$ | System Clock Frequency | – | 20.0 | MHz | Read-only from Flash memory. |
| | | 0.032768 | 20.0 | MHz | Program or erasure of the Flash memory. |
| $F_{XTAL}$ | Crystal Oscillator Frequency | 1.0 | 20.0 | MHz | System clock frequencies below the crystal oscillator minimum require an external clock driver. |
| $T_{XIN}$ | System Clock Period | 50 | – | ns | $T_{CLK} = 1/F_{sysclk}$ |
| $T_{XINH}$ | System Clock High Time | 20 | 30 | ns | $T_{CLK}$ = 50ns |
| $T_{XINL}$ | System Clock Low Time | 20 | 30 | ns | $T_{CLK}$ = 50ns |
| $T_{XINR}$ | System Clock Rise Time | – | 3 | ns | $T_{CLK}$ = 50ns |
| $T_{XINF}$ | System Clock Fall Time | – | 3 | ns | $T_{CLK}$ = 50ns |

## On-Chip Peripheral AC and DC Electrical Characteristics

**Table 103. Power-On Reset and Voltage Brown-Out Electrical Characteristics and Timing**

| Symbol | Parameter | $T_A$ = -40°C to 105°C | | | Units | Conditions |
|---|---|---|---|---|---|---|
| | | Minimum | Typical[1] | Maximum | | |
| $V_{POR}$ | Power-On Reset Voltage Threshold | 2.40 | 2.70 | 2.90 | V | $V_{DD} = V_{POR}$ |
| $V_{VBO}$ | Voltage Brown-Out Reset Voltage Threshold | 2.30 | 2.60 | 2.85 | V | $V_{DD} = V_{VBO}$ |
| | $V_{POR}$ to $V_{VBO}$ hysteresis | 50 | 100 | – | mV | |
| | Starting $V_{DD}$ voltage to ensure valid Power-On Reset. | – | $V_{SS}$ | – | V | |
| 1 Data in the typical column is from characterization at 3.3V and 0°C. These values are provided for design guidance only and are not tested in production. | | | | | | |
| $T_{ANA}$ | Power-On Reset Analog Delay | – | 50 | – | µs | $V_{DD} > V_{POR}$; $T_{POR}$ Digital Reset delay follows $T_{ANA}$ |
| $T_{POR}$ | Power-On Reset Digital Delay | – | 10.2 | – | ms | 512 WDT Oscillator cycles (50KHz) + 70 System Clock cycles (20MHz) |
| $T_{VBO}$ | Voltage Brown-Out Pulse Rejection Period | – | 10 | – | ns | $V_{DD} < V_{VBO}$ to generate a Reset. |
| $T_{RAMP}$ | Time for VDD to transition from $V_{SS}$ to $V_{POR}$ to ensure valid Reset | 0.10 | – | 100 | ms | |

**Table 104. Flash Memory Electrical Characteristics and Timing**

| Parameter | $V_{DD}$ = 3.0 - 3.6V<br>$T_A$ = -40°C to 105°C | | | Units | Notes |
|---|---|---|---|---|---|
| | Minimum | Typical | Maximum | | |
| Flash Byte Read Time | 50 | – | – | ns | |
| Flash Byte Program Time | 20 | – | 40 | µs | |
| Flash Page Erase Time | 10 | – | – | ms | |

**Table 104. Flash Memory Electrical Characteristics and Timing (Continued)**

| Parameter | $V_{DD}$ = 3.0 - 3.6V<br>$T_A$ = -40°C to 105°C<br>Minimum | Typical | Maximum | Units | Notes |
|---|---|---|---|---|---|
| Writes to Single Address Before Next Erase | – | – | 2 | | |
| Flash Row Program Time | – | – | 8 | ms | Cumulative program time for single row cannot exceed limit before next erase. This parameter is only an issue when bypassing the Flash Controller. |
| Data Retention | 100 | – | – | years | 25°C |
| Endurance | 10,000 | – | – | cycles | Program / erase cycles |

**Table 105. Watch-Dog Timer Electrical Characteristics and Timing**

| Symbol | Parameter | $V_{DD}$ = 3.0 - 3.6V<br>$T_A$ = -40°C to 105°C<br>Minimum | Typical | Maximum | Units | Conditions |
|---|---|---|---|---|---|---|
| $F_{WDT}$ | WDT Oscillator Frequency | 25 | 50 | 100 | kHz | |

**Table 106. Analog-to-Digital Converter Electrical Characteristics and Timing**

| Symbol | Parameter | $V_{DD}$ = 3.0 - 3.6V<br>$T_A$ = -40°C to 105°C<br>Minimum | Typical | Maximum | Units | Conditions |
|---|---|---|---|---|---|---|
| | Resolution | – | 10 | – | bits | External $V_{REF}$ = 3.0V;<br>$R_S$ <= 3.0kΩ |
| | Differential Nonlinearity (DNL) | -1.0 | – | 1.0 | LSB | External $V_{REF}$ = 3.0V;<br>$R_S$ <= 3.0kΩ |
| | Integral Nonlinearity (INL) | -3.0 | – | 3.0 | LSB | External $V_{REF}$ = 3.0V;<br>$R_S$ <= 3.0kΩ |
| | DC Offset Error | -35 | – | 25 | mV | 80-pin QFP and 64-pin LQFP packages. |

[1] Analog source impedance affects the ADC offset voltage (because of pin leakage) and input settling time.

**Table 106. Analog-to-Digital Converter Electrical Characteristics and Timing (Continued)**

| Symbol | Parameter | $V_{DD}$ = 3.0 - 3.6V, $T_A$ = -40°C to 105°C | | | Units | Conditions |
|---|---|---|---|---|---|---|
| | | Minimum | Typical | Maximum | | |
| | DC Offset Error | -50 | – | 25 | mV | 40-pin PDIP, 44-pin LQFP, 44-pin PLCC, and 68-pin PLCC packages. |
| $V_{REF}$ | Internal Reference Voltage | – | 2.0 | – | V | |
| | Single-Shot Conversion Time | – | 5129 | – | cycles | System clock cycles |
| | Continuous Conversion Time | – | 256 | – | cycles | System clock cycles |
| | Sampling Rate | System Clock / 256 | | | Hz | |
| | Signal Input Bandwidth | – | – | 3.5 | kHz | |
| $R_S$ | Analog Source Impedance | – | – | 10[1] | kΩ | |
| Zin | Input Impedance | | 150 | | kΩ | 20MHz system clock. Input impedance increases with lower system clock frequency. |
| $V_{REF}$ | External Reference Voltage | | | AVDD | V | AVDD <= VDD. When using an external reference voltage, decoupling capacitance should be placed from VREF to AVSS. |

[1] Analog source impedance affects the ADC offset voltage (because of pin leakage) and input settling time.

## General Purpose I/O Port Input Data Sample Timing

Figure 93 illustrates timing of the GPIO Port input sampling. The input value on a GPIO Port pin is sampled on the rising edge of the system clock. The Port value is then available to the eZ8 CPU on the second rising clock edge following the change of the Port value.

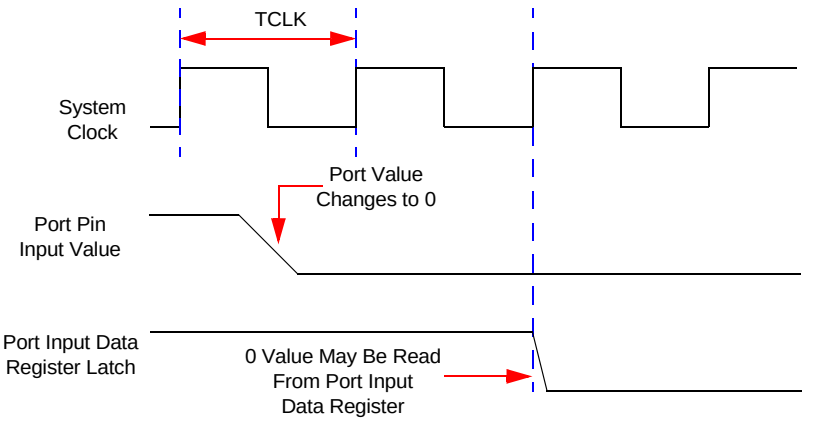



**Figure 93. Port Input Sample Timing**

**Table 107. GPIO Port Input Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_{S_PORT}$ | Port Input Transition to XIN Rise Setup Time (Not pictured) | 5 | – |
| $T_{H_PORT}$ | XIN Rise to Port Input Transition Hold Time (Not pictured) | 5 | – |
| $T_{SMR}$ | GPIO Port Pin Pulse Width to Insure Stop Mode Recovery (for GPIO Port Pins enabled as SMR sources) | 1µs | |

## General Purpose I/O Port Output Timing

Figure 94 and Table 108 provide timing information for GPIO Port pins.

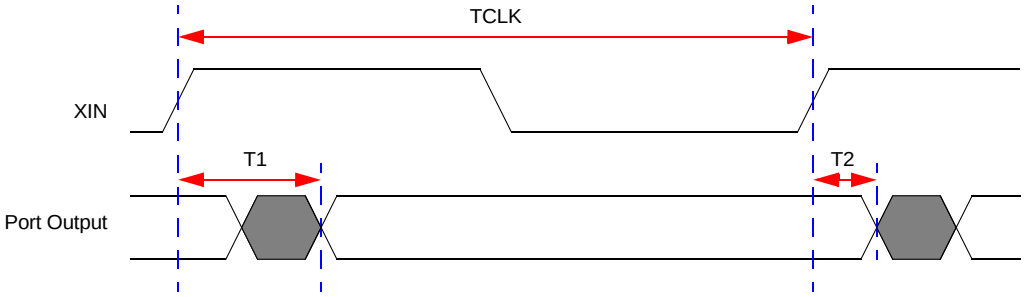



**Figure 94. GPIO Port Output Timing**

**Table 108. GPIO Port Output Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_1$ | XIN Rise to Port Output Valid Delay | – | 15 |
| $T_2$ | XIN Rise to Port Output Hold Time | 2 | – |

## On-Chip Debugger Timing

Figure 95 and Table 109 provide timing information for DBG pins. The timing specifications presume a rise and fall time on DBG of less than 4µs.

