Embedded in Life

An IXYS Company

## High Performance 8-Bit Microcontrollers

# Z8 Encore! XP® F64xx Series

## Product Specification

PS019926-1114

Copyright ©2014 Zilog®, Inc. All rights reserved.

www.zilog.com

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

## LIFE SUPPORT POLICY

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

### As used herein

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

### Document Disclaimer

©2014 Zilog, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

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

# Revision History

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

| Date | Revision Level | Description | Page |
|---|---|---|---|
| Nov 2014 | 26 | Corrected $I_{PU}$ Units value in DC Characteristics table to µA from incorrect mA. | 203 |
| Feb 2014 | 25 | Added footnote to Z8 Encore! XP F64xx Series Ordering Matrix table specific to 64-pin LQFP packages. | 287 |
| Jan 2013 | 24 | Restored 40-pin PDIP package to Signal and Pin Descriptions and Packaging chapters. | 7, 286 |
| Feb 2012 | 23 | Corrected formatting of $I_{DDS}$ section, Table 107; corrected language in the General Purpose RAM section of Appendix A; | 202, 248 |
| Sep 2011 | 22 | Revised Flash Sector Protect Register description; revised Packaging chapter. | 178, 286 |
| Mar 2008 | 21 | Changed title to Z8 Encore! XP F64xx Series. | All |
| Feb 2008 | 20 | Changed Z8 Encore! XP 64K Series Flash Microcontrollers to Z8 Encore! XP F64xx Series Flash Microcontrollers. Deleted three sentences that mentioned Z8R642. Removed the 40 PDIP package. Added ZENETSC0100ZACG to the end of the Ordering Information table. Changed the flag status to unaffected for BIT, BSET, and BCLR in the eZ8 CPU Instruction Summary table. | 287, 234 |
| Dec 2007 | 19 | Updated Zilog logo, Disclaimer section, and implemented style guide. Updated Table 113. Changed Z8 Encore! 64K Series to Z8 Encore! XP 64K Series Flash Microcontrollers throughout the document. | All |
| Dec 2006 | 18 | Updated Flash Memory Electrical Characteristics and Timing table and Ordering Information chapter. | 213, 287 |
| Nov 2006 | 17 | Updated Part Number Suffix Designations section. | 292 |

# Table of Contents

Revision History . . . . . iii

List of Figures . . . . . xi

List of Tables . . . . . xiii

Introduction . . . . . 1
Features . . . . . 1
Part Selection Guide . . . . . 2
Block Diagram . . . . . 3
CPU and Peripheral Overview . . . . . 3
General-Purpose Input/Output . . . . . 4
Flash Controller . . . . . 4
10-Bit Analog-to-Digital Converter . . . . . 4
UARTs . . . . . 4
$I^2C$ . . . . . 5
Serial Peripheral Interface . . . . . 5
Timers . . . . . 5
Interrupt Controller . . . . . 5
Reset Controller . . . . . 5
On-Chip Debugger . . . . . 5
DMA Controller . . . . . 6

Signal and Pin Descriptions . . . . . 7
Available Packages . . . . . 7
Pin Configurations . . . . . 8
Signal Descriptions . . . . . 14
Pin Characteristics . . . . . 17

Address Space . . . . . 18
Register File . . . . . 18
Program Memory . . . . . 19
Data Memory . . . . . 20
Information Area . . . . . 20

Register File Address Map . . . . . 22

Reset and Stop Mode Recovery . . . . . 28
Reset Types . . . . . 28
Reset Sources . . . . . 29
Power-On Reset . . . . . 30
Voltage Brown-Out Reset . . . . . 31

Watchdog Timer Reset . . . . . 32
External Pin Reset . . . . . 32
On-Chip Debugger Initiated Reset . . . . . 32
Stop Mode Recovery . . . . . 32
Stop Mode Recovery Using Watchdog Timer Time-Out . . . . . 33
Stop Mode Recovery Using a GPIO Port Pin Transition HALT . . . . . 33

Low-Power Modes . . . . . 34
Stop Mode . . . . . 34
Halt Mode . . . . . 35

General-Purpose I/O . . . . . 36
GPIO Port Availability By Device . . . . . 36
Architecture . . . . . 37
GPIO Alternate Functions . . . . . 37
GPIO Interrupts . . . . . 39
GPIO Control Register Definitions . . . . . 39
Port A–H Address Registers . . . . . 40
Port A–H Control Registers . . . . . 41
Port A–H Input Data Registers . . . . . 46
Port A–H Output Data Register . . . . . 46

Interrupt Controller . . . . . 47
Interrupt Vector Listing . . . . . 47
Architecture . . . . . 49
Operation . . . . . 49
Master Interrupt Enable . . . . . 49
Interrupt Vectors and Priority . . . . . 50
Interrupt Assertion . . . . . 50
Software Interrupt Assertion . . . . . 51
Interrupt Control Register Definitions . . . . . 51
Interrupt Request 0 Register . . . . . 51
Interrupt Request 1 Register . . . . . 53
Interrupt Request 2 Register . . . . . 54
IRQ0 Enable High and Low Bit Registers . . . . . 55
IRQ1 Enable High and Low Bit Registers . . . . . 56
IRQ2 Enable High and Low Bit Registers . . . . . 58
Interrupt Edge Select Register . . . . . 60
Interrupt Port Select Register . . . . . 60
Interrupt Control Register . . . . . 61

Timers . . . . . 62
Architecture . . . . . 62

Operation . . . . . 63
Timer Operating Modes . . . . . 63
Reading the Timer Count Values . . . . . 71
Timer Output Signal Operation . . . . . 72
Timer Control Register Definitions . . . . . 72
Timer 0–3 High and Low Byte Registers . . . . . 72
Timer Reload High and Low Byte Registers . . . . . 74
Timer 0–3 PWM High and Low Byte Registers . . . . . 75
Timer 0–3 Control 0 Registers . . . . . 76
Timer 0–3 Control 1 Registers . . . . . 77

Watchdog Timer . . . . . 80
Operation . . . . . 80
Watchdog Timer Refresh . . . . . 81
Watchdog Timer Time-Out Response . . . . . 81
Watchdog Timer Reload Unlock Sequence . . . . . 82
Watchdog Timer Control Register Definitions . . . . . 83
Watchdog Timer Control Register . . . . . 83
Watchdog Timer Reload Upper, High and Low Byte Registers . . . . . 85

Universal Asynchronous Receiver/Transmitter . . . . . 87
Architecture . . . . . 87
Operation . . . . . 88
Transmitting Data using the Polled Method . . . . . 89
Transmitting Data using the Interrupt-Driven Method . . . . . 90
Receiving Data using the Polled Method . . . . . 91
Receiving Data using the Interrupt-Driven Method . . . . . 92
Clear To Send (CTS) Operation . . . . . 93
Multiprocessor (9-Bit) Mode . . . . . 93
External Driver Enable . . . . . 95
UART Interrupts . . . . . 96
UART Baud Rate Generator . . . . . 98
UART Control Register Definitions . . . . . 98
UART Transmit Data Register . . . . . 98
UART Receive Data Register . . . . . 99
UART Status 0 Register . . . . . 100
UART Status 1 Register . . . . . 101
UART Control 0 and Control 1 Registers . . . . . 102
UART Address Compare Register . . . . . 105
UART Baud Rate High and Low Byte Registers . . . . . 105

Infrared Encoder/Decoder . . . . . 109
Architecture . . . . . 109

Operation . . . . . 109
Transmitting IrDA Data . . . . . 110
Receiving IrDA Data . . . . . 111
Infrared Encoder/Decoder Control Register Definitions . . . . . 112

Serial Peripheral Interface . . . . . 113
Architecture . . . . . 113
Operation . . . . . 115
SPI Signals . . . . . 115
SPI Clock Phase and Polarity Control . . . . . 116
Multimaster Operation . . . . . 118
Slave Operation . . . . . 119
Error Detection . . . . . 119
SPI Interrupts . . . . . 120
SPI Baud Rate Generator . . . . . 120
SPI Control Register Definitions . . . . . 121
SPI Data Register . . . . . 121
SPI Control Register . . . . . 122
SPI Status Register . . . . . 123
SPI Mode Register . . . . . 125
SPI Diagnostic State Register . . . . . 126
SPI Baud Rate High and Low Byte Registers . . . . . 126

$I^2C$ Controller . . . . . 128
Architecture . . . . . 128
Operation . . . . . 129
SDA and SCL Signals . . . . . 130
$I^2C$ Interrupts . . . . . 130
Software Control of $I^2C$ Transactions . . . . . 131
Start and Stop Conditions . . . . . 132
Master Write and Read Transactions . . . . . 132
Address Only Transaction with a 7-bit Address . . . . . 133
Write Transaction with a 7-Bit Address . . . . . 133
Address Only Transaction with a 10-bit Address . . . . . 135
Write Transaction with a 10-Bit Address . . . . . 136
Read Transaction with a 7-Bit Address . . . . . 138
Read Transaction with a 10-Bit Address . . . . . 139
$I^2C$ Control Register Definitions . . . . . 141
$I^2C$ Data Register . . . . . 141
$I^2C$ Status Register . . . . . 142
$I^2C$ Control Register . . . . . 144
$I^2C$ Baud Rate High and Low Byte Registers . . . . . 145

I2C Diagnostic State Register . . . . . 147
I2C Diagnostic Control Register . . . . . 149

Direct Memory Access Controller . . . . . 150
Operation . . . . . 150
Configuring DMA0 and DMA1 for Data Transfer . . . . . 150
DMA_ADC Operation . . . . . 151
Configuring DMA_ADC for Data Transfer . . . . . 152
DMA Control Register Definitions . . . . . 152
DMAx Control Register . . . . . 153
DMAx I/O Address Register . . . . . 154
DMAx Address High Nibble Register . . . . . 155
DMAx Start/Current Address Low Byte Register . . . . . 156
DMAx End Address Low Byte Register . . . . . 156
DMA_ADC Address Register . . . . . 157
DMA_ADC Control Register . . . . . 158
DMA_ADC Status Register . . . . . 159

Analog-to-Digital Converter . . . . . 161
Architecture . . . . . 161
Operation . . . . . 163
Automatic Power-Down . . . . . 163
Single-Shot Conversion . . . . . 163
Continuous Conversion . . . . . 164
DMA Control of the ADC . . . . . 165
ADC Control Register Definitions . . . . . 165
ADC Control Register . . . . . 165
ADC Data High Byte Register . . . . . 167
ADC Data Low Bits Register . . . . . 168

Flash Memory . . . . . 169
Information Area . . . . . 170
Operation . . . . . 171
Timing Using the Flash Frequency Registers . . . . . 171
Flash Read Protection . . . . . 172
Flash Write/Erase Protection . . . . . 172
Byte Programming . . . . . 173
Page Erase . . . . . 174
Mass Erase . . . . . 174
Flash Controller Bypass . . . . . 174
Flash Controller Behavior in Debug Mode . . . . . 175
Flash Control Register Definitions . . . . . 175
Flash Control Register . . . . . 175

Flash Status Register . . . . . 177
Page Select Register . . . . . 177
Flash Sector Protect Register . . . . . 178
Flash Frequency High and Low Byte Registers . . . . . 179

Option Bits . . . . . 180
Operation . . . . . 180
Option Bit Configuration By Reset . . . . . 180
Option Bit Address Space . . . . . 180
Flash Memory Address 0000h . . . . . 181
Flash Memory Address 0001h . . . . . 182

On-Chip Debugger . . . . . 183
Architecture . . . . . 183
Operation . . . . . 184
OCD Interface . . . . . 184
Debug Mode . . . . . 185
OCD Data Format . . . . . 186
OCD Autobaud Detector/Generator . . . . . 186
OCD Serial Errors . . . . . 187
Breakpoints . . . . . 187
On-Chip Debugger Commands . . . . . 188
On-Chip Debugger Control Register Definitions . . . . . 193
OCD Control Register . . . . . 193
OCD Status Register . . . . . 194

On-Chip Oscillator . . . . . 196
Operating Modes . . . . . 196
Crystal Oscillator Operation . . . . . 196
Oscillator Operation with an External RC Network . . . . . 198

Electrical Characteristics . . . . . 200
Absolute Maximum Ratings . . . . . 200
DC Characteristics . . . . . 202
On-Chip Peripheral AC and DC Electrical Characteristics . . . . . 211
AC Characteristics . . . . . 216
General-Purpose I/O Port Input Data Sample Timing . . . . . 217
General-Purpose I/O Port Output Timing . . . . . 218
On-Chip Debugger Timing . . . . . 219
SPI Master Mode Timing . . . . . 220
SPI Slave Mode Timing . . . . . 221
$I^2C$ Timing . . . . . 222
UART Timing . . . . . 223

eZ8 CPU Instruction Set . . . . . 225
Assembly Language Programming Introduction . . . . . 225
Assembly Language Syntax . . . . . 226
eZ8 CPU Instruction Notation . . . . . 227
Condition Codes . . . . . 229
eZ8 CPU Instruction Classes . . . . . 230
eZ8 CPU Instruction Summary . . . . . 234
Flags Register . . . . . 243
Op Code Maps . . . . . 244
Appendix A. Register Tables . . . . . 248
General Purpose RAM . . . . . 248
Timer 0 . . . . . 248
Universal Asynchronous Receiver/Transmitter (UART) . . . . . 256
Inter-Integrated Circuit ($I^2C$) . . . . . 261
Serial Peripheral Interface . . . . . 263
Analog-to-Digital Converter (ADC) . . . . . 266
Direct Memory Access (DMA) . . . . . 266
Interrupt Request (IRQ) . . . . . 270
General-Purpose Input/Output (GPIO) . . . . . 274
Watchdog Timer . . . . . 282
Flash . . . . . 284
Packaging . . . . . 286
Ordering Information . . . . . 287
Part Number Suffix Designations . . . . . 292
Index . . . . . 293
Customer Support . . . . . 303