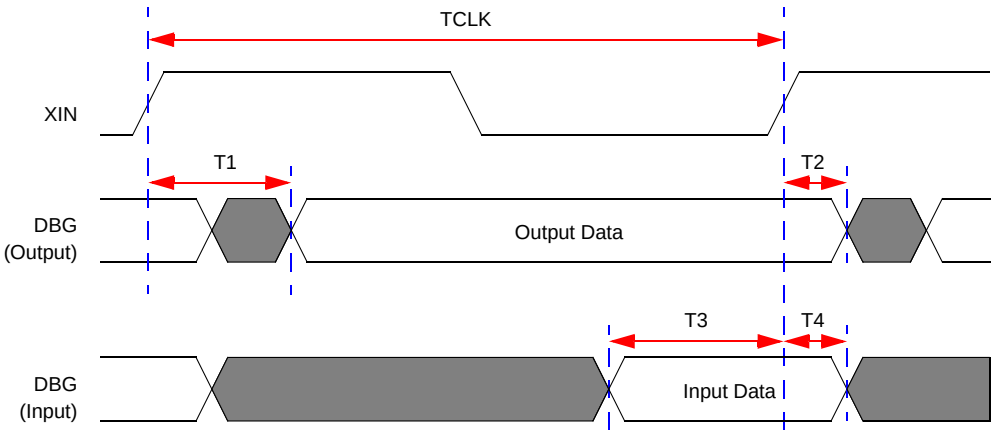



**Figure 95. On-Chip Debugger Timing**

**Table 109. On-Chip Debugger Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| **DBG** | | | |
| $T_1$ | XIN Rise to DBG Valid Delay | – | 15 |
| $T_2$ | XIN Rise to DBG Output Hold Time | 2 | – |
| $T_3$ | DBG to XIN Rise Input Setup Time | 10 | – |
| $T_4$ | DBG to XIN Rise Input Hold Time | 5 | – |
| | DBG frequency | | System Clock / 4 |

## SPI Master Mode Timing

Figure 96 and Table 110 provide timing information for SPI Master mode pins. Timing is shown with SCK rising edge used to source MOSI output data, SCK falling edge used to sample MISO input data. Timing on the SS output pin(s) is controlled by software.

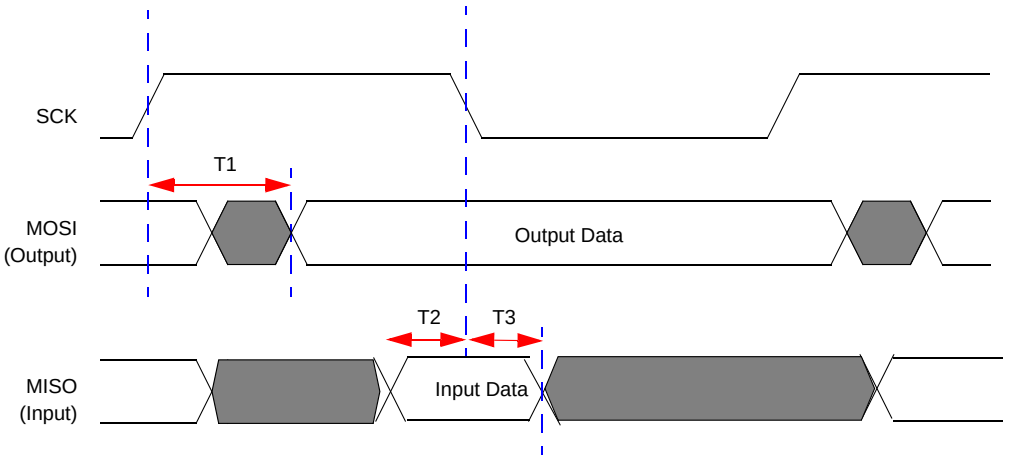



**Figure 96. SPI Master Mode Timing**

**Table 110. SPI Master Mode Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_1$ | SCK Rise to MOSI output Valid Delay | -5 | +5 |
| $T_2$ | MISO input to SCK (receive edge) Setup Time | 20 | |
| $T_3$ | MISO input to SCK (receive edge) Hold Time | 0 | |

## SPI Slave Mode Timing

Figure 97 and Table 111 provide timing information for the SPI slave mode pins. Timing is shown with SCK rising edge used to source MISO output data, SCK falling edge used to sample MOSI input data.

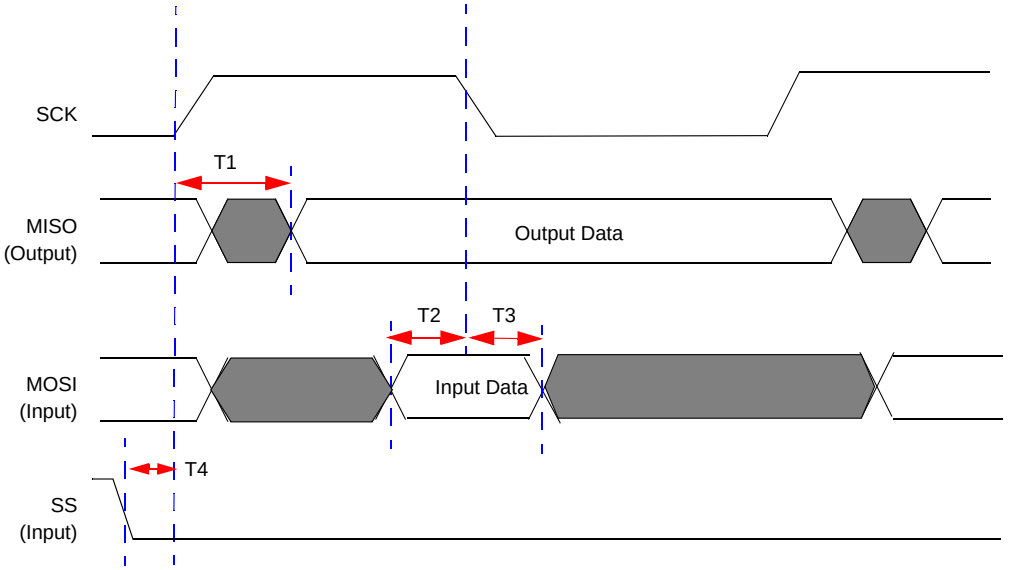



**Figure 97. SPI Slave Mode Timing**

**Table 111. SPI Slave Mode Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_1$ | SCK (transmit edge) to MISO output Valid Delay | 2 * Xin period | 3 * Xin period + 20 nsec |
| $T_2$ | MOSI input to SCK (receive edge) Setup Time | 0 | |
| $T_3$ | MOSI input to SCK (receive edge) Hold Time | 3 * Xin period | |
| $T_4$ | SS input assertion to SCK setup | 1 * Xin period | |

## I²C Timing

Figure 98 and Table 112 provide timing information for I2C pins.

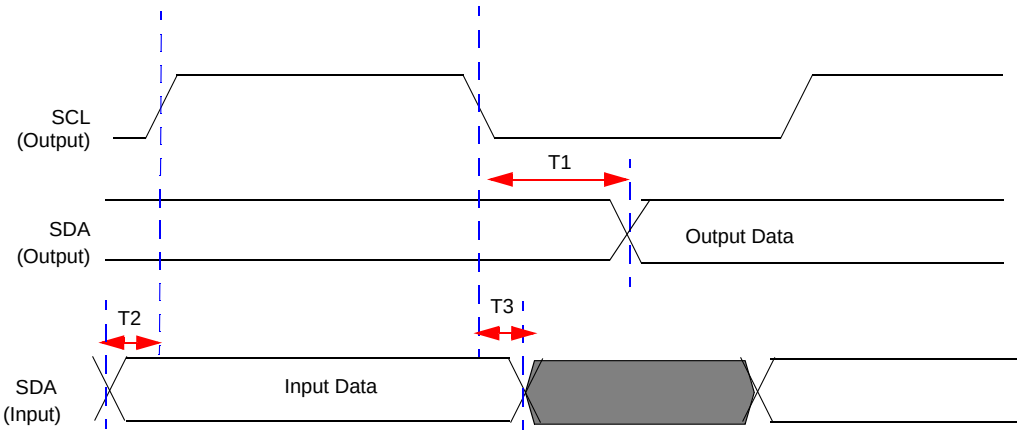



**Figure 98. I²C Timing**

**Table 112. I²C Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_1$ | SCL Fall to SDA output delay | SCL period/4 | |
| $T_2$ | SDA Input to SCL rising edge Setup Time | 0 | |
| $T_3$ | SDA Input to SCL falling edge Hold Time | 0 | |

# eZ8 CPU Instruction Set

## Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without having to be concerned with actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (opcodes and operands) to represent the instructions themselves. The opcodes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement can contain labels, operations, operands and comments.

Labels can be assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or pseudo-ops, are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the following example.

### Assembly Language Source Program Example

```
JP START          ; Everything after the semicolon is a comment.

START:            ; A label called “START”. The first instruction (JP START) in this
                  ; example causes program execution to jump to the point within the
                  ; program where the START label occurs.

LD R4, R7         ; A Load (LD) instruction with two operands. The first operand,
                  ; Working Register R4, is the destination. The second operand,
                  ; Working Register R7, is the source. The contents of R7 is
                  ; written into R4.

LD 234H, #%01     ; Another Load (LD) instruction with two operands.
                  ; The first operand, Extended Mode Register Address 234H,
                  ; identifies the destination. The second operand, Immediate Data
```

```
; value 01H, is the source. The value 01H is written into the
; Register at address 234H.
```

## Assembly Language Syntax

For proper instruction execution, eZ8 CPU assembly language syntax requires that the operands be written as ‘destination, source’. After assembly, the object code usually has the operands in the order ’source, destination’, but ordering is opcode-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. This binary format must be followed by users that prefer manual program coding or intend to implement their own assembler.

**Example 1**: If the contents of Registers 43H and 08H are added and the result is stored in 43H, the assembly syntax and resulting object code is:

**Table 113. Assembly Language Syntax Example 1**

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

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

**Table 114. Assembly Language Syntax Example 2**

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

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

## eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary and Description sections, the operands, condition codes, status flags, and address modes are represented by a notational shorthand that is described in Table 115

**Table 115. Notational Shorthand**

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

**Table 116. Additional Symbols**

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

Assignment of a value is indicated by an arrow. For example,

dst ← dst + src

indicates the source data is added to the destination data and the result is stored in the destination location.

## Condition Codes

The C, Z, S and V flags control the operation of the conditional jump (JP cc and JR cc) instructions. Sixteen frequently useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms Bits 7:4 of the conditional jump instructions. The condition codes are summarized in Table 117. Some binary condition codes can be created using more than one assembly code mnemonic. The result of the flag test operation is used to decide if the conditional jump is executed.