# List of Figures

Figure 1. Z8 Encore! XP F64xx Series Block Diagram . . . . . 3
Figure 2. Z8 Encore! XP F64xx Series in 40-Pin Dual Inline Package (PDIP) . . . . . 8
Figure 3. Z8 Encore! XP F64xx Series in 44-Pin Plastic Leaded Chip Carrier (PLCC) 9
Figure 4. Z8 Encore! XP F64xx Series in 44-Pin Low-Profile Quad Flat Package (LQFP) . . . . . 10
Figure 5. Z8 Encore! XP F64xx Series in 64-Pin Low-Profile Quad Flat Package (LQFP) . . . . . 11
Figure 6. Z8 Encore! XP F64xx Series in 68-Pin Plastic Leaded Chip Carrier (PLCC) . . . . . 12
Figure 7. Z8 Encore! XP F64xx Series in 80-Pin Quad Flat Package (QFP) . . . . . 13
Figure 8. Power-On Reset Operation . . . . . 30
Figure 9. Voltage Brown-Out Reset Operation . . . . . 31
Figure 10. GPIO Port Pin Block Diagram . . . . . 37
Figure 11. Interrupt Controller Block Diagram . . . . . 49
Figure 12. Timer Block Diagram . . . . . 63
Figure 13. UART Block Diagram . . . . . 88
Figure 14. UART Asynchronous Data Format without Parity . . . . . 89
Figure 15. UART Asynchronous Data Format with Parity . . . . . 89
Figure 16. UART Asynchronous Multiprocessor Mode Data Format . . . . . 93
Figure 17. UART Driver Enable Signal Timing (shown with 1 Stop Bit and Parity) . 95
Figure 18. UART Receiver Interrupt Service Routine Flow . . . . . 97
Figure 19. Infrared Data Communication System Block Diagram . . . . . 109
Figure 20. Infrared Data Transmission . . . . . 110
Figure 21. Infrared Data Reception . . . . . 111
Figure 22. SPI Configured as a Master in a Single-Master, Single-Slave System . . . 113
Figure 23. SPI Configured as a Master in a Single-Master, Multiple-Slave System . 114
Figure 24. SPI Configured as a Slave . . . . . 114
Figure 25. SPI Timing When PHASE is 0 . . . . . 117
Figure 26. SPI Timing When PHASE is 1 . . . . . 118
Figure 27. $I^2C$ Controller Block Diagram . . . . . 129
Figure 28. 7-Bit Address Only Transaction Format . . . . . 133
Figure 29. 7-Bit Addressed Slave Data Transfer Format . . . . . 134
Figure 30. 10-Bit Address Only Transaction Format . . . . . 135

Figure 31. 10-Bit Addressed Slave Data Transfer Format . . . . . 136
Figure 32. Receive Data Transfer Format for a 7-Bit Addressed Slave . . . . . 138
Figure 33. Receive Data Format for a 10-Bit Addressed Slave . . . . . 139
Figure 34. Analog-to-Digital Converter Block Diagram . . . . . 162
Figure 35. Flash Memory Arrangement . . . . . 170
Figure 36. On-Chip Debugger Block Diagram . . . . . 183
Figure 37. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #1 of 2 . . . . . 184
Figure 38. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #2 of 2 . . . . . 185
Figure 39. OCD Data Format . . . . . 186
Figure 40. Recommended 20MHz Crystal Oscillator Configuration . . . . . 197
Figure 41. Connecting the On-Chip Oscillator to an External RC Network . . . . . 198
Figure 42. Typical RC Oscillator Frequency as a Function of the External Capacitance with a 45kΩ Resistor . . . . . 199
Figure 43. Typical Active Mode $I_{DD}$ vs. System Clock Frequency . . . . . 205
Figure 44. Maximum Active Mode $I_{DD}$ vs. System Clock Frequency . . . . . 206
Figure 45. Typical Halt Mode $I_{DD}$ vs. System Clock Frequency . . . . . 207
Figure 46. Maximum Halt Mode Icc vs. System Clock Frequency . . . . . 208
Figure 47. Maximum Stop Mode $I_{DD}$ with VBO Enabled vs. Power Supply Voltage . . . . . 209
Figure 48. Maximum Stop Mode $I_{DD}$ with VBO Disabled vs. Power Supply Voltage . . . . . 210
Figure 49. Analog-to-Digital Converter Frequency Response . . . . . 215
Figure 50. Port Input Sample Timing . . . . . 217
Figure 51. GPIO Port Output Timing . . . . . 218
Figure 52. On-Chip Debugger Timing . . . . . 219
Figure 53. SPI Master Mode Timing . . . . . 220
Figure 54. SPI Slave Mode Timing . . . . . 221
Figure 55. $I^2C$ Timing . . . . . 222
Figure 56. UART Timing with CTS . . . . . 223
Figure 57. UART Timing without CTS . . . . . 224
Figure 58. Flags Register . . . . . 243
Figure 59. Op Code Map Cell Description . . . . . 244
Figure 60. First Op Code Map . . . . . 246
Figure 61. Second Op Code Map after 1Fh . . . . . 247

# List of Tables

Table 1. Z8 Encore! XP F64xx Series Part Selection Guide . . . . . 2
Table 2. Z8 Encore! XP F64xx Series Package Options . . . . . 7
Table 3. Signal Descriptions . . . . . 14
Table 4. Pin Characteristics of the Z8 Encore! XP F64xx Series . . . . . 17
Table 5. Z8 Encore! XP F64xx Series Program Memory Maps . . . . . 19
Table 6. Z8 Encore! XP F64xx Series Information Area Map . . . . . 21
Table 7. Z8 Encore! XP F64xx Series Register File Address Map . . . . . 22
Table 8. Reset and Stop Mode Recovery Characteristics and Latency . . . . . 28
Table 9. Reset Sources and Resulting Reset Type . . . . . 29
Table 10. Stop Mode Recovery Sources and Resulting Action . . . . . 33
Table 11. Port Availability by Device and Package Type . . . . . 36
Table 12. Port Alternate Function Mapping . . . . . 38
Table 13. GPIO Port Registers and Subregisters . . . . . 39
Table 14. Port A–H GPIO Address Registers (PxADDR) . . . . . 40
Table 15. Port A–H Control Registers (PxCTL) . . . . . 41
Table 16. Port A–H Data Direction Subregisters . . . . . 41
Table 17. Port A–H Alternate Function Subregisters . . . . . 42
Table 18. Port A–H Output Control Subregisters . . . . . 43
Table 19. Port A–H High Drive Enable Subregisters . . . . . 44
Table 20. Port A–H Stop Mode Recovery Source Enable Subregisters . . . . . 45
Table 21. Port A–H Input Data Registers (PxIN) . . . . . 46
Table 22. Port A–H Output Data Register (PxOUT) . . . . . 46
Table 23. Interrupt Vectors in Order of Priority . . . . . 48
Table 24. Interrupt Request 0 Register (IRQ0) . . . . . 52
Table 25. Interrupt Request 1 Register (IRQ1) . . . . . 53
Table 26. Interrupt Request 2 Register (IRQ2) . . . . . 54
Table 27. IRQ0 Enable and Priority Encoding . . . . . 55
Table 28. IRQ0 Enable High Bit Register (IRQ0ENH) . . . . . 55
Table 29. IRQ0 Enable Low Bit Register (IRQ0ENL) . . . . . 56
Table 30. IRQ1 Enable and Priority Encoding . . . . . 56
Table 31. IRQ1 Enable High Bit Register (IRQ1ENH) . . . . . 57
Table 32. IRQ1 Enable Low Bit Register (IRQ1ENL) . . . . . 57
Table 33. IRQ2 Enable and Priority Encoding . . . . . 58

Table 34. IRQ2 Enable High Bit Register (IRQ2ENH) . . . . . 58
Table 35. IRQ2 Enable Low Bit Register (IRQ2ENL) . . . . . 59
Table 36. Interrupt Edge Select Register (IRQES) . . . . . 60
Table 37. Interrupt Port Select Register (IRQPS) . . . . . 60
Table 38. Interrupt Control Register (IRQCTL) . . . . . 61
Table 39. Timer 0–3 High Byte Register (TxH) . . . . . 73
Table 40. Timer 0–3 Low Byte Register (TxL) . . . . . 73
Table 41. Timer 0–3 Reload High Byte Register (TxRH) . . . . . 74
Table 42. Timer 0–3 Reload Low Byte Register (TxRL) . . . . . 74
Table 43. Timer 0–3 PWM High Byte Register (TxPWMH) . . . . . 75
Table 44. Timer 0–3 PWM Low Byte Register (TxPWML) . . . . . 75
Table 45. Timer 0–3 Control 0 Register (TxCTL0) . . . . . 76
Table 46. Timer 0–3 Control 1 Register (TxCTL1) . . . . . 77
Table 47. Watchdog Timer Approximate Time-Out Delays . . . . . 81
Table 48. Watchdog Timer Control Register (WDTCTL) . . . . . 84
Table 49. Watchdog Timer Events . . . . . 85
Table 50. Watchdog Timer Reload Upper Byte Register (WDTU) . . . . . 85
Table 51. Watchdog Timer Reload High Byte Register (WDTH) . . . . . 86
Table 52. Watchdog Timer Reload Low Byte Register (WDTL) . . . . . 86
Table 53. UART Transmit Data Register (UxTXD) . . . . . 99
Table 54. UART Receive Data Register (UxRXD) . . . . . 99
Table 55. UART Status 0 Register (UxSTAT0) . . . . . 100
Table 56. UART Status 1 Register (UxSTAT1) . . . . . 101
Table 57. UART Control 0 Register (UxCTL0) . . . . . 102
Table 58. UART Control 1 Register (UxCTL1) . . . . . 103
Table 59. UART Address Compare Register (UxADDR) . . . . . 105
Table 60. UART Baud Rate High Byte Register (UxBRH) . . . . . 106
Table 61. UART Baud Rate Low Byte Register (UxBRL) . . . . . 106
Table 62. UART Baud Rates . . . . . 107
Table 63. SPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation . . . 117
Table 64. SPI Data Register (SPIDATA) . . . . . 122
Table 65. SPI Control Register (SPICTL) . . . . . 122
Table 66. SPI Status Register (SPISTAT) . . . . . 123
Table 67. SPI Mode Register (SPIMODE) . . . . . 125
Table 68. SPI Diagnostic State Register (SPIDST) . . . . . 126
Table 69. SPI Baud Rate High Byte Register (SPIBRH) . . . . . 127

Table 70. SPI Baud Rate Low Byte Register (SPIBRL) . . . . . 127
Table 71. I²C Data Register (I2CDATA) . . . . . 142
Table 72. I2C Status Register (I2CSTAT) . . . . . 142
Table 73. I2C Control Register (I2CCTL) . . . . . 144
Table 74. I²C Baud Rate Low Byte Register (I2CBRL) . . . . . 146
Table 75. I²C Baud Rate High Byte Register (I2CBRH) . . . . . 146
Table 76. I²C Diagnostic State Register (I2CDST) . . . . . 147
Table 77. I²C Diagnostic Control Register (I2CDIAG) . . . . . 149
Table 78. DMAx Control Register (DMAxCTL) . . . . . 153
Table 79. DMAx I/O Address Register (DMAxIO) . . . . . 154
Table 80. DMAx Address High Nibble Register (DMAxH) . . . . . 155
Table 81. DMAx Start/Current Address Low Byte Register (DMAxSTART) . . . . . 156
Table 82. DMAx End Address Low Byte Register (DMAxEND) . . . . . 156
Table 83. DMA_ADC Register File Address Example . . . . . 157
Table 84. DMA_ADC Control Register (DMAACTL) . . . . . 158
Table 85. DMA_ADC Address Register (DMAA_ADDR) . . . . . 158
Table 86. DMA_ADC Status Register (DMAA_STAT) . . . . . 159
Table 87. ADC Control Register (ADCCTL) . . . . . 165
Table 88. ADC Data High Byte Register (ADCD_H) . . . . . 167
Table 89. ADC Data Low Bits Register (ADCD_L) . . . . . 168
Table 90. Flash Memory Configurations . . . . . 169
Table 91. Flash Memory Sector Addresses . . . . . 169
Table 92. Z8 Encore! XP F64xx Series Information Area Map . . . . . 171
Table 93. Flash Control Register (FCTL) . . . . . 176
Table 94. Flash Status Register (FSTAT) . . . . . 177
Table 95. Flash Sector Protect Register (FPROT) . . . . . 178
Table 96. Page Select Register (FPS) . . . . . 178
Table 97. Flash Frequency High Byte Register (FFREQH) . . . . . 179
Table 98. Flash Frequency Low Byte Register (FFREQL) . . . . . 179
Table 99. Flash Option Bits At Flash Memory Address 0000h . . . . . 181
Table 100. Options Bits at Flash Memory Address 0001h . . . . . 182
Table 101. OCD Baud-Rate Limits . . . . . 186
Table 102. On-Chip Debugger Commands . . . . . 189
Table 103. OCD Control Register (OCDCTL) . . . . . 193
Table 104. OCD Status Register (OCDSTAT) . . . . . 194
Table 105. Recommended Crystal Oscillator Specifications (20 MHz Operation) . . . 197