**Table 117. Condition Codes**

| Binary | Hex | Assembly Mnemonic | Definition | Flag Test Operation |
|---|---|---|---|---|
| 0000 | 0 | F | Always False | – |
| 0001 | 1 | LT | Less Than | (S XOR V) = 1 |
| 0010 | 2 | LE | Less Than or Equal | (Z OR (S XOR V)) = 1 |
| 0011 | 3 | ULE | Unsigned Less Than or Equal | (C OR Z) = 1 |
| 0100 | 4 | OV | Overflow | V = 1 |
| 0101 | 5 | Ml | Minus | S = 1 |
| 0110 | 6 | Z | Zero | Z = 1 |
| 0110 | 6 | EQ | Equal | Z = 1 |
| 0111 | 7 | C | Carry | C = 1 |
| 0111 | 7 | ULT | Unsigned Less Than | C = 1 |
| 1000 | 8 | T (or blank) | Always True | – |
| 1001 | 9 | GE | Greater Than or Equal | (S XOR V) = 0 |
| 1010 | A | GT | Greater Than | (Z OR (S XOR V)) = 0 |
| 1011 | B | UGT | Unsigned Greater Than | (C = 0 AND Z = 0) = 1 |
| 1100 | C | NOV | No Overflow | V = 0 |
| 1101 | D | PL | Plus | S = 0 |
| 1110 | E | NZ | Non-Zero | Z = 0 |
| 1110 | E | NE | Not Equal | Z = 0 |
| 1111 | F | NC | No Carry | C = 0 |
| 1111 | F | UGE | Unsigned Greater Than or Equal | C = 0 |

## eZ8 CPU Instruction Classes

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

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Tables 118 through 125 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table as these instruction can be considered as a subset of more than one category. Within these tables, the source operand is identified as 'src', the destination operand is 'dst' and a condition code is 'cc'.

**Table 118. Arithmetic Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| ADC | dst, src | Add with Carry |
| ADCX | dst, src | Add with Carry using Extended Addressing |
| ADD | dst, src | Add |
| ADDX | dst, src | Add using Extended Addressing |
| CP | dst, src | Compare |
| CPC | dst, src | Compare with Carry |
| CPCX | dst, src | Compare with Carry using Extended Addressing |
| CPX | dst, src | Compare using Extended Addressing |
| DA | dst | Decimal Adjust |
| DEC | dst | Decrement |
| DECW | dst | Decrement Word |
| INC | dst | Increment |
| INCW | dst | Increment Word |
| MULT | dst | Multiply |

**Table 118. Arithmetic Instructions (Continued)**

| Mnemonic | Operands | Instruction |
|---|---|---|
| SBC | dst, src | Subtract with Carry |
| SBCX | dst, src | Subtract with Carry using Extended Addressing |
| SUB | dst, src | Subtract |
| SUBX | dst, src | Subtract using Extended Addressing |

**Table 119. Bit Manipulation Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| BCLR | bit, dst | Bit Clear |
| BIT | p, bit, dst | Bit Set or Clear |
| BSET | bit, dst | Bit Set |
| BSWAP | dst | Bit Swap |
| CCF | — | Complement Carry Flag |
| RCF | — | Reset Carry Flag |
| SCF | — | Set Carry Flag |
| TCM | dst, src | Test Complement Under Mask |
| TCMX | dst, src | Test Complement Under Mask using Extended Addressing |
| TM | dst, src | Test Under Mask |
| TMX | dst, src | Test Under Mask using Extended Addressing |

**Table 120. Block Transfer Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| LDCI | dst, src | Load Constant to/from Program Memory and Auto-Increment Addresses |
| LDEI | dst, src | Load External Data to/from Data Memory and Auto-Increment Addresses |

**Table 121. CPU Control Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| CCF | — | Complement Carry Flag |
| DI | — | Disable Interrupts |
| EI | — | Enable Interrupts |
| HALT | — | Halt Mode |
| NOP | — | No Operation |
| RCF | — | Reset Carry Flag |
| SCF | — | Set Carry Flag |
| SRP | src | Set Register Pointer |
| STOP | — | Stop Mode |
| WDT | — | Watch-Dog Timer Refresh |

**Table 122. Load Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| CLR | dst | Clear |
| LD | dst, src | Load |
| LDC | dst, src | Load Constant to/from Program Memory |
| LDCI | dst, src | Load Constant to/from Program Memory and Auto-Increment Addresses |
| LDE | dst, src | Load External Data to/from Data Memory |
| LDEI | dst, src | Load External Data to/from Data Memory and Auto-Increment Addresses |
| LDX | dst, src | Load using Extended Addressing |
| LEA | dst, X(src) | Load Effective Address |
| POP | dst | Pop |
| POPX | dst | Pop using Extended Addressing |
| PUSH | src | Push |
| PUSHX | src | Push using Extended Addressing |

**Table 123. Logical Instructions**

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

**Table 124. Program Control Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| BRK | — | On-Chip Debugger Break |
| BTJ | p, bit, src, DA | Bit Test and Jump |
| BTJNZ | bit, src, DA | Bit Test and Jump if Non-Zero |
| BTJZ | bit, src, DA | Bit Test and Jump if Zero |
| CALL | dst | Call Procedure |
| DJNZ | dst, src, RA | Decrement and Jump Non-Zero |
| IRET | — | Interrupt Return |
| JP | dst | Jump |
| JP cc | dst | Jump Conditional |
| JR | DA | Jump Relative |
| JR cc | DA | Jump Relative Conditional |
| RET | — | Return |
| TRAP | vector | Software Trap |

**Table 125. Rotate and Shift Instructions**

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

## eZ8 CPU Instruction Summary

Table 126 summarizes the eZ8 CPU instructions. The table identifies the addressing modes employed by the instruction, the effect upon the Flags register, the number of CPU clock cycles required for the instruction fetch, and the number of CPU clock cycles required for the instruction execution.

**Table 126. eZ8 CPU Instruction Summary**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(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 |
| Flags Notation: | * = Value is a function of the result of the operation.<br>- = Unaffected<br>X = Undefined | | | | 0 = Reset to 0<br>1 = Set to 1 | | | | | | | |

**Table 126. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD dst, src | dst ← dst + src | r | r | 02 | * | * | * | * | 0 | * | 2 | 3 |
| | | r | Ir | 03 | | | | | | | 2 | 4 |
| | | R | R | 04 | | | | | | | 3 | 3 |
| | | R | IR | 05 | | | | | | | 3 | 4 |
| | | R | IM | 06 | | | | | | | 3 | 3 |
| | | IR | IM | 07 | | | | | | | 3 | 4 |
| ADDX dst, src | dst ← dst + src | ER | ER | 08 | * | * | * | * | 0 | * | 4 | 3 |
| | | ER | IM | 09 | | | | | | | 4 | 3 |
| AND dst, src | dst ← dst AND src | r | r | 52 | - | * | * | 0 | - | - | 2 | 3 |
| | | r | Ir | 53 | | | | | | | 2 | 4 |
| | | R | R | 54 | | | | | | | 3 | 3 |
| | | R | IR | 55 | | | | | | | 3 | 4 |
| | | R | IM | 56 | | | | | | | 3 | 3 |
| | | IR | IM | 57 | | | | | | | 3 | 4 |
| ANDX dst, src | dst ← dst AND src | ER | ER | 58 | - | * | * | 0 | - | - | 4 | 3 |
| | | ER | IM | 59 | | | | | | | 4 | 3 |
| BCLR bit, dst | dst[bit] ← 0 | r | | E2 | - | * | * | 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 |

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

**Table 126. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BTJZ bit, src, dst | if src[bit] = 0<br>PC ← PC + X | | r | F6 | - | - | - | - | - | - | 3 | 3 |
| | | | Ir | F7 | | | | | | | 3 | 4 |
| CALL dst | SP ← SP -2<br>@SP ← PC<br>PC ← dst | IRR | | D4 | - | - | - | - | - | - | 2 | 6 |
| | | DA | | D6 | | | | | | | 3 | 3 |
| CCF | C ← ~C | | | EF | * | - | - | - | - | - | 1 | 2 |
| CLR dst | dst ← 00H | R | | B0 | - | - | - | - | - | - | 2 | 2 |
| | | IR | | B1 | | | | | | | 2 | 3 |
| COM dst | dst ← ~dst | R | | 60 | - | * | * | 0 | - | - | 2 | 2 |
| | | IR | | 61 | | | | | | | 2 | 3 |
| CP dst, src | dst - src | r | r | A2 | * | * | * | * | - | - | 2 | 3 |
| | | r | Ir | A3 | | | | | | | 2 | 4 |
| | | R | R | A4 | | | | | | | 3 | 3 |
| | | R | IR | A5 | | | | | | | 3 | 4 |
| | | R | IM | A6 | | | | | | | 3 | 3 |
| | | IR | IM | A7 | | | | | | | 3 | 4 |
| CPC dst, src | dst - src - C | r | r | 1F A2 | * | * | * | * | - | - | 3 | 3 |
| | | r | Ir | 1F A3 | | | | | | | 3 | 4 |
| | | R | R | 1F A4 | | | | | | | 4 | 3 |
| | | R | IR | 1F A5 | | | | | | | 4 | 4 |
| | | R | IM | 1F A6 | | | | | | | 4 | 3 |
| | | IR | IM | 1F A7 | | | | | | | 4 | 4 |
| CPCX dst, src | dst - src - C | ER | ER | 1F A8 | * | * | * | * | - | - | 5 | 3 |
| | | ER | IM | 1F A9 | | | | | | | 5 | 3 |
| CPX dst, src | dst - src | ER | ER | A8 | * | * | * | * | - | - | 4 | 3 |
| | | ER | IM | A9 | | | | | | | 4 | 3 |

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

**Table 126. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DA dst | dst ← DA(dst) | R | | 40 | * | * | * | X | - | - | 2 | 2 |
| | | IR | | 41 | | | | | | | 2 | 3 |
| DEC dst | dst ← dst - 1 | R | | 30 | - | * | * | * | - | - | 2 | 2 |
| | | IR | | 31 | | | | | | | 2 | 3 |
| DECW dst | dst ← dst - 1 | RR | | 80 | - | * | * | * | - | - | 2 | 5 |
| | | IRR | | 81 | | | | | | | 2 | 6 |
| DI | IRQCTL[7] ← 0 | | | 8F | - | - | - | - | - | - | 1 | 2 |
| DJNZ dst, RA | dst ← dst – 1<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 126. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 3 |
| | | R | IM | E6 | | | | | | | 3 | 3 |
| | | 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 |
| Flags Notation: | * = Value is a function of the result of the operation.<br>- = Unaffected<br>X = Undefined | | | | 0 = Reset to 0<br>1 = Set to 1 | | | | | | | |