Table 106. Absolute Maximum Ratings . . . . . 200
Table 107. DC Characteristics . . . . . 202
Table 108. Power-On Reset and Voltage Brown-Out Electrical Characteristics and Timing . . . . . 211
Table 109. Reset and Stop Mode Recovery Pin Timing . . . . . 212
Table 110. External RC Oscillator Electrical Characteristics and Timing . . . . . 212
Table 111. Flash Memory Electrical Characteristics and Timing . . . . . 213
Table 112. Watchdog Timer Electrical Characteristics and Timing . . . . . 213
Table 113. Analog-to-Digital Converter Electrical Characteristics and Timing . . . . . 214
Table 114. AC Characteristics . . . . . 216
Table 115. GPIO Port Input Timing . . . . . 217
Table 116. GPIO Port Output Timing . . . . . 218
Table 117. On-Chip Debugger Timing . . . . . 219
Table 118. SPI Master Mode Timing . . . . . 220
Table 119. SPI Slave Mode Timing . . . . . 221
Table 120. $I^2C$ Timing . . . . . 222
Table 121. UART Timing with CTS . . . . . 223
Table 122. UART Timing without CTS . . . . . 224
Table 123. Assembly Language Syntax Example 1 . . . . . 226
Table 124. Assembly Language Syntax Example 2 . . . . . 227
Table 125. Notational Shorthand . . . . . 227
Table 126. Additional Symbols . . . . . 228
Table 127. Condition Codes . . . . . 229
Table 128. Arithmetic Instructions . . . . . 230
Table 129. Bit Manipulation Instructions . . . . . 231
Table 130. Block Transfer Instructions . . . . . 231
Table 131. CPU Control Instructions . . . . . 232
Table 132. Load Instructions . . . . . 232
Table 133. Logical Instructions . . . . . 233
Table 134. Program Control Instructions . . . . . 233
Table 135. Rotate and Shift Instructions . . . . . 234
Table 136. eZ8 CPU Instruction Summary . . . . . 234
Table 137. Op Code Map Abbreviations . . . . . 244
Table 138. Timer 0–3 High Byte Register (TxH) . . . . . 248
Table 139. Timer 0–3 Low Byte Register (TxL) . . . . . 249
Table 140. Timer 0–3 Reload High Byte Register (TxRH) . . . . . 249

Table 141. Timer 0–3 Reload Low Byte Register (TxRL) . . . . . 249
Table 142. Timer 0–3 PWM High Byte Register (TxPWMH) . . . . . 249
Table 143. Timer 0–3 PWM Low Byte Register (TxPWML) . . . . . 250
Table 144. Timer 0–3 Control 0 Register (TxCTL0) . . . . . 250
Table 145. Timer 0–3 Control 1 Register (TxCTL1) . . . . . 250
Table 146. Timer 0–3 High Byte Register (TxH) . . . . . 250
Table 147. Timer 0–3 Low Byte Register (TxL) . . . . . 251
Table 148. Timer 0–3 Reload High Byte Register (TxRH) . . . . . 251
Table 149. Timer 0–3 Reload Low Byte Register (TxRL) . . . . . 251
Table 150. Timer 0–3 PWM High Byte Register (TxPWMH) . . . . . 251
Table 151. Timer 0–3 PWM Low Byte Register (TxPWML) . . . . . 252
Table 152. Timer 0–3 Control 0 Register (TxCTL0) . . . . . 252
Table 153. Timer 0–3 Control 1 Register (TxCTL1) . . . . . 252
Table 154. Timer 0–3 High Byte Register (TxH) . . . . . 252
Table 155. Timer 0–3 Low Byte Register (TxL) . . . . . 253
Table 156. Timer 0–3 Reload High Byte Register (TxRH) . . . . . 253
Table 157. Timer 0–3 Reload Low Byte Register (TxRL) . . . . . 253
Table 158. Timer 0–3 PWM High Byte Register (TxPWMH) . . . . . 253
Table 159. Timer 0–3 PWM Low Byte Register (TxPWML) . . . . . 254
Table 160. Timer 0–3 Control 0 Register (TxCTL0) . . . . . 254
Table 161. Timer 0–3 Control 1 Register (TxCTL1) . . . . . 254
Table 162. Timer 0–3 High Byte Register (TxH) . . . . . 254
Table 163. Timer 0–3 Low Byte Register (TxL) . . . . . 255
Table 164. Timer 0–3 Reload High Byte Register (TxRH) . . . . . 255
Table 165. Timer 0–3 Reload Low Byte Register (TxRL) . . . . . 255
Table 166. Timer 0–3 PWM High Byte Register (TxPWMH) . . . . . 255
Table 167. Timer 0–3 PWM Low Byte Register (TxPWML) . . . . . 256
Table 168. Timer 0–3 Control 0 Register (TxCTL0) . . . . . 256
Table 169. Timer 0–3 Control 1 Register (TxCTL1) . . . . . 256
Table 170. UART Transmit Data Register (UxTXD) . . . . . 257
Table 171. UART Receive Data Register (UxRXD) . . . . . 257
Table 172. UART Status 0 Register (UxSTAT0) . . . . . 257
Table 173. UART Control 0 Register (UxCTL0) . . . . . 257
Table 174. UART Control 1 Register (UxCTL1) . . . . . 258
Table 175. UART Status 1 Register (UxSTAT1) . . . . . 258
Table 176. UART Address Compare Register (UxADDR) . . . . . 258

Table 177. UART Baud Rate High Byte Register (UxBRH) . . . . . 258
Table 178. UART Baud Rate Low Byte Register (UxBRL) . . . . . 259
Table 179. UART Transmit Data Register (UxTXD) . . . . . 259
Table 180. UART Receive Data Register (UxRXD) . . . . . 259
Table 181. UART Status 0 Register (UxSTAT0) . . . . . 259
Table 182. UART Control 0 Register (UxCTL0) . . . . . 260
Table 183. UART Control 1 Register (UxCTL1) . . . . . 260
Table 184. UART Status 1 Register (UxSTAT1) . . . . . 260
Table 185. UART Address Compare Register (UxADDR) . . . . . 260
Table 186. UART Baud Rate High Byte Register (UxBRH) . . . . . 261
Table 187. UART Baud Rate Low Byte Register (UxBRL) . . . . . 261
Table 188. $I^2C$ Data Register (I2CDATA) . . . . . 261
Table 189. $I^2C$ Status Register (I2CSTAT) . . . . . 262
Table 190. $I^2C$ Control Register (I2CCTL) . . . . . 262
Table 191. $I^2C$ Baud Rate High Byte Register (I2CBRH) . . . . . 262
Table 192. $I^2C$ Baud Rate Low Byte Register (I2CBRL) . . . . . 262
Table 193. $I^2C$ Diagnostic State Register (I2CDST) . . . . . 263
Table 194. $I^2C$ Diagnostic Control Register (I2CDIAG) . . . . . 263
Table 195. SPI Data Register (SPIDATA) . . . . . 263
Table 196. SPI Control Register (SPICTL) . . . . . 264
Table 197. SPI Status Register (SPISTAT) . . . . . 264
Table 198. SPI Mode Register (SPIMODE) . . . . . 264
Table 199. SPI Diagnostic State Register (SPIDST) . . . . . 265
Table 200. SPI Baud Rate High Byte Register (SPIBRH) . . . . . 265
Table 201. SPI Baud Rate Low Byte Register (SPIBRL) . . . . . 265
Table 202. ADC Data High Byte Register (ADCD_H) . . . . . 266
Table 203. ADC Data Low Bits Register (ADCD_L) . . . . . 266
Table 204. DMAx Control Register (DMAxCTL) . . . . . 267
Table 205. DMAx I/O Address Register (DMAxIO) . . . . . 267
Table 206. DMAx Address High Nibble Register (DMAxH) . . . . . 267
Table 207. DMAx Start/Current Address Low Byte Register (DMAxSTART) . . . . . 267
Table 208. DMAx End Address Low Byte Register (DMAxEND) . . . . . 268
Table 209. DMAx Control Register (DMAxCTL) . . . . . 268
Table 210. DMAx I/O Address Register (DMAxIO) . . . . . 268
Table 211. DMAx Address High Nibble Register (DMAxH) . . . . . 269
Table 212. DMAx Start/Current Address Low Byte Register (DMAxSTART) . . . . . 269

Table 213. DMAx End Address Low Byte Register (DMAxEND) . . . . . 269
Table 214. DMA_ADC Address Register (DMAA_ADDR) . . . . . 269
Table 215. DMA_ADC Control Register (DMAACTL) . . . . . 270
Table 216. DMA_ADC Status Register (DMAA_STAT) . . . . . 270
Table 217. Interrupt Request 0 Register (IRQ0) . . . . . 270
Table 218. IRQ0 Enable High Bit Register (IRQ0ENH) . . . . . 271
Table 219. IRQ0 Enable Low Bit Register (IRQ0ENL) . . . . . 271
Table 220. Interrupt Request 1 Register (IRQ1) . . . . . 271
Table 221. IRQ1 Enable High Bit Register (IRQ1ENH) . . . . . 271
Table 222. IRQ1 Enable Low Bit Register (IRQ1ENL) . . . . . 272
Table 223. Interrupt Request 2 Register (IRQ2) . . . . . 272
Table 224. IRQ2 Enable High Bit Register (IRQ2ENH) . . . . . 272
Table 225. IRQ2 Enable Low Bit Register (IRQ2ENL) . . . . . 272
Table 226. Interrupt Edge Select Register (IRQES) . . . . . 273
Table 227. Interrupt Port Select Register (IRQPS) . . . . . 273
Table 228. Interrupt Control Register (IRQCTL) . . . . . 273
Table 229. Port A–H GPIO Address Registers (PxADDR) . . . . . 274
Table 230. Port A–H Control Registers (PxCTL) . . . . . 274
Table 231. Port A–H Input Data Registers (PxIN) . . . . . 274
Table 232. Port A–H Output Data Register (PxOUT) . . . . . 275
Table 233. Port A–H GPIO Address Registers (PxADDR) . . . . . 275
Table 234. Port A–H Control Registers (PxCTL) . . . . . 275
Table 235. Port A–H Input Data Registers (PxIN) . . . . . 275
Table 236. Port A–H Output Data Register (PxOUT) . . . . . 276
Table 237. Port A–H GPIO Address Registers (PxADDR) . . . . . 276
Table 238. Port A–H Control Registers (PxCTL) . . . . . 276
Table 239. Port A–H Input Data Registers (PxIN) . . . . . 276
Table 240. Port A–H Output Data Register (PxOUT) . . . . . 277
Table 241. Port A–H GPIO Address Registers (PxADDR) . . . . . 277
Table 242. Port A–H Control Registers (PxCTL) . . . . . 277
Table 243. Port A–H Input Data Registers (PxIN) . . . . . 277
Table 244. Port A–H Output Data Register (PxOUT) . . . . . 278
Table 245. Port A–H GPIO Address Registers (PxADDR) . . . . . 278
Table 246. Port A–H Control Registers (PxCTL) . . . . . 278
Table 247. Port A–H Input Data Registers (PxIN) . . . . . 278
Table 248. Port A–H Output Data Register (PxOUT) . . . . . 279

Table 249. Port A–H GPIO Address Registers (PxADDR) . . . . . 279
Table 250. Port A–H Control Registers (PxCTL) . . . . . 279
Table 251. Port A–H Input Data Registers (PxIN) . . . . . 279
Table 252. Port A–H Output Data Register (PxOUT) . . . . . 280
Table 253. Port A–H GPIO Address Registers (PxADDR) . . . . . 280
Table 254. Port A–H Control Registers (PxCTL) . . . . . 280
Table 255. Port A–H Input Data Registers (PxIN) . . . . . 280
Table 256. Port A–H Output Data Register (PxOUT) . . . . . 281
Table 257. Port A–H GPIO Address Registers (PxADDR) . . . . . 281
Table 258. Port A–H Control Registers (PxCTL) . . . . . 281
Table 259. Port A–H Input Data Registers (PxIN) . . . . . 281
Table 260. Port A–H Output Data Register (PxOUT) . . . . . 282
Table 261. Watchdog Timer Control Register (WDTCTL) . . . . . 282
Table 262. Watchdog Timer Reload Upper Byte Register (WDTU) . . . . . 282
Table 263. Watchdog Timer Reload High Byte Register (WDTH) . . . . . 283
Table 264. Watchdog Timer Reload Low Byte Register (WDTL) . . . . . 283
Table 265. Flash Control Register (FCTL) . . . . . 284
Table 266. Flash Status Register (FSTAT) . . . . . 284
Table 267. Page Select Register (FPS) . . . . . 284
Table 268. Flash Frequency High Byte Register (FFREQH) . . . . . 285
Table 269. Flash Frequency Low Byte Register (FFREQL) . . . . . 285
Table 270. Flash Sector Protect Register (FPROT) . . . . . 285
Table 271. Z8 Encore! XP F64xx Series Ordering Matrix . . . . . 287

# Introduction

Zilog's Z8 Encore! XP F64xx Series MCU family of products are a line of Zilog microcontroller products based upon the 8-bit eZ8 CPU. The Z8 Encore! XP F64xx Series adds 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 Z8 Encore! XP F64xx Series makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

## Features

The features of Z8 Encore! XP F64xx Series include:

- 20MHz eZ8 CPU
- Up to 64KB Flash with in-circuit programming capability
- Up to 4KB register RAM
- 12-channel, 10-bit Analog-to-Digital Converter (ADC)
- Two full-duplex 9-bit UARTs with bus transceiver Driver Enable control
- Inter-integrated circuit ($I^2C$)
- Serial Peripheral Interface (SPI)
- Two Infrared Data Association (IrDA)-compliant infrared encoder/decoders
- Up to four 16-bit timers with capture, compare and PWM capability
- Watchdog Timer (WDT) with internal RC oscillator
- Three-channel DMA
- Up to 60 input/output (I/O) pins
- 24 interrupts with configurable priority
- On-Chip Debugger
- Voltage Brown-Out (VBO) Protection
- Power-On Reset (POR)
- Operating voltage of 3.0V to 3.6V with 5V-tolerant inputs
- 0°C to +70°C, –40°C to +105°C, and –40°C to +125°C operating temperature ranges

# Part Selection Guide

Table 1 identifies the basic features and package styles available for each device within the Z8 Encore! XP product line.