**Table 126. eZ8 CPU Instruction Summary (Continued)**

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

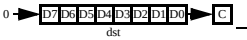
Flags Notation: * = Value is a function of the result of the operation.
- = Unaffected
X = Undefined
0 = Reset to 0
1 = Set to 1

**Table 126. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POP dst | dst ← @SP<br>SP ← SP + 1 | R | | 50 | - | - | - | - | - | - | 2 | 2 |
| | | IR | | 51 | | | | | | | 2 | 3 |
| 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 |
| 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 |

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

**Table 126. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SBC dst, src | dst ← dst – src - C | r | r | 32 | * | * | * | * | 1 | * | 2 | 3 |
| | | r | Ir | 33 | | | | | | | 2 | 4 |
| | | R | R | 34 | | | | | | | 3 | 3 |
| | | R | IR | 35 | | | | | | | 3 | 4 |
| | | R | IM | 36 | | | | | | | 3 | 3 |
| | | IR | IM | 37 | | | | | | | 3 | 4 |
| SBCX dst, src | dst ← dst – src - C | ER | ER | 38 | * | * | * | * | 1 | * | 4 | 3 |
| | | ER | IM | 39 | | | | | | | 4 | 3 |
| SCF | C ← 1 | | | DF | 1 | - | - | - | - | - | 1 | 2 |
| SRA dst | | R | | D0 | * | * | * | 0 | - | - | 2 | 2 |
| | | IR | | D1 | | | | | | | 2 | 3 |
| 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 |

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

**Table 126. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 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 |
| Flags Notation: | * = Value is a function of the result of the operation.<br>- = Unaffected<br>X = Undefined | | | | 0 = Reset to 0<br>1 = Set to 1 | | | | | | | |

**Table 126. eZ8 CPU Instruction Summary (Continued)**

| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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.<br>- = Unaffected<br>X = Undefined | | | | 0 = Reset to 0<br>1 = Set to 1 | | | | | | | |

## Flags Register

The Flags Register contains the status information regarding the most recent arithmetic, logical, bit manipulation or rotate and shift operation. The Flags Register contains six bits of status information that are set or cleared by CPU operations. Four of the bits (C, V, Z and S) can be tested for use with conditional jump instructions. Two flags (H and D) cannot be tested and are used for Binary-Coded Decimal (BCD) arithmetic.

The two remaining bits, User Flags (F1 and F2), are available as general-purpose status bits. User Flags are unaffected by arithmetic operations and must be set or cleared by instructions. The User Flags cannot be used with conditional Jumps. They are undefined at initial power-up and are unaffected by Reset. Figure 99 illustrates the flags and their bit positions in the Flags Register.

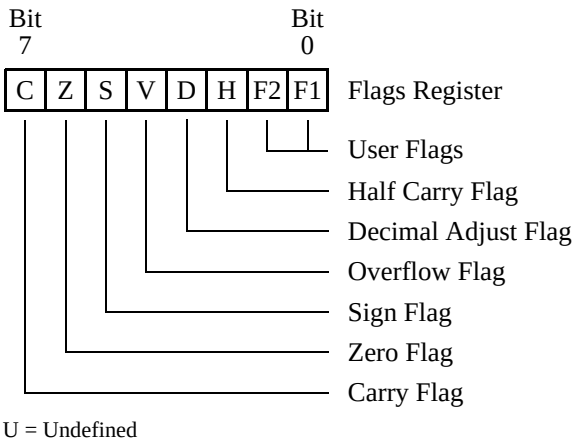



**Figure 99. Flags Register**

Interrupts, the Software Trap (TRAP) instruction, and Illegal Instruction Traps all write the value of the Flags Register to the stack. Executing an Interrupt Return (IRET) instruction restores the value saved on the stack into the Flags Register.

# Opcode Maps

Figures 101 and 102 provide information on each of the eZ8 CPU instructions. A description of the opcode map data and the abbreviations are provided in Figure 100 and Table 127.

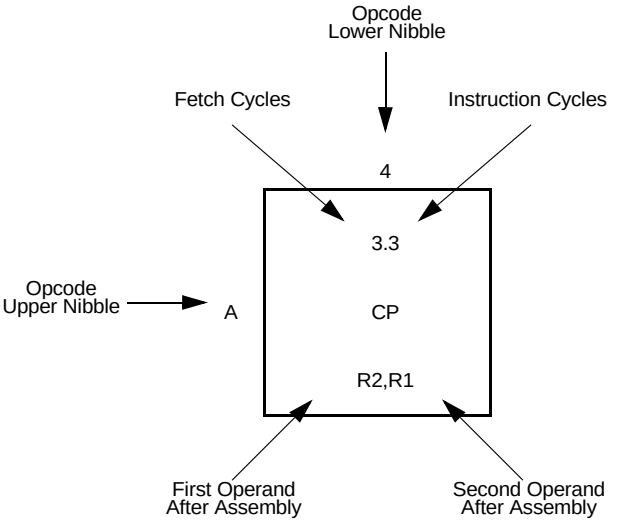



**Figure 100. Opcode Map Cell Description**

**Table 127. Opcode Map Abbreviations**

| Abbreviation | Description | Abbreviation | Description |
|---|---|---|---|
| b | Bit position | IRR | Indirect Register Pair |
| cc | Condition code | p | Polarity (0 or 1) |
| X | 8-bit signed index or displacement | r | 4-bit Working Register |
| DA | Destination address | R | 8-bit register |
| ER | Extended Addressing register | r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1 | Destination address |
| IM | Immediate data value | r2, R2, Ir2, Irr2, IR2, rr2, RR2, IRR2, ER2 | Source address |
| Ir | Indirect Working Register | RA | Relative |
| IR | Indirect register | rr | Working Register Pair |
| Irr | Indirect Working Register Pair | RR | Register Pair |

| Upper Nibble (Hex) \ Lower Nibble (Hex) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.2 **BRK** | 2.2 **SRP** IM | 2.3 **ADD** r1,r2 | 2.4 **ADD** r1,Ir2 | 3.3 **ADD** R2,R1 | 3.4 **ADD** IR2,R1 | 3.3 **ADD** R1,IM | 3.4 **ADD** IR1,IM | 4.3 **ADDX** ER2,ER1 | 4.3 **ADDX** IM,ER1 | 2.3 **DJNZ** r1,X | 2.2 **JR** cc,X | 2.2 **LD** r1,IM | 3.2 **JP** cc,DA | 1.2 **INC** r1 | 1.2 **NOP** |
| 1 | 2.2 **RLC** R1 | 2.3 **RLC** IR1 | 2.3 **ADC** r1,r2 | 2.4 **ADC** r1,Ir2 | 3.3 **ADC** R2,R1 | 3.4 **ADC** IR2,R1 | 3.3 **ADC** R1,IM | 3.4 **ADC** IR1,IM | 4.3 **ADCX** ER2,ER1 | 4.3 **ADCX** IM,ER1 | | | | | | See 2nd Opcode Map |
| 2 | 2.2 **INC** R1 | 2.3 **INC** IR1 | 2.3 **SUB** r1,r2 | 2.4 **SUB** r1,Ir2 | 3.3 **SUB** R2,R1 | 3.4 **SUB** IR2,R1 | 3.3 **SUB** R1,IM | 3.4 **SUB** IR1,IM | 4.3 **SUBX** ER2,ER1 | 4.3 **SUBX** IM,ER1 | | | | | | |
| 3 | 2.2 **DEC** R1 | 2.3 **DEC** IR1 | 2.3 **SBC** r1,r2 | 2.4 **SBC** r1,Ir2 | 3.3 **SBC** R2,R1 | 3.4 **SBC** IR2,R1 | 3.3 **SBC** R1,IM | 3.4 **SBC** IR1,IM | 4.3 **SBCX** ER2,ER1 | 4.3 **SBCX** IM,ER1 | | | | | | |
| 4 | 2.2 **DA** R1 | 2.3 **DA** IR1 | 2.3 **OR** r1,r2 | 2.4 **OR** r1,Ir2 | 3.3 **OR** R2,R1 | 3.4 **OR** IR2,R1 | 3.3 **OR** R1,IM | 3.4 **OR** IR1,IM | 4.3 **ORX** ER2,ER1 | 4.3 **ORX** IM,ER1 | | | | | | |
| 5 | 2.2 **POP** R1 | 2.3 **POP** IR1 | 2.3 **AND** r1,r2 | 2.4 **AND** r1,Ir2 | 3.3 **AND** R2,R1 | 3.4 **AND** IR2,R1 | 3.3 **AND** R1,IM | 3.4 **AND** IR1,IM | 4.3 **ANDX** ER2,ER1 | 4.3 **ANDX** IM,ER1 | | | | | | 1.2 **WDT** |
| 6 | 2.2 **COM** R1 | 2.3 **COM** IR1 | 2.3 **TCM** r1,r2 | 2.4 **TCM** r1,Ir2 | 3.3 **TCM** R2,R1 | 3.4 **TCM** IR2,R1 | 3.3 **TCM** R1,IM | 3.4 **TCM** IR1,IM | 4.3 **TCMX** ER2,ER1 | 4.3 **TCMX** IM,ER1 | | | | | | 1.2 **STOP** |
| 7 | 2.2 **PUSH** R2 | 2.3 **PUSH** IR2 | 2.3 **TM** r1,r2 | 2.4 **TM** r1,Ir2 | 3.3 **TM** R2,R1 | 3.4 **TM** IR2,R1 | 3.3 **TM** R1,IM | 3.4 **TM** IR1,IM | 4.3 **TMX** ER2,ER1 | 4.3 **TMX** IM,ER1 | | | | | | 1.2 **HALT** |
| 8 | 2.5 **DECW** RR1 | 2.6 **DECW** IRR1 | 2.5 **LDE** r1,Irr2 | 2.9 **LDEI** Ir1,Irr2 | 3.2 **LDX** r1,ER2 | 3.3 **LDX** Ir1,ER2 | 3.4 **LDX** IRR2,R1 | 3.5 **LDX** IRR2,IR1 | 3.4 **LDX** r1,rr2,X | 3.4 **LDX** rr1,r2,X | | | | | | 1.2 **DI** |
| 9 | 2.2 **RL** R1 | 2.3 **RL** IR1 | 2.5 **LDE** r2,Irr1 | 2.9 **LDEI** Ir2,Irr1 | 3.2 **LDX** r2,ER1 | 3.3 **LDX** Ir2,ER1 | 3.4 **LDX** R2,IRR1 | 3.5 **LDX** IR2,IRR1 | 3.3 **LEA** r1,r2,X | 3.5 **LEA** rr1,rr2,X | | | | | | 1.2 **EI** |
| A | 2.5 **INCW** RR1 | 2.6 **INCW** IRR1 | 2.3 **CP** r1,r2 | 2.4 **CP** r1,Ir2 | 3.3 **CP** R2,R1 | 3.4 **CP** IR2,R1 | 3.3 **CP** R1,IM | 3.4 **CP** IR1,IM | 4.3 **CPX** ER2,ER1 | 4.3 **CPX** IM,ER1 | | | | | | 1.4 **RET** |
| B | 2.2 **CLR** R1 | 2.3 **CLR** IR1 | 2.3 **XOR** r1,r2 | 2.4 **XOR** r1,Ir2 | 3.3 **XOR** R2,R1 | 3.4 **XOR** IR2,R1 | 3.3 **XOR** R1,IM | 3.4 **XOR** IR1,IM | 4.3 **XORX** ER2,ER1 | 4.3 **XORX** IM,ER1 | | | | | | 1.5 **IRET** |
| C | 2.2 **RRC** R1 | 2.3 **RRC** IR1 | 2.5 **LDC** r1,Irr2 | 2.9 **LDCI** Ir1,Irr2 | 2.3 **JP** IRR1 | 2.9 **LDC** Ir1,Irr2 | | 3.3 **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 101. First Opcode Map**