**Table 1. Z8 Encore! XP F64xx Series Part Selection Guide**

| Part Number | Flash (KB) | RAM (KB) | I/O | 16-bit Timers with PWM | ADC Inputs | UARTs with IrDA | $I^2C$ | SPI | 40-/ 44-Pin Package | 64/68-Pin Package | 80-Pin Package |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Z8F1621 | 16 | 2 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F1622 | 16 | 2 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F2421 | 24 | 2 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F2422 | 24 | 2 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F3221 | 32 | 2 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F3222 | 32 | 2 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F4821 | 48 | 4 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F4822 | 48 | 4 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F4823 | 48 | 4 | 60 | 4 | 12 | 2 | 1 | 1 | | | X |
| Z8F6421 | 64 | 4 | 31 | 3 | 8 | 2 | 1 | 1 | X | | |
| Z8F6422 | 64 | 4 | 46 | 4 | 12 | 2 | 1 | 1 | | X | |
| Z8F6423 | 64 | 4 | 60 | 4 | 12 | 2 | 1 | 1 | | | X |

Note: For die form sales, contact your local Zilog Sales Office.

# Block Diagram

Figure 1 displays the architecture of the Z8 Encore! XP F64xx Series.

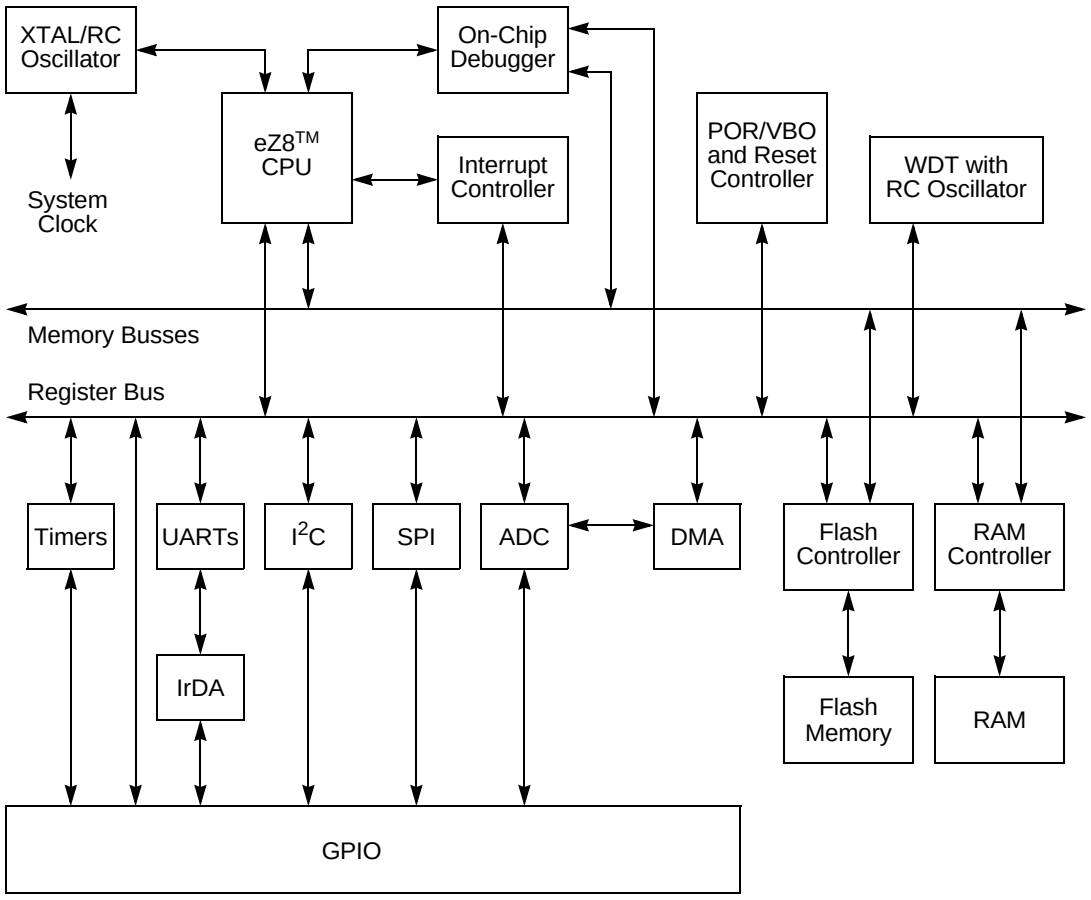



**Figure 1. Z8 Encore! XP F64xx Series Block Diagram**

# CPU and Peripheral Overview

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

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 to 9 clock cycles per instruction

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

## General-Purpose Input/Output

The Z8 Encore! XP F64xx Series features seven 8-bit ports (ports A–G) and one 4-bit port (Port H) for general-purpose input/output (GPIO). Each pin is individually programmable. All ports (except B and H) support 5V-tolerant inputs.

## Flash Controller

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

## 10-Bit Analog-to-Digital Converter

The Analog-to-Digital Converter 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, selectable parity, and an efficient bus transceiver Driver Enable signal for controlling a multitransceiver bus, such as RS-485.

## I²C

The I²C controller makes the Z8 Encore! XP F64xx Series compatible with the I²C protocol. The I²C controller consists of two bidirectional bus lines, a serial data (SDA) line and a serial clock (SCL) line.

## Serial Peripheral Interface

The serial peripheral interface allows the Z8 Encore! XP F64xx Series 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/Compare and PWM modes. Only 3 timers (Timer 0–2) are available in the 44-pin package.

## Interrupt Controller

The Z8 Encore! XP F64xx Series products support up to 24 interrupts. These interrupts consist of 12 internal and 12 GPIO pins. The interrupts have 3 levels of programmable interrupt priority.

## Reset Controller

The Z8 Encore! XP F64xx Series can be reset using the $\overline{\text{RESET}}$ pin, Power-On Reset, Watchdog Timer, Stop Mode exit, or Voltage Brown-Out (**VBO**) warning signal.

## On-Chip Debugger

The Z8 Encore! XP F64xx Series features an integrated On-Chip Debugger. 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 Z8 Encore! XP F64xx Series feature 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

The Z8 Encore! XP F64xx Series product 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 about physical package specifications, see the Packaging chapter on page 286.

## Available Packages

Table 2 identifies the package styles that are available for each device within the Z8 Encore! XP F64xx Series product line.

**Table 2. Z8 Encore! XP F64xx Series Package Options**

| Part Number | 40-Pin PDIP | 44-Pin LQFP | 44-Pin PLCC | 64-Pin LQFP | 68-Pin PLCC | 80-Pin QFP |
|---|---|---|---|---|---|---|
| Z8F1621 | X | X | X | | | |
| Z8F1622 | | | | X | X | |
| Z8F2421 | X | X | X | | | |
| Z8F2422 | | | | X | X | |
| Z8F3221 | X | X | X | | | |
| Z8F3222 | | | | X | X | |
| Z8F4821 | X | X | X | | | |
| Z8F4822 | | | | X | X | |
| Z8F4823 | | | | | | X |
| Z8F6421 | X | X | X | | | |
| Z8F6422 | | | | X | X | |
| Z8F6423 | | | | | | X |

# Pin Configurations

Figures 2 through 7 display the pin configurations for all of the packages available in the Z8 Encore! XP F64xx Series. For signal descriptions, see Table 3 on page 14.

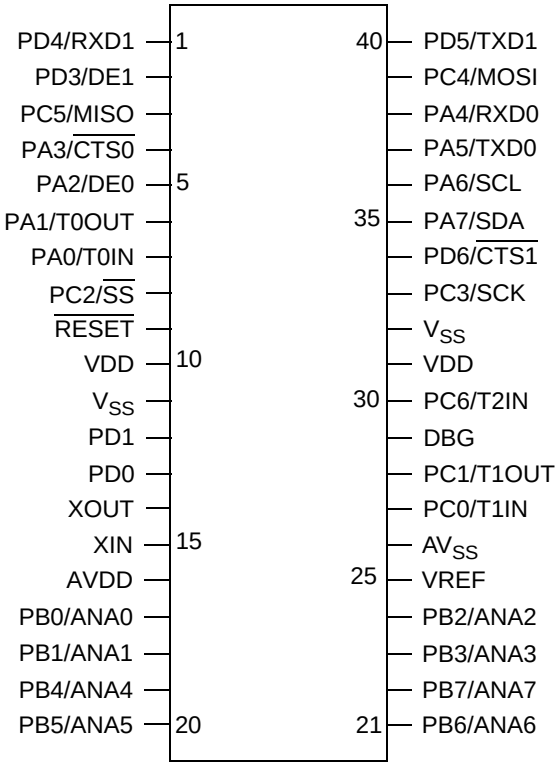



**Figure 2. Z8 Encore! XP F64xx Series in 40-Pin Dual Inline Package (PDIP)**

**Note:** Timer 3 and T2OUT are not supported in the 40-pin PDIP package.

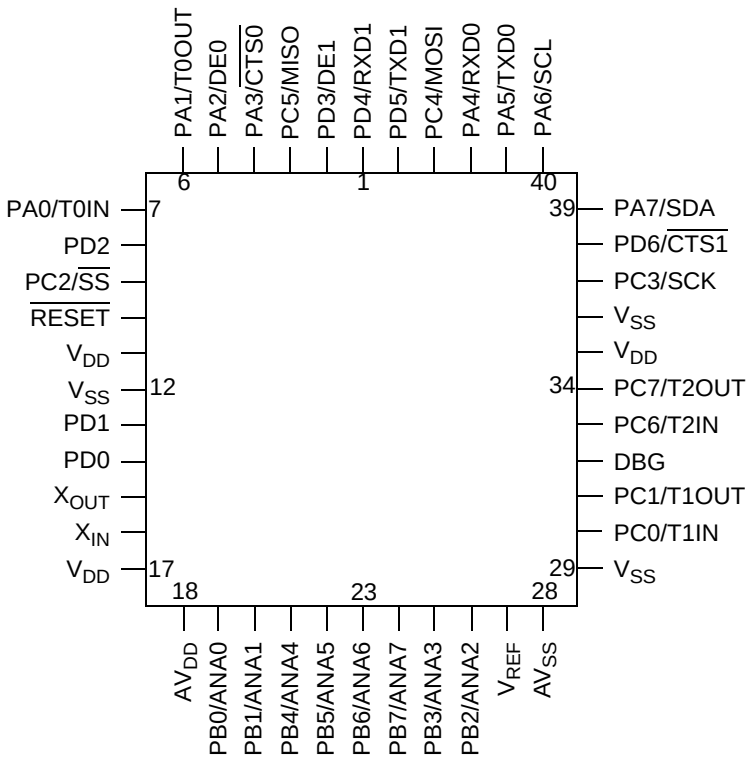



**Figure 3. Z8 Encore! XP F64xx Series in 44-Pin Plastic Leaded Chip Carrier (PLCC)**

**Note:** Timer 3 is not available in the 44-pin PLCC package.

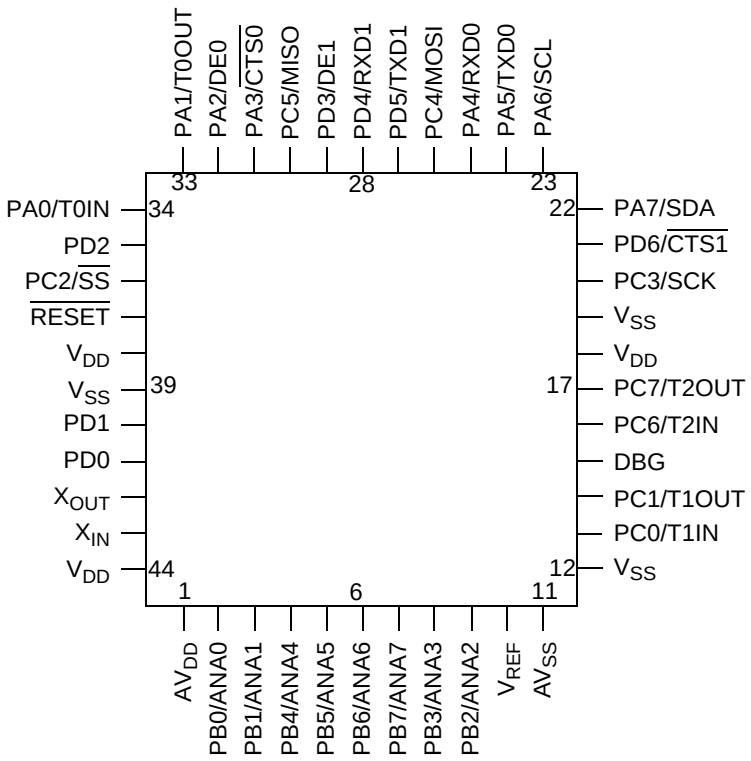



**Figure 4. Z8 Encore! XP F64xx Series in 44-Pin Low-Profile Quad Flat Package (LQFP)**

**Note:** Timer 3 is not available in the 44-pin LQFP package.

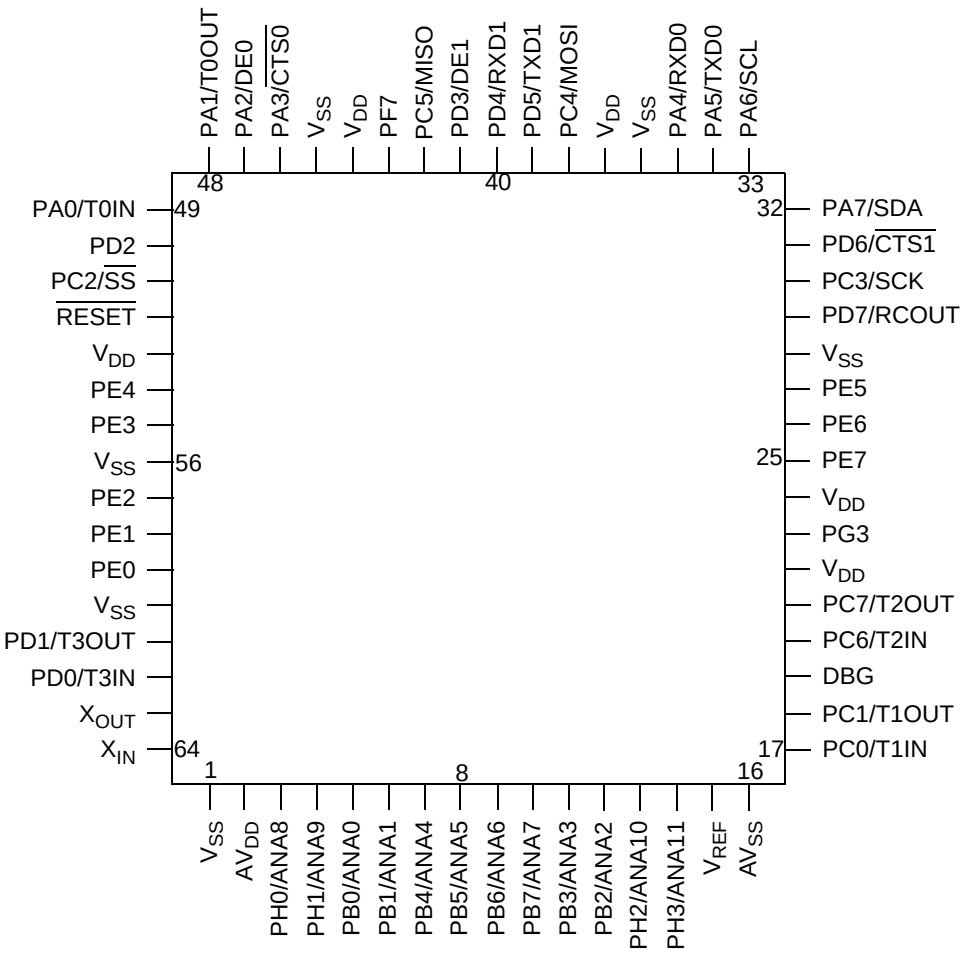



**Figure 5. Z8 Encore! XP F64xx Series in 64-Pin Low-Profile Quad Flat Package (LQFP)**

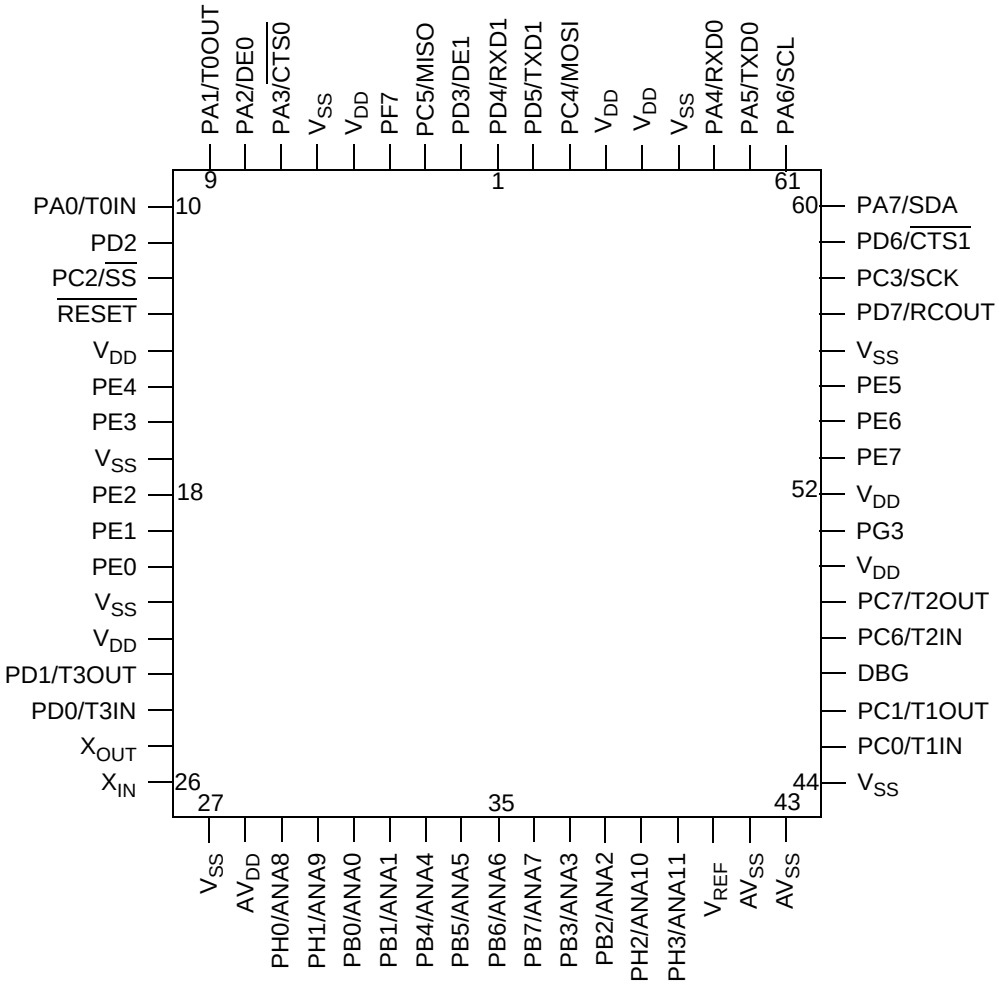



**Figure 6. Z8 Encore! XP F64xx Series in 68-Pin Plastic Leaded Chip Carrier (PLCC)**

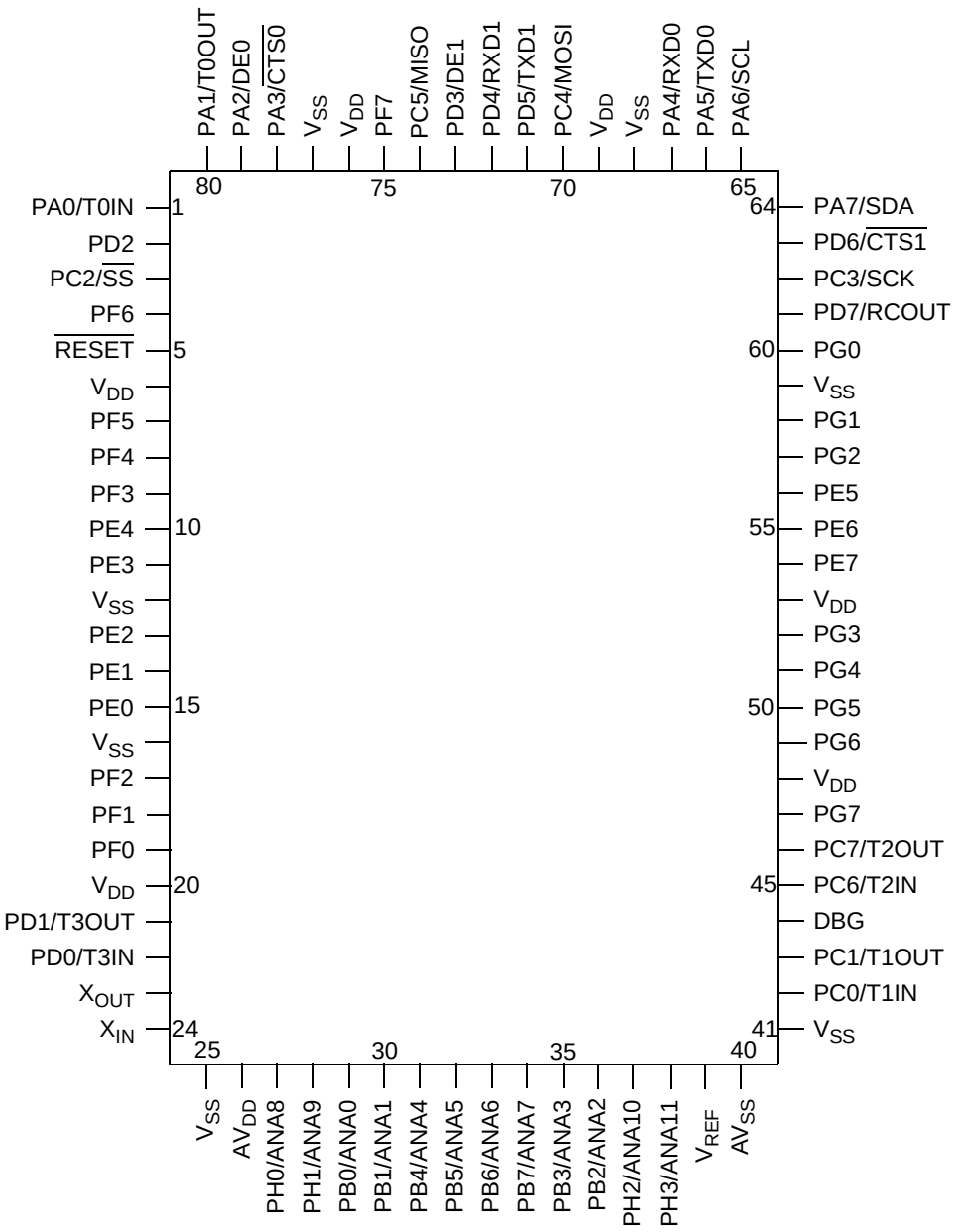



**Figure 7. Z8 Encore! XP F64xx Series in 80-Pin Quad Flat Package (QFP)**

# Signal Descriptions

Table 3 lists the Z8 Encore! XP signals. To determine the available signals for a specific package style, see the Pin Configurations section on page 8.

**Table 3. 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 and support 5V-tolerant inputs. |
| 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. These pins are used for general-purpose I/O and support 5V-tolerant inputs |
| PD[7:0] | I/O | Port D[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5V-tolerant inputs |
| PE[7:0] | I/O | Port E[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5V-tolerant inputs. |
| PF[7:0] | I/O | Port F[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5V-tolerant inputs. |
| PG[7:0] | I/O | Port G[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5V-tolerant inputs. |
| 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 transfers 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! XP F64xx Series is the SPI master, this pin may be configured as the Slave Select output. If the Z8 Encore! XP F64xx Series 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! XP F64xx Series is the SPI master, this pin is an output. If the Z8 Encore! XP F64xx Series is the SPI slave, this pin is an input. It is multiplexed with a general-purpose I/O pin. |

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

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **SPI Controller (continued)** | | |
| 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. |
| **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. |
| DE0/DE1 | O | Driver Enable. This signal allows automatic control of external RS-485 drivers. This signal is approximately the inverse of the Transmit Empty (TXE) bit in the UART Status 0 Register. The DE signal may be used to ensure an external RS-485 driver is enabled when data is transmitted by the UART. |
| **Timers** | | |
| 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. T3OUT is not available in 44-pin package devices. |
| 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 available in 44-pin package devices. |
| **Analog** | | |
| ANA[11:0] | I | Analog Input. These signals are inputs to the ADC. The ADC analog inputs are multiplexed with general-purpose I/O pins. |
| $V_{REF}$ | I | Analog-to-Digital converter reference voltage input. The $V_{REF}$ pin must be left unconnected (or capacitively coupled to analog ground) if the internal voltage reference is selected as the ADC reference voltage. |

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

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **Oscillators** | | |
| $X_{IN}$ | I | External Crystal Input. This is the input pin to the crystal oscillator. A crystal can be connected between it and the $X_{OUT}$ pin to form the oscillator. This signal is usable with external RC networks and an external clock driver. |
| $X_{OUT}$ | O | External Crystal Output. This pin is the output of the crystal oscillator. A crystal can be connected between it and the $X_{IN}$ 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. This pin must be left unconnected when not using a crystal. |
| $RC_{OUT}$ | O | RC Oscillator Output. This signal is the output of the RC oscillator. It is multiplexed with a general-purpose I/O pin. This signal must be left unconnected when not using a crystal. |
| **On-Chip Debugger** | | |
| DBG | I/O | Debug. This pin is the control and data input and output to and from the On-Chip Debugger. This pin is open-drain.<br><br>**Caution:** 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 properly grounded. The DBG pin is open-drain and must have an external pull-up resistor to ensure proper operation. |
| **Reset** | | |
| $\overline{\text{RESET}}$ | I | RESET. Generates a Reset when asserted (driven Low). |
| **Power Supply** | | |
| $V_{DD}$ | I | Power Supply. |
| $AV_{DD}$ | I | Analog Power Supply. |
| $V_{SS}$ | I | Ground. |
| $AV_{SS}$ | I | Analog Ground. |

# Pin Characteristics

Table 4 lists the characteristics for each pin available on the Z8 Encore! XP F64xx Series products and the data is sorted alphabetically by the pin symbol mnemonic.

**Table 4. Pin Characteristics of the Z8 Encore! XP F64xx Series**

| Symbol Mnemonic | Direction | Reset Direction | Active Low or Active High | Tri-State Output | Internal Pull-Up or Pull-Down | Schmitt-Trigger Input | Open-Drain Output |
|---|---|---|---|---|---|---|---|
| $AV_{SS}$ | N/A | N/A | N/A | N/A | No | No | N/A |
| $AV_{DD}$ | N/A | N/A | N/A | N/A | No | No | N/A |
| DBG | I/O | I | N/A | Yes | No | Yes | Yes |
| $V_{SS}$ | 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 |
| 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 |
| RESET | I | I | Low | N/A | Pull-up | Yes | N/A |
| $V_{DD}$ | N/A | N/A | N/A | N/A | No | No | N/A |
| $X_{IN}$ | I | I | N/A | N/A | No | No | N/A |
| $X_{OUT}$ | O | O | N/A | Yes, in Stop Mode | No | No | No |

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

# Address Space

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 consists of the addresses for all memory locations that hold only data

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

## Register File

The register file address space in the Z8 Encore! XP F64xx Series 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 Z8 Encore! XP F64xx Series provide 2KB to 4KB of on-chip RAM depending upon the device. Reading from register file addresses outside the available RAM addresses (and not within the control register address space) returns an undefined value. Writing to these register file addresses produces no effect. To determine the amount of RAM available for the specific Z8 Encore! XP F64xx Series device, see the Part Selection Guide section on page 2.