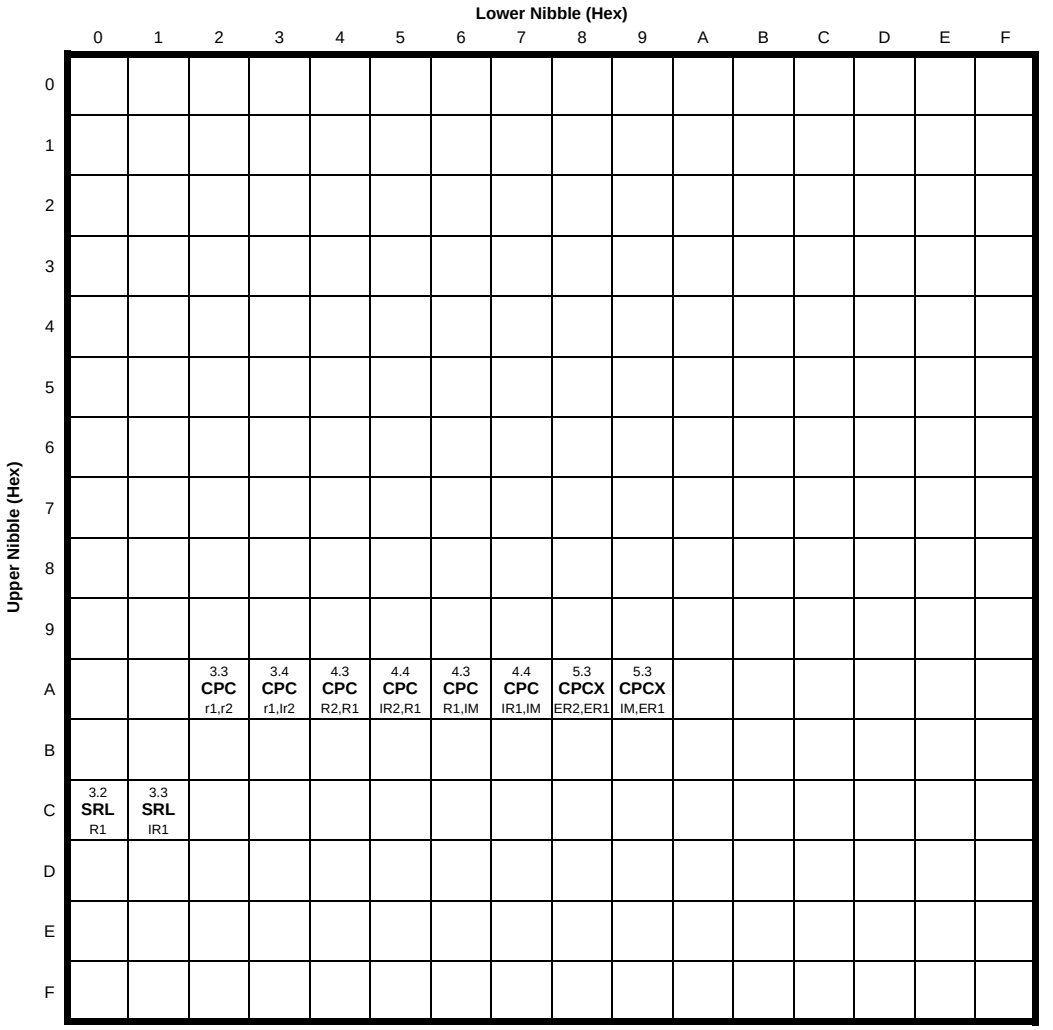
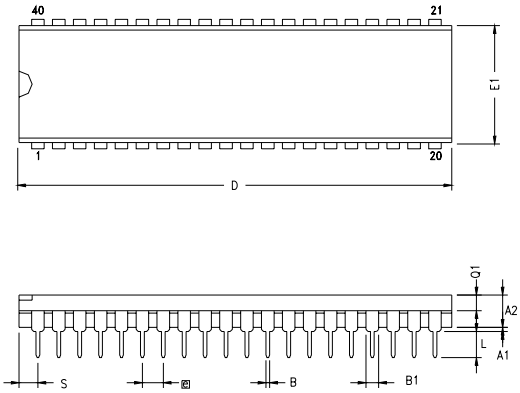



**Figure 102. Second Opcode Map after 1FH**

# *Packaging*

Figure 103 illustrates the 40-pin PDIP (plastic dual-inline package) available for the Z8F1601, Z8F2401, Z8F3201, Z8F4801, and Z8F6401 devices.

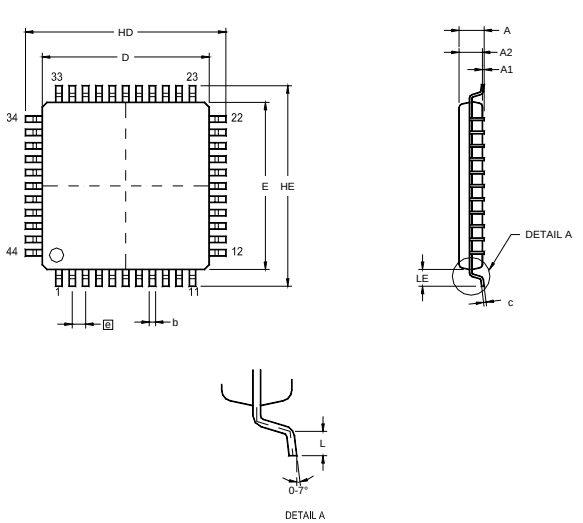



| SYMBOL | MILLIMETER | | INCH | |
|---|---|---|---|---|
| | MIN | MAX | MIN | MAX |
| A1 | 0.51 | 1.02 | .020 | .040 |
| A2 | 3.18 | 3.94 | .125 | .155 |
| B | 0.38 | 0.53 | .015 | .021 |
| B1 | 1.02 | 1.52 | .040 | .060 |
| C | 0.23 | 0.38 | .009 | .015 |
| D | 52.07 | 52.58 | 2.050 | 2.070 |
| E | 15.24 | 15.75 | .600 | .620 |
| E1 | 13.59 | 14.22 | .535 | .560 |
| e | 2.54 TYP | | .100 TYP | |
| eA | 15.49 | 16.76 | .610 | .660 |
| L | 3.05 | 3.81 | .120 | .150 |
| Q1 | 1.40 | 1.91 | .055 | .075 |
| S | 1.52 | 2.29 | .060 | .090 |

CONTROLLING DIMENSIONS : INCH

**Figure 103. 40-Lead Plastic Dual-Inline Package (PDIP)**

Figure 104 illustrates the 44-pin LQFP (low profile quad flat package) available for the Z8F1601, Z8F2401, Z8F3201, Z8F4801, and Z8F6401 devices.

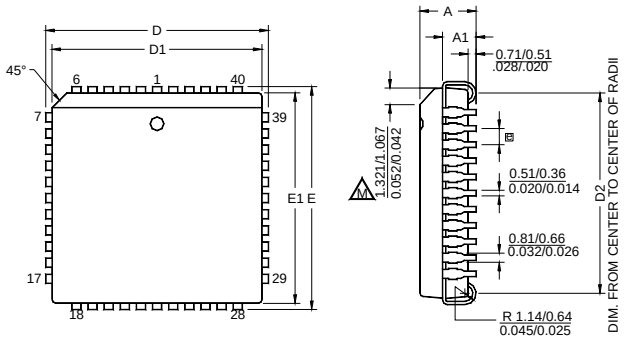



| SYMBOL | MILLIMETER | | INCH | |
|---|---|---|---|---|
| | MIN | MAX | MIN | MAX |
| A | 1.40 | 1.60 | 0.055 | 0.063 |
| A1 | 0.05 | 0.15 | 0.002 | 0.006 |
| A2 | 1.35 | 1.45 | 0.053 | 0.057 |
| b | 0.30 | 0.45 | 0.012 | 0.018 |
| c | 0.09 | 0.20 | 0.004 | 0.008 |
| HD | 11.75 | 12.25 | 0.463 | 0.482 |
| D | 9.90 | 10.10 | 0.390 | 0.398 |
| HE | 11.75 | 12.25 | 0.463 | 0.482 |
| E | 9.90 | 10.10 | 0.390 | 0.398 |
| e | 0.80 BSC | | 0.031 BSC | |
| L | 0.45 | 0.75 | 0.018 | 0.030 |
| LE | 1.00 REF | | 0.039 REF | |

1. CONTROLLING DIMENSIONS : mm
2. MAX. COPLANARITY : .10mm / 0.004"

**Figure 104. 44-Lead Low-Profile Quad Flat Package (LQFP)**

Figure 105 illustrates the 44-pin PLCC (plastic lead chip carrier) package available for the Z8F1601, Z8F2401, Z8F3201, Z8F4801, and Z8F6401 devices.