# Program Memory

The eZ8 CPU supports 64KB of program memory address space. The Z8 Encore! XP F64xx Series contains 16KB to 64KB of on-chip Flash in the program memory address space, depending upon the device. Reading from program memory addresses outside the available Flash memory addresses returns FFh. Writing to these unimplemented program memory addresses produces no effect. Table 5 describes the program memory maps for the Z8 Encore! XP F64xx Series products.

**Table 5. Z8 Encore! XP F64xx Series Program Memory Maps**

| Program Memory Address (Hex) | Function |
|---|---|
| **Z8F162x Products** | |
| 0000-0001 | Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-3FFF | Program Memory |
| **Z8F242x Products** | |
| 0000-0001 | Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-5FFF | Program Memory |
| **Z8F322x Products** | |
| 0000-0001 | Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-7FFF | Program Memory |
| **Z8F482x Products** | |
| 0000-0001 | Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |

Note: *See Table 23 on page 48 for a list of the interrupt vectors.

**Table 5. Z8 Encore! XP F64xx Series Program Memory Maps (Continued)**

| Program Memory Address (Hex) | Function |
|---|---|
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-BFFF | Program Memory |
| **Z8F642x Products** | |
| 0000-0001 | Option Bits |
| 0002-0003 | Reset Vector |
| 0004-0005 | WDT Interrupt Vector |
| 0006-0007 | Illegal Instruction Trap |
| 0008-0037 | Interrupt Vectors* |
| 0038-FFFF | Program Memory |

Note: *See Table 23 on page 48 for a list of the interrupt vectors.

# Data Memory

The Z8 Encore! XP F64xx Series does not use the eZ8 CPU's 64KB data memory address space.

# Information Area

Table 6 describes the Z8 Encore! XP F64xx Series' Information Area. This 512-byte Information Area is accessed by setting bit 7 of the Page Select Register to 1. When access is enabled, the Information Area is mapped into program memory and overlays the 512 bytes at addresses FE00h to FFFFh. When the Information Area access is enabled, execution of the LDC and LDCI instructions from these program memory addresses return the Information Area data rather than the program memory data. Reads of these addresses through the On-Chip Debugger also returns the Information Area data. Execution of code from these addresses continues to correctly use program memory. Access to the Information Area is read-only.

**Table 6. Z8 Encore! XP F64xx Series Information Area Map**

| Program Memory Address (Hex) | Function |
|---|---|
| FE00h–FE3Fh | Reserved |
| FE40h–FE53h | Part Number<br>20-character ASCII alphanumeric code<br>Left-justified and filled with zeros (ASCII Null character) |
| FE54h–FFFFh | Reserved |

# Register File Address Map

Table 7 provides the address map for the register file of the Z8 Encore! XP F64xx Series products. Not all devices and package styles in the Z8 Encore! XP F64xx Series support Timer 3 and all of the GPIO ports. Consider registers for unimplemented peripherals to be reserved.

**Table 7. Z8 Encore! XP F64xx Series 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 | 72 |
| F01 | Timer 0 Low Byte | T0L | 01 | 72 |
| F02 | Timer 0 Reload High Byte | T0RH | FF | 74 |
| F03 | Timer 0 Reload Low Byte | T0RL | FF | 74 |
| F04 | Timer 0 PWM High Byte | T0PWMH | 00 | 75 |
| F05 | Timer 0 PWM Low Byte | T0PWML | 00 | 75 |
| F06 | Timer 0 Control 0 | T0CTL0 | 00 | 76 |
| F07 | Timer 0 Control 1 | T0CTL1 | 00 | 77 |
| **Timer 1** | | | | |
| F08 | Timer 1 High Byte | T1H | 00 | 72 |
| F09 | Timer 1 Low Byte | T1L | 01 | 72 |
| F0A | Timer 1 Reload High Byte | T1RH | FF | 74 |
| F0B | Timer 1 Reload Low Byte | T1RL | FF | 74 |
| F0C | Timer 1 PWM High Byte | T1PWMH | 00 | 75 |
| F0D | Timer 1 PWM Low Byte | T1PWML | 00 | 75 |
| F0E | Timer 1 Control 0 | T1CTL0 | 00 | 76 |
| F0F | Timer 1 Control 1 | T1CTL1 | 00 | 77 |
| **Timer 2** | | | | |
| F10 | Timer 2 High Byte | T2H | 00 | 72 |
| F11 | Timer 2 Low Byte | T2L | 01 | 72 |
| F12 | Timer 2 Reload High Byte | T2RH | FF | 74 |
| F13 | Timer 2 Reload Low Byte | T2RL | FF | 74 |

Note: XX = Undefined.

Table 7. Z8 Encore! XP F64xx Series Register File Address Map (Continued)

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page |
|---|---|---|---|---|
| **Timer 2 (continued)** | | | | |
| F14 | Timer 2 PWM High Byte | T2PWMH | 00 | 75 |
| F15 | Timer 2 PWM Low Byte | T2PWML | 00 | 75 |
| F16 | Timer 2 Control 0 | T2CTL0 | 00 | 76 |
| F17 | Timer 2 Control 1 | T2CTL1 | 00 | 77 |
| **Timer 3 (Unavailable in the 44-Pin Package)** | | | | |
| F18 | Timer 3 High Byte | T3H | 00 | 72 |
| F19 | Timer 3 Low Byte | T3L | 01 | 72 |
| F1A | Timer 3 Reload High Byte | T3RH | FF | 74 |
| F1B | Timer 3 Reload Low Byte | T3RL | FF | 74 |
| F1C | Timer 3 PWM High Byte | T3PWMH | 00 | 75 |
| F1D | Timer 3 PWM Low Byte | T3PWML | 00 | 75 |
| F1E | Timer 3 Control 0 | T3CTL0 | 00 | 76 |
| F1F | Timer 3 Control 1 | T3CTL1 | 00 | 77 |
| 20–3F | Reserved | — | XX | |
| **UART 0** | | | | |
| F40 | UART0 Transmit Data | U0TXD | XX | 98 |
| | UART0 Receive Data | U0RXD | XX | 99 |
| F41 | UART0 Status 0 | U0STAT0 | 0000011Xb | 100 |
| F42 | UART0 Control 0 | U0CTL0 | 00 | 102 |
| F43 | UART0 Control 1 | U0CTL1 | 00 | 102 |
| F44 | UART0 Status 1 | U0STAT1 | 00 | 100 |
| F45 | UART0 Address Compare Register | U0ADDR | 00 | 105 |
| F46 | UART0 Baud Rate High Byte | U0BRH | FF | 105 |
| F47 | UART0 Baud Rate Low Byte | U0BRL | FF | 105 |
| **UART 1** | | | | |
| F48 | UART1 Transmit Data | U1TXD | XX | 98 |
| | UART1 Receive Data | U1RXD | XX | 99 |
| F49 | UART1 Status 0 | U1STAT0 | 0000011Xb | 100 |
| F4A | UART1 Control 0 | U1CTL0 | 00 | 102 |
| F4B | UART1 Control 1 | U1CTL1 | 00 | 102 |
| F4C | UART1 Status 1 | U1STAT1 | 00 | 100 |

Note: XX = Undefined.

**Table 7. Z8 Encore! XP F64xx Series Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page |
|---|---|---|---|---|
| **UART 1 (continued)** | | | | |
| F4D | UART1 Address Compare Register | U1ADDR | 00 | 105 |
| F4E | UART1 Baud Rate High Byte | U1BRH | FF | 105 |
| F4F | UART1 Baud Rate Low Byte | U1BRL | FF | 105 |
| **I²C** | | | | |
| F50 | I²C Data | I2CDATA | 00 | 141 |
| F51 | I²C Status | I2CSTAT | 80 | 142 |
| F52 | I²C Control | I2CCTL | 00 | 144 |
| F53 | I²C Baud Rate High Byte | I2CBRH | FF | 145 |
| F54 | I²C Baud Rate Low Byte | I2CBRL | FF | 145 |
| F55 | I²C Diagnostic State | I2CDST | C0 | 147 |
| F56 | I²C Diagnostic Control | I2CDIAG | 00 | 149 |
| F57–F5F | Reserved | — | XX | |
| **Serial Peripheral Interface (SPI)** | | | | |
| F60 | SPI Data | SPIDATA | XX | 121 |
| F61 | SPI Control | SPICTL | 00 | 122 |
| F62 | SPI Status | SPISTAT | 01 | 123 |
| F63 | SPI Mode | SPIMODE | 00 | 125 |
| F64 | SPI Diagnostic State | SPIDST | 00 | 126 |
| F65 | Reserved | — | XX | |
| F66 | SPI Baud Rate High Byte | SPIBRH | FF | 126 |
| F67 | SPI Baud Rate Low Byte | SPIBRL | FF | 126 |
| F68–F6F | Reserved | — | XX | |
| **Analog-to-Digital Converter** | | | | |
| F70 | ADC Control | ADCCTL | 20 | 165 |
| F71 | Reserved | — | XX | |
| F72 | ADC Data High Byte | ADCD_H | XX | 167 |
| F73 | ADC Data Low Bits | ADCD_L | XX | 168 |
| F74–FAF | Reserved | — | XX | |
| **DMA 0** | | | | |
| FB0 | DMA0 Control | DMA0CTL | 00 | 153 |
| FB1 | DMA0 I/O Address | DMA0IO | XX | 154 |

Note: XX = Undefined.

**Table 7. Z8 Encore! XP F64xx Series Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page |
|---|---|---|---|---|
| **DMA 0 (continued)** | | | | |
| FB2 | DMA0 End/Start Address High Nibble | DMA0H | XX | 155 |
| FB3 | DMA0 Start Address Low Byte | DMA0START | XX | 156 |
| FB4 | DMA0 End Address Low Byte | DMA0END | XX | 156 |
| **DMA 1** | | | | |
| FB8 | DMA1 Control | DMA1CTL | 00 | 153 |
| FB9 | DMA1 I/O Address | DMA1IO | XX | 154 |
| FBA | DMA1 End/Start Address High Nibble | DMA1H | XX | 155 |
| FBB | DMA1 Start Address Low Byte | DMA1START | XX | 156 |
| FBC | DMA1 End Address Low Byte | DMA1END | XX | 156 |
| **DMA ADC** | | | | |
| FBD | DMA_ADC Address | DMAA_ADDR | XX | 157 |
| FBE | DMA_ADC Control | DMAACTL | 00 | 158 |
| FBF | DMA_ADC Status | DMAASTAT | 00 | 159 |
| **Interrupt Controller** | | | | |
| FC0 | Interrupt Request 0 | IRQ0 | 00 | 51 |
| FC1 | IRQ0 Enable High Bit | IRQ0ENH | 00 | 55 |
| FC2 | IRQ0 Enable Low Bit | IRQ0ENL | 00 | 55 |
| FC3 | Interrupt Request 1 | IRQ1 | 00 | 53 |
| FC4 | IRQ1 Enable High Bit | IRQ1ENH | 00 | 56 |
| FC5 | IRQ1 Enable Low Bit | IRQ1ENL | 00 | 56 |
| FC6 | Interrupt Request 2 | IRQ2 | 00 | 54 |
| FC7 | IRQ2 Enable High Bit | IRQ2ENH | 00 | 58 |
| FC8 | IRQ2 Enable Low Bit | IRQ2ENL | 00 | 58 |
| FC9–FCC | Reserved | — | XX | |
| FCD | Interrupt Edge Select | IRQES | 00 | 60 |
| FCE | Interrupt Port Select | IRQPS | 00 | 60 |
| FCF | Interrupt Control | IRQCTL | 00 | 61 |
| **GPIO Port A** | | | | |
| FD0 | Port A Address | PAADDR | 00 | 40 |
| FD1 | Port A Control | PACTL | 00 | 41 |
| FD2 | Port A Input Data | PAIN | XX | 46 |

Note: XX = Undefined.

**Table 7. Z8 Encore! XP F64xx Series Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page |
|---|---|---|---|---|
| **GPIO Port A (continued)** | | | | |
| FD3 | Port A Output Data | PAOUT | 00 | 46 |
| **GPIO Port B** | | | | |
| FD4 | Port B Address | PBADDR | 00 | 40 |
| FD5 | Port B Control | PBCTL | 00 | 41 |
| FD6 | Port B Input Data | PBIN | XX | 46 |
| FD7 | Port B Output Data | PBOUT | 00 | 46 |
| **GPIO Port C** | | | | |
| FD8 | Port C Address | PCADDR | 00 | 40 |
| FD9 | Port C Control | PCCTL | 00 | 41 |
| FDA | Port C Input Data | PCIN | XX | 46 |
| FDB | Port C Output Data | PCOUT | 00 | 46 |
| **GPIO Port D** | | | | |
| FDC | Port D Address | PDADDR | 00 | 40 |
| FDD | Port D Control | PDCTL | 00 | 41 |
| FDE | Port D Input Data | PDIN | XX | 46 |
| FDF | Port D Output Data | PDOUT | 00 | 46 |
| **GPIO Port E** | | | | |
| FE0 | Port E Address | PEADDR | 00 | 40 |
| FE1 | Port E Control | PECTL | 00 | 41 |
| FE2 | Port E Input Data | PEIN | XX | 46 |
| FE3 | Port E Output Data | PEOUT | 00 | 46 |
| **GPIO Port F** | | | | |
| FE4 | Port F Address | PFADDR | 00 | 40 |
| FE5 | Port F Control | PFCTL | 00 | 41 |
| FE6 | Port F Input Data | PFIN | XX | 46 |
| FE7 | Port F Output Data | PFOUT | 00 | 46 |
| **GPIO Port G** | | | | |
| FE8 | Port G Address | PGADDR | 00 | 40 |
| FE9 | Port G Control | PGCTL | 00 | 41 |
| FEA | Port G Input Data | PGIN | XX | 46 |
| FEB | Port G Output Data | PGOUT | 00 | 46 |

Note: XX = Undefined.