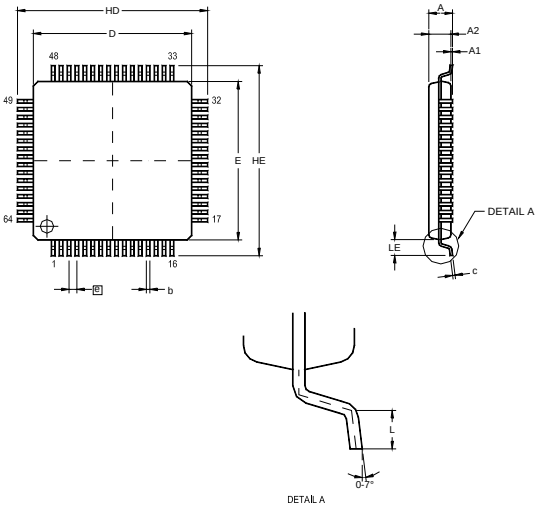



| SYMBOL | MILLIMETER | | INCH | |
|---|---|---|---|---|
| | MIN | MAX | MIN | MAX |
| A | 4.27 | 4.57 | 0.168 | 0.180 |
| A1 | 2.41 | 2.92 | 0.095 | 0.115 |
| D/E | 17.40 | 17.65 | 0.685 | 0.695 |
| D1/E1 | 16.51 | 16.66 | 0.650 | 0.656 |
| D2 | 15.24 | 16.00 | 0.600 | 0.630 |
| e | 1.27 BSC | | 0.050 BSC | |

NOTES:
1. CONTROLLING DIMENSION : INCH
2. LEADS ARE COPLANAR WITHIN 0.004".
3. DIMENSION : MM / INCH

**Figure 105. 44-Lead Plastic Lead Chip Carrier Package (PLCC)**

Figure 105 illustrates the 64-pin LQFP (low-profile quad flat package) available for the Z8F1602, Z8F2402, Z8F3202, Z8F4802, and Z8F6402 devices.

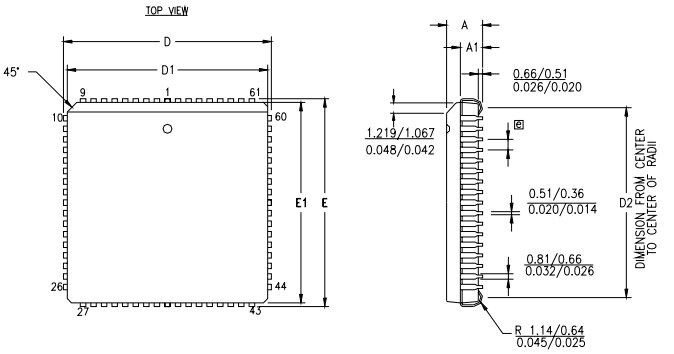



| SYMBOL | MILLIMETER | | INCH | |
|---|---|---|---|---|
| | MIN | MAX | MIN | MAX |
| A | 1.40 | 1.60 | 0.055 | 0.063 |
| A1 | 0.05 | 0.15 | 0.002 | 0.006 |
| A2 | 1.35 | 1.45 | 0.053 | 0.057 |
| b | 0.17 | 0.27 | 0.007 | 0.011 |
| c | 0.09 | 0.20 | 0.004 | 0.008 |
| HD | 11.75 | 12.25 | 0.463 | 0.482 |
| D | 9.90 | 10.10 | 0.390 | 0.398 |
| HE | 11.75 | 12.25 | 0.463 | 0.482 |
| E | 9.90 | 10.10 | 0.390 | 0.398 |
| e | 0.50 BSC | | 0.0197 BSC | |
| L | 0,45 | 0,75 | 0,018 | 0,030 |
| LE | 1.00 REF | | 0.039 REF | |

1. CONTROLLING DIMENSIONS : mm
2. MAX. COPLANARITY : 0.10mm / 0.004"

**Figure 106. 64-Lead Low-Profile Quad Flat Package (LQFP)**

Figure 107 illustrates the 68-pin PLCC (plastic lead chip carrier) package available for the Z8F1602, Z8F2402, Z8F3202, Z8F4802, and Z8F6402 devices.

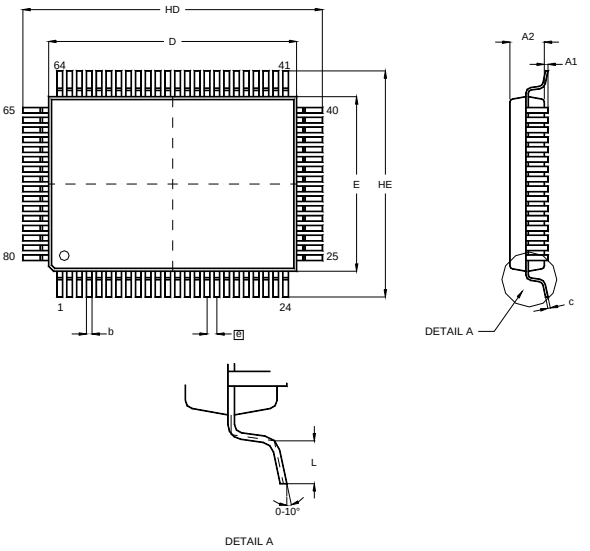



| SYMBOL | MILLIMETER | | INCH | |
|---|---|---|---|---|
| | MIN | MAX | MIN | MAX |
| A | 4.32 | 4.57 | .170 | .180 |
| A1 | 2.43 | 2.92 | .095 | .115 |
| D/E | 25.02 | 25.40 | .985 | 1.000 |
| D1/E1 | 24.13 | 24.33 | .950 | .958 |
| D2 | 22.86 | 23.62 | .900 | .930 |
| e | 1.27 BSC | | .050 BSC | |

NOTE:
1. CONTROLLING DIMENSIONS : INCH.
2. LEADS ARE COPLANAR WITHIN 0.004 IN. RANGE.
3. DIMENSION : MM
INCH.

**Figure 107. 68-Lead Plastic Lead Chip Carrier Package (PLCC)**

Figure 108 illustrates the 80-pin QFP (quad flat package) available for the Z8F4803 and Z8F6403 devices.




| SYMBOL | MILLIMETER | | INCH | |
|---|---|---|---|---|
| | MIN | MAX | MIN | MAX |
| A1 | 0.10 | 0.38 | .004 | .015 |
| A2 | 2.60 | 2.80 | .102 | .110 |
| b | 0.30 | 0.45 | .012 | .018 |
| c | 0.13 | 0.20 | .005 | .008 |
| HD | 23.70 | 24.15 | .933 | .951 |
| D | 19.90 | 20.10 | .783 | .791 |
| HE | 17.70 | 18.15 | .697 | .715 |
| E | 13.90 | 14.10 | .547 | .555 |
| e | 0.80 BSC | | .0315 BSC | |
| L | 0.70 | 1.10 | .028 | .043 |

NOTES:
1. CONTROLLING DIMENSIONS : MILLIMETER
2. LEAD COPLANARITY : MAX .10 / .004"

**Figure 108. 80-Lead Quad-Flat Package (QFP)**

# Ordering Information

**Table 128. Ordering Information**

| Part | Flash KB (Bytes) | RAM KB (Bytes) | Max. Speed (MHz) | Temp ($^0$C) | Voltage (V) | Package | Part Number |
|---|---|---|---|---|---|---|---|
| **Z8 Encore!® with 16KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F1601PM020SC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F1601AN020SC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F1601VN020SC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F1602AR020SC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F1602VS020SC |
| **Z8 Encore!® with 24KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F2401PM020SC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F2401AN020SC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F2401VN020SC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F2402AR020SC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F2402VS020SC |
| **Z8 Encore!® with 32KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F3201PM020SC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F3201AN020SC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F3201VN020SC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F3202AR020SC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F3202VS020SC |
| **Z8 Encore!®with 48KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F4801PM020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F4801AN020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F4801VN020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F4802AR020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F4802VS020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | QFP-80 | Z8F4803FT020SC |

**Table 128. Ordering Information (Continued)**

| Part | Flash KB (Bytes) | RAM KB (Bytes) | Max. Speed (MHz) | Temp ($^0$C) | Voltage (V) | Package | Part Number |
|---|---|---|---|---|---|---|---|
| **Z8 Encore! with 64KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F6401PM020SC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F6401AN020SC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F6401VN020SC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F6402AR020SC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F6402VS020SC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | QFP-80 | Z8F6403FT020SC |
| **Z8 Encore!® with 16KB Flash, Extended Temperature** | | | | | | | |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PDIP-40 | Z8F1601PM020EC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-44 | Z8F1601AN020EC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-44 | Z8F1601VN020EC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-64 | Z8F1602AR020EC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-68 | Z8F1602VS020EC |
| **Z8 Encore!® with 24KB Flash, Extended Temperature** | | | | | | | |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PDIP-40 | Z8F2401PM020EC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-44 | Z8F2401AN020EC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-44 | Z8F2401VN020EC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-64 | Z8F2402AR020EC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-68 | Z8F2402VS020EC |
| **Z8 Encore! with 32KB Flash, Extended Temperature** | | | | | | | |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PDIP-40 | Z8F3201PM020EC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-44 | Z8F3201AN020EC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-44 | Z8F3201VN020EC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-64 | Z8F3202AR020EC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-68 | Z8F3202VS020EC |

**Table 128. Ordering Information (Continued)**

| Part | Flash KB (Bytes) | RAM KB (Bytes) | Max. Speed (MHz) | Temp ($^0$C) | Voltage (V) | Package | Part Number |
|---|---|---|---|---|---|---|---|
| **Z8 Encore! ®with 48KB Flash, Extended Temperature** | | | | | | | |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | PDIP-40 | Z8F4801PM020EC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-44 | Z8F4801AN020EC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-44 | Z8F4801VN020EC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-64 | Z8F4802AR020EC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-68 | Z8F4802VS020EC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | QFP-80 | Z8F4803FT020EC |
| **Z8 Encore!® with 64KB Flash, Extended Temperature** | | | | | | | |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | PDIP-40 | Z8F6401PM020EC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-44 | Z8F6401AN020EC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-44 | Z8F6401VN020EC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | LQFP-64 | Z8F6402AR020EC |
| Z8 Encore!v | 64 (65,536) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | PLCC-68 | Z8F6402VS020EC |
| Z8 Encore!® | 64 (65,536) | 4 (4096) | 20 | -40 to +105 | 3.0 - 3.6 | QFP-80 | Z8F6403FT020EC |
| **Z8 Encore! ®Development Tools** | | | | | | | |
| Z8 Encore!® Developer Kit | | | | | | | Z8ENCORE000ZCO |

Contact ZILOG's worldwide customer support center for more information on ordering the Z8 Encore!®. The customer support center is open from 7 a.m. to 7 p.m. Pacific Time.

The customer support toll-free number for ZiLOG is 1-877-ZiLOGCS (1-877-945-6427). For Z8 Encore!® the customer support toll-free number is 1-866-498-3636. The FAX number for the customer support center is 1-603-316-0345. Customers can also gain access to customer support using the ZiLOG website. Z8 Encore!® has its own web page at www.zilog.com/z8encore.