**Table 7. Z8 Encore! XP F64xx Series Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page |
|---|---|---|---|---|
| **GPIO Port H** | | | | |
| FEC | Port H Address | PHADDR | 00 | 40 |
| FED | Port H Control | PHCTL | 00 | 41 |
| FEE | Port H Input Data | PHIN | XX | 46 |
| FEF | Port H Output Data | PHOUT | 00 | 46 |
| **Watchdog Timer** | | | | |
| FF0 | Watchdog Timer Control | WDTCTL | XXX00000b | 83 |
| FF1 | Watchdog Timer Reload Upper Byte | WDTU | FF | 85 |
| FF2 | Watchdog Timer Reload High Byte | WDTH | FF | 85 |
| FF3 | Watchdog Timer Reload Low Byte | WDTL | FF | 85 |
| FF4–FF7 | Reserved | — | XX | |
| **Flash Memory Controller** | | | | |
| FF8 | Flash Control | FCTL | 00 | 175 |
| FF8 | Flash Status | FSTAT | 00 | 177 |
| FF9 | Page Select | FPS | 00 | 177 |
| FF9 (if enabled) | Flash Sector Protect | FPROT | 00 | 178 |
| FFA | Flash Programming Frequency High Byte | FFREQH | 00 | 179 |
| FFB | Flash Programming Frequency Low Byte | FFREQL | 00 | 179 |
| **eZ8 CPU** | | | | |
| FFC | Flags | — | XX | Refer to the eZ8 CPU Core User Manual (UM0128) |
| FFD | Register Pointer | RP | XX | |
| FFE | Stack Pointer High Byte | SPH | XX | |
| FFF | Stack Pointer Low Byte | SPL | XX | |

Note: XX = Undefined.

# Reset and Stop Mode Recovery

The Reset Controller within the Z8 Encore! XP F64xx Series controls Reset and Stop Mode Recovery operation. In typical operation, the following events cause a Reset to occur:

- Power-On Reset
- Voltage Brown-Out
- Watchdog 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[0] set to 1)

When the Z8 Encore! XP F64xx Series devices are in Stop Mode, a Stop Mode Recovery is initiated by either of the following events:

- Watchdog Timer time-out
- GPIO port input pin transition on an enabled Stop Mode Recovery source
- DBG pin driven Low

## Reset Types

The Z8 Encore! XP F64xx Series provides two different types of reset operation (system reset and Stop Mode Recovery). The type of Reset is a function of both the current operating mode of the Z8 Encore! XP F64xx Series devices and the source of the Reset. Table 8 lists the types of Reset and their operating characteristics.

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

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

### System Reset

During a system reset, the Z8 Encore! XP F64xx Series devices are held in Reset for 66 cycles of the Watchdog Timer oscillator followed by 16 cycles of the system clock. At the beginning of Reset, all GPIO pins are configured as inputs.

During Reset, the eZ8 CPU and on-chip peripherals are idle; however, the on-chip crystal oscillator and Watchdog Timer oscillator continue to run. The system clock begins operating following the Watchdog 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 9 lists the reset sources as a function of the operating mode. The text following provides more detailed information about the individual Reset sources. A Power-On Reset/Voltage Brown-Out event always takes priority over all other possible reset sources to ensure a full system reset occurs.

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

| Operating Mode | Reset Source | Reset Type |
|---|---|---|
| NORMAL or Halt modes | Power-On Reset/Voltage Brown-Out | system reset |
| | Watchdog Timer time-out when configured for Reset | system reset |
| | $\overline{\text{RESET}}$ pin assertion | system reset |
| | On-Chip Debugger initiated Reset (OCDCTL[0] 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

Each device in the Z8 Encore! XP F64xx Series contains an internal Power-On Reset circuit. The POR circuit monitors the supply voltage and holds the device in the Reset state

until the supply voltage reaches a safe operating level. After the supply voltage exceeds the POR voltage threshold ($V_{POR}$), the POR counter is enabled and counts 66 cycles of the Watchdog Timer oscillator. After the POR counter times out, the XTAL counter is enabled to count a total of 16 system clock pulses. The devices are held in the Reset state until both the POR counter and XTAL counter have timed out. After the Z8 Encore! XP F64xx Series devices exit the Power-On Reset state, the eZ8 CPU fetches the Reset vector. Following Power-On Reset, the POR status bit in the Watchdog Timer Control (WDTCTL) Register is set to 1.

Figure 8 displays Power-On Reset operation. For the POR threshold voltage ($V_{POR}$), see the Electrical Characteristics chapter on page 200.

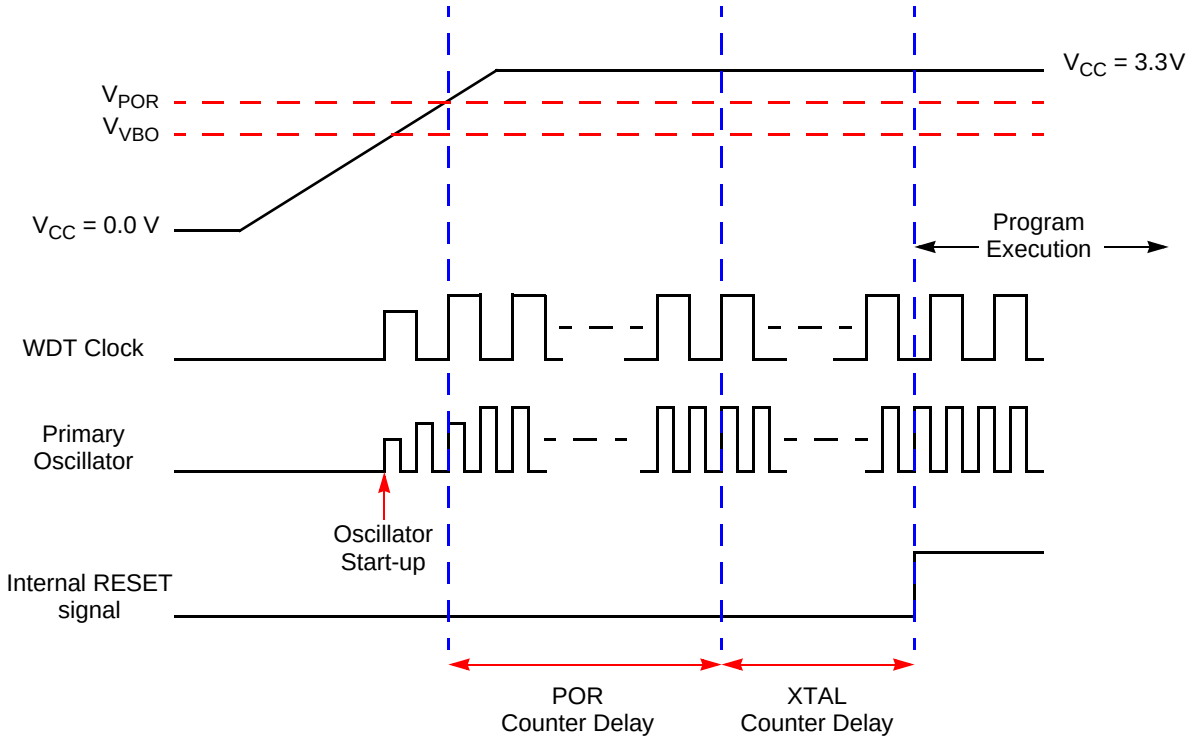



**Figure 8. Power-On Reset Operation**

## Voltage Brown-Out Reset

The devices in the Z8 Encore! XP F64xx Series provide low Voltage Brown-Out protection. The VBO circuit senses when the supply voltage drops to an unsafe level (below the VBO threshold voltage) and forces the device into the Reset state. While the supply voltage remains below the Power-On Reset voltage threshold ($V_{POR}$), the VBO block holds the device in the Reset state.

After the supply voltage again exceeds the Power-On Reset voltage threshold, the devices progress through a full system reset sequence, as described in the Power-On Reset section. Following Power-On Reset, the POR status bit in the Watchdog Timer Control (WDTCTL) Register is set to 1. Figure 9 displays Voltage Brown-Out operation. For the VBO and POR threshold voltages ($V_{VBO}$ and $V_{POR}$), see the Electrical Characteristics chapter on page 200.

The Voltage Brown-Out circuit can be either enabled or disabled during Stop Mode. Operation during Stop Mode is set by the VBO_AO option bit. For information about configuring VBO_AO, see the Option Bits chapter on page 180.

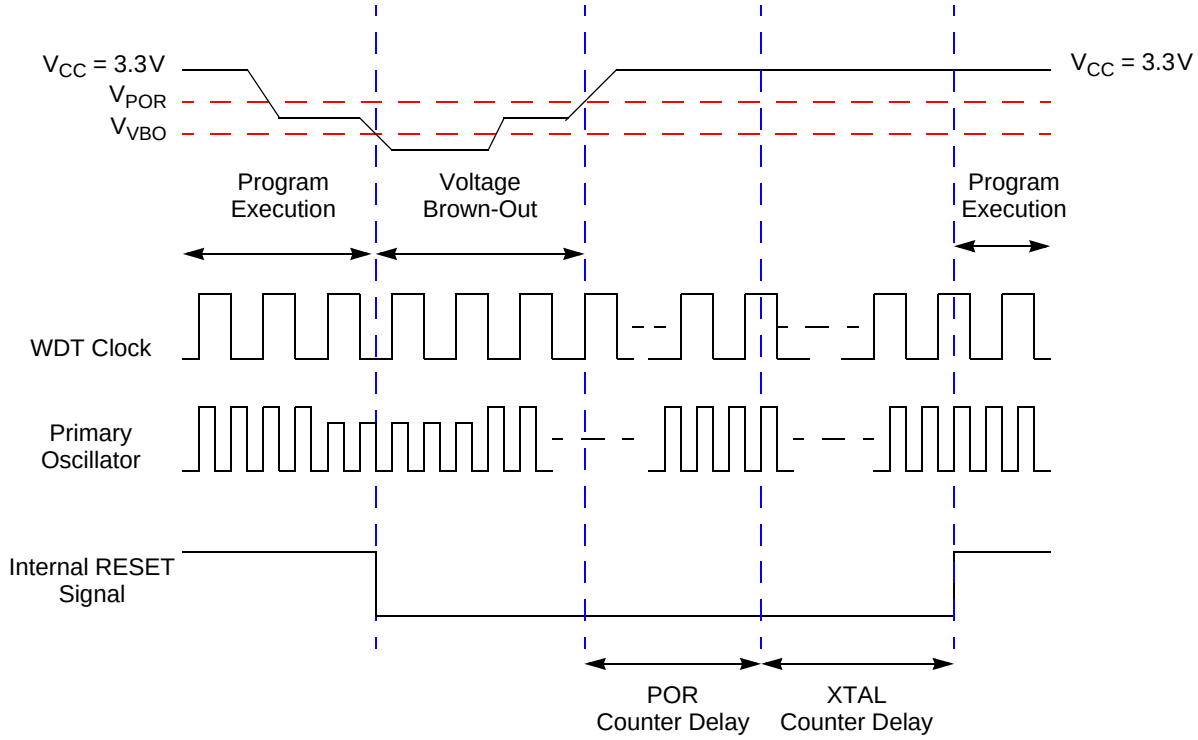



**Figure 9. Voltage Brown-Out Reset Operation**

## Watchdog Timer Reset

If the device is in normal or Halt Mode, the Watchdog Timer can initiate a system reset at time-out if the WDT_RES option bit is set to 1. This capability 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 Watchdog Timer.

### External Pin Reset

The $\overline{\text{RESET}}$ pin has a Schmitt-triggered input, an internal pull-up, an analog filter and a digital filter to reject noise. Once the $\overline{\text{RESET}}$ pin is asserted for at least 4 system clock cycles, the devices progress through the system reset sequence. While the $\overline{\text{RESET}}$ input pin is asserted Low, the Z8 Encore! XP F64xx Series devices continue to be held in the Reset state. If the $\overline{\text{RESET}}$ pin is held Low beyond the system reset time-out, the devices exit the Reset state immediately following $\overline{\text{RESET}}$ pin deassertion. Following a system reset initiated by the external $\overline{\text{RESET}}$ pin, the EXT status bit in the Watchdog Timer Control (WDTCTL) Register is set to 1.

### On-Chip Debugger Initiated Reset

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

## Stop Mode Recovery

Stop Mode is entered by the eZ8 executing a stop instruction. For detailed Stop Mode information, see the Low-Power Modes chapter on page 34. During Stop Mode Recovery, the devices are held in reset for 66 cycles of the Watchdog Timer oscillator followed by 16 cycles of the system clock. Stop Mode Recovery only affects the contents of the Watchdog Timer Control Register. Stop Mode Recovery does not affect any other values in the register file, including the Stack Pointer, Register Pointer, Flags, peripheral control registers, and general-purpose RAM.

The eZ8 CPU fetches the Reset vector at program memory addresses 0002h and 0003h and loads that value into the program counter. Program execution begins at the Reset vector address. Following Stop Mode Recovery, the stop bit in the Watchdog Timer Control Register is set to 1. Table 10 lists the Stop Mode Recovery sources and resulting actions.

**Table 10. Stop Mode Recovery Sources and Resulting Action**

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

## Stop Mode Recovery Using Watchdog Timer Time-Out

If the Watchdog Timer times out during Stop Mode, the device undergoes a Stop Mode Recovery sequence. In the Watchdog Timer Control Register, the WDT and stop bits are set to 1. If the Watchdog Timer is configured to generate an interrupt upon time-out and the Z8 Encore! XP F64xx Series devices are configured to respond to interrupts, the eZ8 CPU services the Watchdog Timer interrupt request following the normal Stop Mode Recovery sequence.

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

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

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

The Z8 Encore! XP F64xx Series products contain power-saving features. The highest level of power reduction is provided by Stop Mode. The next level of power reduction is provided by Halt Mode.