For customer service, navigate your browser to:

- http://register.zilog.com/login.asp?login = servicelogin

For technical support, navigate your browser to:

- http://register.zilog.com/login.asp?login = supportlogin

For valuable information about hardware and software development tools, visit the ZiLOG web site at www.zilog.com. The latest released version of ZDS can be downloaded from this site.

## Part Number Description

ZiLOG part numbers consist of a number of components, as indicated in the following examples:

| ZiLOG Base Products | |
|---|---|
| Z8 | ZiLOG 8-bit microcontroller product |
| F6 | Flash Memory |
| 64 | Program Memory Size |
| 01 | Device Number |
| A | Package |
| N | Pin Count |
| 020 | Speed |
| S | Temperature Range |
| C | Environmental Flow |

| | |
|---|---|
| **Packages** | A = LQFP<br>S = SOIC<br>H = SSOP<br>P = PDIP<br>V = PLCC<br>F = QFP |
| **Pin Count** | H = 20 pins<br>J = 28 pins<br>M = 40 pins<br>N = 44 pins<br>R = 64 pins<br>S = 68 pins<br>T = 80 pins |
| **Speed** | 020 = 20MHz |
| **Temperature** | S = 0ºC to +70ºC<br>E = -40ºC to +105ºC |
| **Environmental Flow** | C = Plastic-Standard |

Example: Part number Z8F06401AN020SC is an 8-bit microcontroller product in an LQFP package, using 44 pins, operating with a maximum 20MHz external clock frequency over a 0ºC to +70ºC temperature range and built using the Plastic-Standard environmental flow.

### 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, due to start-up yield issues.

ZiLOG, Inc.
532 Race Street
San Jose, CA 95126
Telephone (408) 558-8500
FAX 408 558-8300
Internet: www.zilog.com

# *Document Information*

### Document Number Description

The Document Control Number that appears in the footer on each page of this document contains unique identifying attributes, as indicated in the following table:

| | |
|---|---|
| PS | Product Specification |
| 0176 | Unique Document Number |
| 01 | Revision Number |
| 0702 | Month and Year Published |

# *Customer Feedback Form*

## The Z8 Encore!™ Product Specification

If you experience any problems while operating this product, or if you note any inaccuracies while reading this Product Specification, please copy and complete this form, then mail or fax it to ZiLOG (see *Return Information*, below). We also welcome your suggestions!

## Customer Information

| | |
|---|---|
| Name | Country |
| Company | Phone |
| Address | Fax |
| City/State/Zip | E-Mail |

## Product Information

| |
|---|
| Part #, Serial #, Board Fab #, or Rev. # |
| Software Version |
| Document Number |
| Host Computer Description/Type |

## Return Information

ZiLOG
532 Race Street
San Jose, CA 95126
Fax: (408) 558-8536
Email: tools@zilog.com

## Problem Description or Suggestion

Provide a complete description of the problem or your suggestion. If you are reporting a specific problem, include all steps leading up to the occurrence of the problem. Attach additional pages as necessary.

# Index

## Symbols

# 185
% 185
@ 185

## Numerics

10-bit ADC 4
40-lead plastic dual-inline package 206
44-lead low-profile quad flat package 207
44-lead plastic lead chip carrier package 207
64-lead low-profile quad flat package 208
68-lead plastic lead chip carrier package 209
80-lead quad flat package 210

## A

absolute maximum ratings 167
AC characteristics 172
ADC 187
  architecture 132
  automatic power-down 133
  block diagram 133
  continuous conversion 134
  control register 135
  control register definitions 135
  data high byte register 137
  data low bits register 137
  DMA control 135
  electrical characteristics and timing 174
  operation 133
  single-shot conversion 133
ADCCTL register 135
ADCDH register 137
ADCDL register 137
ADCX 187
ADD 187
add - extended addressing 187
add with carry 187
add with carry - extended addressing 187
additional symbols 185
address space 17
ADDX 187
analog signals 14
analog-to-digital converter (ADC) 132
AND 190
ANDX 190
arithmetic instructions 187
assembly language programming 182
assembly language syntax 183

## B

B 185
b 184
baud rate generator, UART 85
BCLR 188
binary number suffix 185
BIT 188
bit 184
  clear 188
  manipulation instructions 188
  set 188
  set or clear 188
  swap 188
  test and jump 190
  test and jump if non-zero 190
  test and jump if zero 190
bit jump and test if non-zero 190
bit swap 191
block diagram 3
block transfer instructions 188
BRK 190
BSET 188
BSWAP 188, 191
BTJ 190
BTJNZ 190
BTJZ 190

## C

CALL procedure 190
capture mode 71
capture/compare mode 71

cc 184
CCF 189
characteristics, electrical 167
clear 189
clock phase (SPI) 102
CLR 189
COM 190
compare 71
compare - extended addressing 187
compare mode 71
compare with carry 187
compare with carry - extended addressing 187
complement 190
complement carry flag 188, 189
condition code 184
continuous assertion interrupt sources 47
continuous conversion (ADC) 134
continuous mode 70
control register definition, UART 86
control register, I2C 119
counter modes 70
CP 187
CPC 187
CPCX 187
CPU and peripheral overview 3
CPU control instructions 189
CPX 187
customer feedback form 216
customer information 216
customer service 213

## D

DA 184, 187
data memory 19
data register, I2C 118
DC characteristics 169
debugger, on-chip 151
DEC 187
decimal adjust 187
decrement 187
decrement and jump non-zero 190
decrement word 187
DECW 187
destination operand 185
device, port availability 33
DI 189
direct address 184
direct memory access controller 122
disable interrupts 189
DJNZ 190
DMA
  address high nibble register 126
  configuring for DMA_ADC data transfer 124
  confiigurting DMA0-1 data transfer 123
  control of ADC 135
  control register 124
  control register definitions 124
  controller 5
  DMA_ADC address register 128
  DMA_ADC control register 130
  DMA_ADC operation 123
  end address low byte register 128
  I/O address register 125
  operation 122
  start/current address low byte register 127
  status register 131
DMAA_STAT register 131
DMAACTL register 130
DMAxCTL register 124
DMAxEND register 128
DMAxH register 126
DMAxI/O address (DMAxIO) 126
DMAxIO register 126
DMAxSTART register 128
document number description 215
dst 185

## E

EI 189
electrical characteristics 167
  ADC 174
  flash memory and timing 173
  GPIO input data sample timing 176
  watch-dog timer 174
enable interrupt 189
ER 184

extended addressing register 184
external pin reset 29
eZ8 CPU features 3
eZ8 CPU instruction classes 187
eZ8 CPU instruction notation 183
eZ8 CPU instruction set 182
eZ8 CPU instruction summary 191

## F

FCTL register 144
features, Z8 Encore!® 1
first opcode map 204
FLAGS 185
flags register 185
flash
  controller 4
  option bit address space 148
  option bit configuration - reset 148
  program memory address 0000H 149
  program memory address 0001H 150
flash memory 138
  arrangement 139
  byte programming 142
  code protection 141
  configurations 138
  control register definitions 144
  controller bypass 143
  electrical characteristics and timing 173
  flash control register 144
  flash option bits 142
  flash status register 145
  flow chart 140
  frequency high and low byte registers 147
  mass erase 143
  operation 139
  operation timing 141
  page erase 143
  page select register 146
FPS register 146
FSTAT register 145

## G

gated mode 71
general-purpose I/O 33
GPIO 4, 33
  alternate functions 34
  architecture 34
  control register definitions 36
  input data sample timing 176
  interrupts 36
  port A-H address registers 37
  port A-H alternate function sub-registers 39
  port A-H control registers 38
  port A-H data direction sub-registers 39
  port A-H high drive enable sub-registers 41
  port A-H input data registers 42
  port A-H output control sub-registers 40
  port A-H output data registers 43
  port A-H stop mode recovery sub-registers 41
  port availability by device 33
  port input timing 176
  port output timing 177

## H

H 185
HALT 189
HALT mode 31, 189
hexadecimal number prefix/suffix 185

## I

$I^2C$ 4
  10-bit address read transaction 116
  10-bit address transaction 114
  10-bit addressed slave data transfer format 114
  10-bit receive data format 116
  7-bit address transaction 112
  7-bit address, reading a transaction 115
  7-bit addressed slave data transfer format 113
  7-bit receive data transfer format 115
  baud high and low byte registers 121
  C status register 118
  control register definitions 118

controller 111
controller signals 13
interrupts 112
operation 111
SDA and SCL signals 111
stop and start conditions 112
I2CBRH register 121
I2CBRL register 121
I2CCTL register 119
I2CDATA register 118
I2CSTAT register 118
IM 184
immediate data 184
immediate operand prefix 185
INC 187
increment 187
increment word 187
INCW 187
indexed 184
indirect address prefix 185
indirect register 184
indirect register pair 184
indirect working register 184
indirect working register pair 184
infrared encoder/decoder (IrDA) 95
instruction set, ez8 CPU 182
instructions
ADC 187
ADCX 187
ADD 187
ADDX 187
AND 190
ANDX 190
arithmetic 187
BCLR 188
BIT 188
bit manipulation 188
block transfer 188
BRK 190
BSET 188
BSWAP 188, 191
BTJ 190
BTJNZ 190
BTJZ 190
CALL 190
CCF 188, 189
CLR 189
COM 190
CP 187
CPC 187
CPCX 187
CPU control 189
CPX 187
DA 187
DEC 187
DECW 187
DI 189
DJNZ 190
EI 189
HALT 189
INC 187
INCW 187
IRET 190
JP 190
LD 189
LDC 189
LDCI 188, 189
LDE 189
LDEI 188
LDX 189
LEA 189
load 189
logical 190
MULT 187
NOP 189
OR 190
ORX 190
POP 189
POPX 189
program control 190
PUSH 189
PUSHX 189
RCF 188, 189
RET 190
RL 191
RLC 191
rotate and shift 191
RR 191

RRC 191
SBC 188
SCF 188, 189
SRA 191
SRL 191
SRP 189
STOP 189
SUB 188
SUBX 188
SWAP 191
TCM 188
TCMX 188
TM 188
TMX 188
TRAP 190
watch-dog timer refresh 189
XOR 190
XORX 190
instructions, eZ8 classes of 187
interrupt control register 56
interrupt controller 5, 44
architecture 44
interrupt assertion types 47
interrupt vectors and priority 47
operation 46
register definitions 48
interrupt edge select register 54
interrupt port select register 55
interrupt request 0 register 48
interrupt request 1 register 49
interrupt request 2 register 50
interrupt return 190
interrupt vector listing 44
interrupts
not acknowledge 112
receive 112
SPI 105
transmit 112
UART 85
introduction 1
IR 184
Ir 184
IrDA
architecture 95
block diagram 95
control register definitions 98
jitter 98
operation 96
receiving data 97
transmitting data 96
IRET 190
IRQ0 enable high and low bit registers 51
IRQ1 enable high and low bit registers 52
IRQ2 enable high and low bit registers 53
IRR 184
Irr 184

## J

jitter 98
JP 190
jump, conditional, relative, and relative conditional 190

## L

LD 189
LDC 189
LDCI 188, 189
LDE 189
LDEI 188, 189
LDX 189
LEA 189
load 189
load constant 188
load constant to/from program memory 189
load constant with auto-increment addresses 189
load effective address 189
load external data 189
load external data to/from data memory and auto-increment addresses 188
load external to/from data memory and auto-increment addresses 189
load instructions 189
load using extended addressing 189
logical AND 190
logical AND/extended addressing 190
logical exclusive OR 190

logical exclusive OR/extended addressing 190
logical instructions 190
logical OR 190
logical OR/extended addressing 190
low power modes 31
LQFP
  44 lead 207
  64 lead 208

## M

master interrupt enable 46
master-in, slave-out and-in 101
memory
  data 19
  program 18
MISO 101
mode
  capture 71
  capture/compare 71
  continuous 70
  counter 70
  gated 71
  one-shot 70
  PWM 70
modes 71
MOSI 101
MULT 187
multiply 187
multiprocessor mode, UART 84

## N

NOP (no operation) 189
not acknowledge interrupt 112
notation
  b 184
  cc 184
  DA 184
  ER 184
  IM 184
  IR 184
  Ir 184
  IRR 184
  Irr 184
  p 184
  R 184
  r 184
  RA 184
  RR 184
  rr 184
  vector 184
  X 184
notational shorthand 184

## O

OCD
  architecture 151
  auto-baud detector/generator 154
  baud rate limits 154
  block diagram 151
  breakpoints 155
  commands 156
  control register 161
  data format 154
  DBG pin to RS-232 Interface 152
  debug mode 153
  debugger break 190
  interface 152
  serial errors 155
  status register 162
  timing 178
  watchpoint address register 164
  watchpoint control register 163
  watchpoint data register 164
  watchpoints 155
OCD commands
  execute instruction (12H) 160
  read data memory (0DH) 160
  read OCD control register (05H) 158
  read OCD revision (00H) 157
  read OCD status register (02H) 157
  read program counter (07H) 158
  read program memory (0BH) 159
  read program memory CRC (0EH) 160
  read register (09H) 158
  read runtime counter (03H) 157

read watchpoint (21H) 161
step instruction (10H) 160
stuff instruction (11H) 160
write data memory (0CH) 159
write OCD control register (04H) 158
write program counter (06H) 158
write program memory (0AH) 159
write register (08H) 158
write watchpoint (20H) 161
on-chip debugger 5
on-chip debugger (OCD) 151
on-chip debugger signals 14
on-chip oscillator 165
one-shot mode 70
opcode map
abbreviations 203
cell description 202
first 204
second after 1FH 205
OR 190
ordering information 211
ORX 190
oscillator signals 14

## P

p 184
packaging
LQFP
44 lead 207
64 lead 208
PDIP 206
PLCC
44 lead 207
68 lead 209
QFP 210
part number description 214
part selection guide 2
PC 185
PDIP 206
peripheral AC and DC electrical characteristics 173
PHASE=0 timing (SPI) 103
PHASE=1 timing (SPI) 104
pin characteristics 15
PLCC
44 lead 207
68-lead 209
polarity 184
POP 189
pop using extended addressing 189
POPX 189
port availability, device 33
port input timing (GPIO) 176
port output timing, GPIO 177
power supply signals 15
power-down, automatic (ADC) 133
power-on and voltage brown-out 173
power-on reset (POR) 27
problem description or suggestion 217
product information 216
program control instructions 190
program counter 185
program memory 18
PUSH 189
push using extended addressing 189
PUSHX 189
PWM mode 70
PxADDR register 37
PxCTL register 38

## Q

QFP 210

## R

R 184
r 184
RA, register address 184
RCF 188, 189
receive
10-bit data format (I2C) 116
7-bit data transfer format (I2C) 115
IrDA data 97
receive interrupt 112
receiving UART data-DMA controller 83
receiving UART data-interrupt-driven method 82
receiving UART data-polled method 82

register 109, 126, 184
ADC control (ADCCTL) 135
ADC data high byte (ADCDH) 137
ADC data low bits (ADCDL) 137
baud low and high byte (I2C) 121
baud rate high and low byte (SPI) 110
control (SPI) 107
control, I2C 119
data, SPI 106
DMA status (DMAA_STAT) 131
DMA_ADC address 128
DMA_ADC control DMAACTL) 130
DMAx address high nibble (DMAxH) 126
DMAx control (DMAxCTL) 124
DMAx end/address low byte (DMAxEND) 128
DMAx start/current address low byte register (DMAxSTART) 128
flash control (FCTL) 144
flash high and low byte (FFREQH and FREEQL) 147
flash page select (FPS) 146
flash status (FSTAT) 145
GPIO port A-H address (PxADDR) 37
GPIO port A-H alternate function sub-registers 39
GPIO port A-H control address (PxCTL) 38
GPIO port A-H data direction sub-registers 39
I2C baud rate high (I2CBRH) 121
I2C control (I2CCTL) 119
I2C data (I2CDATA) 118
I2C status 118
I2C status (I2CSTAT) 118
I2Cbaud rate low (I2CBRL) 121
mode, SPI 109
OCD control 161
OCD status 162
OCD watchpoint address 164
OCD watchpoint control 163
OCD watchpoint data 164
SPI baud rate high byte (SPIBRH) 110
SPI baud rate low byte (SPIBRL) 110
SPI control (SPICTL) 107
SPI data (SPIDATA) 106
SPI status (SPISTAT) 108
status, I2C 118
status, SPI 108
UARTx baud rate high byte (UxBRH) 91
UARTx baud rate low byte (UxBRL) 92
UARTx Control 0 (UxCTL0) 89
UARTx control 1 (UxCTL1) 90
UARTx receive data (UxRXD) 87
UARTx status 0 (UxSTAT0) 87
UARTx status 1 (UxSTAT1) 89
UARTx transmit data (UxTXD) 86
watch-dog timer control (WDTCTL) 75
watch-dog timer reload high byte (WDTH) 76
watch-dog timer reload low byte (WDTL) 77
watch-dog timer reload upper byte (WDTU) 76
register file 17
register file address map 20
register pair 184
register pointer 185
reset
and stop mode characteristics 25
and stop mode recovery 25
carry flag 188
controller 5
sources 26
RET 190
return 190
return information 216
RL 191
RLC 191
rotate and shift instructions 191
rotate left 191
rotate left through carry 191
rotate right 191
rotate right through carry 191
RP 185
RR 184, 191
rr 184
RRC 191

## S

SBC 188
SCF 188, 189
SCK 101

SDA and SCL (IrDA) signals 111
second opcode map after 1FH 205
serial clock 101
serial peripheral interface (SPI) 99
set carry flag 188, 189
set register pointer 189
shift right arithmetic 191
shift right logical 191
signal descriptions 13
single assertion (pulse) interrupt sources 47
single-shot conversion (ADC) 133
SIO 5
slave data transfer formats (I2C) 114
slave select 102
software trap 190
source operand 185
SP 185
SPI
  architecture 99
  baud rate generator 105
  baud rate high and low byte register 110
  clock phase 102
  configured as slave 100
  control register 107
  control register definitions 106
  data register 106
  error detection 105
  interrupts 105
  mode fault error 105
  mode register 109
  multi-master operation 104
  operation 100
  overrun error 105
  signals 101
  single master, multiple slave system 100
  single master, single slave system 99
  status register 108
  timing, PHASE = 0 103
  timing, PHASE=1 104
SPI controller signals 13
SPI mode (SPIMODE) 109
SPIBRH register 110
SPIBRL register 110
SPICTL register 107
SPIDATA register 106
SPIMODE register 109
SPISTAT register 108
SRA 191
src 185
SRL 191
SRP 189
SS, SPI signal 101
stack pointer 185
status register, I2C 118
STOP 189
stop mode 31, 189
stop mode recovery
  sources 29
  using a GPIO port pin transition 30
  using watch-dog timer time-out 29
SUB 188
subtract 188
subtract - extended addressing 188
subtract with carry 188
subtract with carry - extended addressing 188
SUBX 188
SWAP 191
swap nibbles 191
symbols, additional 185
system and short resets 26

## T

TCM 188
TCMX 188
technical support 213
test complement under mask 188
test under mask 188
timer signals 14
timers 5, 57
  architecture 57
  block diagram 58
  capture mode 62, 71
  capture/compare mode 65, 71
  compare mode 63, 71
  continuous mode 59, 70
  counter mode 60
  counter modes 70

gated mode 64, 71
one-shot mode 58, 70
operating mode 58
PWM mode 61, 70
reading the timer count values 66
reload high and low byte registers 67
timer control register definitions 66
timer output signal operation 66
timers 0-3
control registers 70
high and low byte registers 66, 69
TM, TMX 188
tools, hardware and software 214
transmit
IrDA data 96
transmit interrupt 112
transmitting UART data-polled method 80
transmitting UART data-interrupt-driven method 81
TRAP 190

## U

UART 4
architecture 78
asynchronous data format without/with parity 80
baud rate generator 85
baud rates table 93
control register definitions 86
controller signals 14
data format 79
interrupts 85
multiprocessor mode 84
receiving data using DMA controller 83
receiving data using interrupt-driven method 82
receiving data using the polled method 82
transmitting data using the interrupt-driven method 81
transmitting data using the polled method 80
x baud rate high and low registers 91
x control 0 and control 1 registers 89
x status 0 and status 1 registers 87
UxBRH register 91
UxBRL register 92
UxCTL0 register 89
UxCTL1 register 90
UxRXD register 87
UxSTAT0 register 87
UxSTAT1 register 89
UxTXD register 86

## V

vector 184
voltage brown-out reset (VBR) 27

## W

watch-dog timer
approximate time-out delays 72, 73
CNTL 28
control register 75
electrical characteristics and timing 174
interrupt in normal operation 73
interrupt in stop mode 73
operation 72
refresh 73, 189
reload unlock sequence 74
reload upper, high and low registers 76
reset 28
reset in normal operation 74
reset in stop mode 74
time-out response 73
WDTCTL register 75
WDTH register 76
WDTL register 77
working register 184
working register pair 184
WTDU register 76

## X

X 184
XOR 190
XORX 190

## Z

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