## Stop Mode

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

- Primary crystal oscillator is stopped; the $X_{IN}$ pin is driven High and the $X_{OUT}$ pin is driven Low
- System clock is stopped
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- The Watchdog Timer and its internal RC oscillator continue to operate, if enabled for operation during Stop Mode
- The Voltage Brown-Out protection circuit continues to operate, if enabled for operation in Stop Mode using the associated option bit
- 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 Voltage Brown-Out protection must be disabled, and the Watchdog Timer must be disabled. The devices can be brought out of Stop Mode using Stop Mode Recovery. For more information about Stop Mode Recovery, see the Reset and Stop Mode Recovery chapter on page 28.

**Caution:** Stop Mode must not be used when driving the Z8 Encore! XP F64xx Series devices with an external clock driver source.

# Halt Mode

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

- Primary crystal oscillator is enabled and continues to operate
- System clock is enabled and continues to operate
- eZ8 CPU is stopped
- Program counter stops incrementing
- Watchdog Timer's internal RC oscillator continues to operate
- The Watchdog Timer continues to operate, if enabled
- All other on-chip peripherals continue to operate

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

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

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

# General-Purpose I/O

The Z8 Encore! XP F64xx Series products support a maximum of seven 8-bit ports (ports A–G) and one 4-bit port (Port H) for general-purpose input/output (GPIO) operations. Each port consists of 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. All ports (except B and H) support 5 V-tolerant inputs.

## GPIO Port Availability By Device

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

**Table 11. Port Availability by Device and Package Type**

| Device | Packages | Port A | Port B | Port C | Port D | Port E | Port F | Port G | Port H |
|---|---|---|---|---|---|---|---|---|---|
| Z8X1621 | 40-pin | [7:0] | [7:0] | [7:0] | [6:3,1:0] | – | – | – | – |
| | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | – | – | – | – |
| Z8X1622 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8X2421 | 40-pin | [7:0] | [7:0] | [7:0] | [6:3,1:0] | – | – | – | – |
| | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | – | – | – | – |
| Z8X2422 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8X3221 | 40-pin | [7:0] | [7:0] | [7:0] | [6:3,1:0] | – | – | – | – |
| | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | – | – | – | – |
| Z8X3222 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8X4821 | 40-pin | [7:0] | [7:0] | [7:0] | [6:3,1:0] | – | – | – | – |
| | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | – | – | – | – |
| Z8X4822 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8X4823 | 80-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [3:0] |
| Z8X6421 | 40-pin | [7:0] | [7:0] | [7:0] | [6:3,1:0] | – | – | – | – |
| | 44-pin | [7:0] | [7:0] | [7:0] | [6:0] | – | – | – | – |
| Z8X6422 | 64- and 68-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7] | [3] | [3:0] |
| Z8X6423 | 80-pin | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [7:0] | [3:0] |

# Architecture

Figure 10 displays 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.

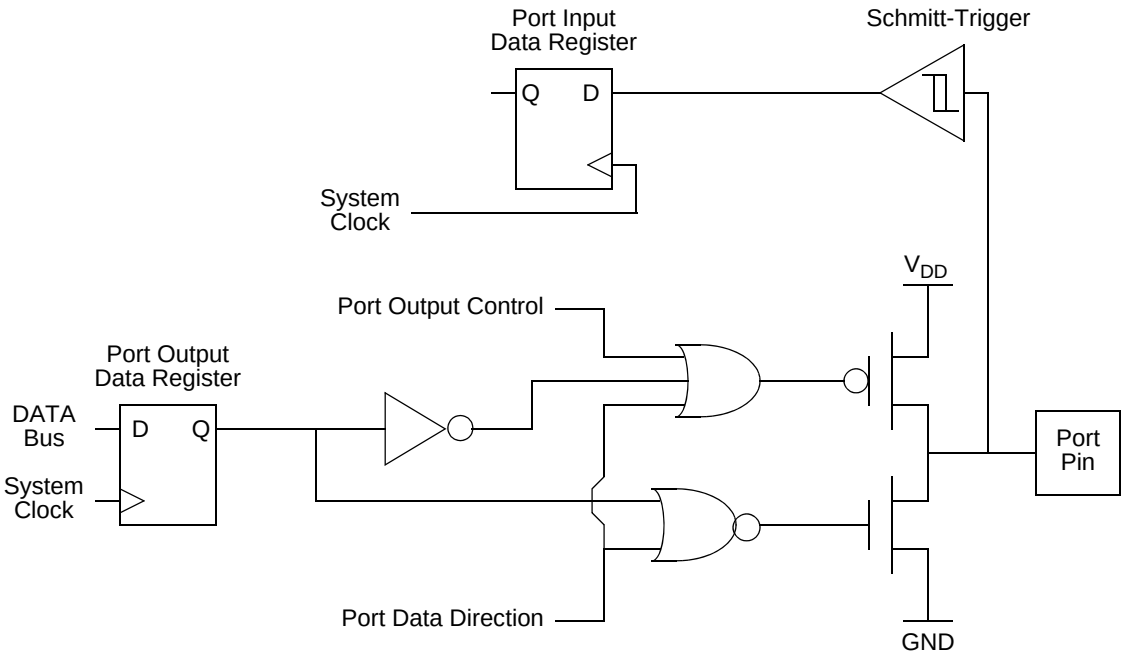



**Figure 10. GPIO Port Pin Block Diagram**

# 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 subregisters configure these pins for either general-purpose I/O or alternate function operation. When a pin is configured for alternate function, control of the port pin direction (input/output) is passed from the Port A–H Data Direction registers to the alternate function assigned to this pin. Table 12 lists the alternate functions associated with each port pin.

**Table 12. Port Alternate Function Mapping**

| Port | Pin | Mnemonic | Alternate Function Description |
|---|---|---|---|
| **Port A** | PA0 | T0IN | Timer 0 Input |
| | PA1 | T0OUT | Timer 0 Output |
| | PA2 | DE0 | UART 0 Driver Enable |
| | PA3 | $\overline{\text{CTS0}}$ | $\overline{\text{UART 0 Clear to Send}}$ |
| | PA4 | RXD0/IRRX0 | UART 0/IrDA 0 Receive Data |
| | PA5 | TXD0/IRTX0 | UART 0/IrDA 0 Transmit Data |
| | PA6 | SCL | $I^2C$ Clock (automatically open-drain) |
| | PA7 | SDA | $I^2C$ Data (automatically open-drain) |
| **Port B** | PB0 | ANA0 | ADC analog input 0 |
| | PB1 | ANA1 | ADC analog input 1 |
| | PB2 | ANA2 | ADC analog input 2 |
| | PB3 | ANA3 | ADC analog input 3 |
| | PB4 | ANA4 | ADC analog input 4 |
| | 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 | $\overline{\text{SS}}$ | $\overline{\text{SPI Slave Select}}$ |
| | 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 |
| **Port D** | PD0 | T3IN | Timer 3 In (unavailable in the 44-pin package) |
| | PD1 | T3OUT | Timer 3 Out (unavailable in the 44-pin package) |
| | PD2 | N/A | No alternate function |
| | PD3 | DE1 | UART 1 Driver Enable |
| | PD4 | RXD1/IRRX1 | UART 1/IrDA 1 Receive Data |
| | PD5 | TXD1/IRTX1 | UART 1/IrDA 1 Transmit Data |
| | PD6 | $\overline{\text{CTS1}}$ | $\overline{\text{UART 1 Clear to Send}}$ |
| | PD7 | RCOUT | Watchdog Timer RC Oscillator Output |
| **Port E** | PE[7:0] | N/A | No alternate functions |

**Table 12. Port Alternate Function Mapping (Continued)**

| Port | Pin | Mnemonic | Alternate Function Description |
|---|---|---|---|
| **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). For more information about interrupts using the GPIO pins, see the Interrupt Controller chapter on page 47.

# GPIO Control Register Definitions

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

**Table 13. GPIO Port Registers and Subregisters**

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

## Port A–H Address Registers

The Port A–H Address registers, shown in Table 14, 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 14. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>PADDR | **Port Address**<br>This port address selects one of the subregisters accessible through the Port A–H Control Registers.<br>00h = No function. Provides some protection against accidental port reconfiguration.<br>01h = Data Direction.<br>02h = Alternate Function.<br>03h = Output Control (Open-Drain).<br>04h = High Drive Enable.<br>05h = Stop Mode Recovery Source Enable.<br>06h–FFh = No function. |

## Port A–H Control Registers

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

**Table 15. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

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

### Port A–H Data Direction Subregisters

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

**Table 16. Port A–H Data Direction Subregisters**

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

Note: If a 01h exists in the Port A–H Address Register, it is accessible through the Port A–H Control Register.

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

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

## Port A–H Alternate Function Subregisters

The Port A–H Alternate Function Subregister, shown in Table 17, 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 subregisters select the alternate functions for the selected pins. To determine the alternate function associated with each port pin, see the GPIO Alternate Functions section on page 37.

**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 17. Port A–H Alternate Function Subregisters**

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

Note: If a 02h exists in the Port A–H Address Register, it is accessible through the Port A–H Control Register.

| Bit | Description |
|---|---|
| [7:0]<br>AFx | **Port Alternate Function Enabled**<br>0 = The port pin is in Normal Mode and the DDx bit in the Port A–H Data Direction Subregister determines the direction of the pin.<br>1 = The alternate function is selected. Port pin operation is controlled by the alternate function. |

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

## Port A–H Output Control Subregisters

The Port A–H Output Control Subregister, shown in Table 18, 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 subregisters to 1 configures the specified port pins for open-drain operation. These subregisters affect the pins directly and, as a result, alternate functions are also affected.

**Table 18. Port A–H Output Control Subregisters**

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

Note: If a 03h exists in the Port A–H Address Register, it is accessible through the Port A–H Control Register.

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

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

## Port A–H High Drive Enable Subregisters

The Port A–H High Drive Enable Subregister, shown in Table 19, 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 subregisters to 1 configures the specified port pins for high-current output drive operation. The Port A–H High Drive Enable Subregister affects the pins directly and, as a result, alternate functions are also affected.

**Table 19. Port A–H High Drive Enable Subregisters**

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

Note: If a 04h exists in the Port A–H Address Register, it is accessible through the Port A–H Control Register.

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

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

## Port A–H Stop Mode Recovery Source Enable Subregisters

The Port A–H Stop Mode Recovery Source Enable Subregister, shown in Table 20, 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 subregisters to 1 configures the specified Port pins as a Stop Mode Recovery source. During Stop Mode, any logic transition on a Port pin enabled as a Stop Mode Recovery source initiates Stop Mode Recovery.

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

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

Note: If a 05h exists in the Port A–H Address Register, it is accessible through the Port A–H Control Register.

| Bit | Description |
|---|---|
| [7:0] PSMRE | **Port Stop Mode Recovery Source Enabled**<br>0 = The port pin is not configured as a Stop Mode Recovery source. Transitions on this pin during Stop Mode do not initiate Stop Mode Recovery.<br>1 = The port pin is configured as a Stop Mode Recovery source. Any logic transition on this pin during Stop Mode initiates Stop Mode Recovery. |

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

## Port A–H Input Data Registers

Reading from the Port A–H Input Data registers, shown in Table 21, returns the sampled values from the corresponding port pins. The Port A–H Input Data registers are read-only.

**Table 21. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

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

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

## Port A–H Output Data Register

The Port A–H Output Data Register, shown in Table 22, writes output data to the pins.

**Table 22. Port A–H Output Data Register (PxOUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>PxOUT | **Port Output Data**<br>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.<br>0 = Drive a logical 0 (Low).<br>1 = Drive a logical 1 (High). High value is not driven if the drain has been disabled by setting the corresponding Port Output Control Register bit to 1. |

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

# Interrupt Controller

The interrupt controller on the Z8 Encore! XP F64xx Series products prioritizes the interrupt requests from the on-chip peripherals and the GPIO port pins. The features of the interrupt controller include:

- 24 unique interrupt vectors:
  - 12 GPIO port pin interrupt sources
  - 12 on-chip peripheral interrupt sources
- Flexible GPIO interrupts
  - Eight selectable rising and falling edge GPIO interrupts
  - Four dual-edge interrupts
- Three levels of individually programmable interrupt priority
- Watchdog 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. For more information about interrupt servicing by the eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download on www.zilog.com.

## Interrupt Vector Listing

Table 23 lists all of the interrupts available in order of priority. The interrupt vector is stored with the most significant byte (MSB) at the even program memory address and the least significant byte (LSB) at the following odd program memory address.

**Table 23. Interrupt Vectors in Order of Priority**

| Priority | Program Memory Vector Address | Interrupt Source |
|---|---|---|
| Highest | 0002h | Reset (not an interrupt) |
| | 0004h | Watchdog Timer (see the Watchdog Timer chapter on page 80) |
| | 0006h | Illegal Instruction Trap (not an interrupt) |
| | 0008h | Timer 2 |
| | 000Ah | Timer 1 |
| | 000Ch | Timer 0 |
| | 000Eh | UART 0 receiver |
| | 0010h | UART 0 transmitter |
| | 0012h | $I^2C$ |
| | 0014h | SPI |
| | 0016h | ADC |
| | 0018h | Port A7 or Port D7, rising or falling input edge |
| | 001Ah | Port A6 or Port D6, rising or falling input edge |
| | 001Ch | Port A5 or Port D5, rising or falling input edge |
| | 001Eh | Port A4 or Port D4, rising or falling input edge |
| | 0020h | Port A3 or Port D3, rising or falling input edge |
| | 0022h | Port A2 or Port D2, rising or falling input edge |
| | 0024h | Port A1 or Port D1, rising or falling input edge |
| | 0026h | Port A0 or Port D0, rising or falling input edge |
| | 0028h | Timer 3 (not available in the 44-pin package) |
| | 002Ah | UART 1 receiver |
| | 002Ch | UART 1 transmitter |
| | 002Eh | DMA |
| | 0030h | Port C3, both input edges |
| | 0032h | Port C2, both input edges |
| | 0034h | Port C1, both input edges |
| Lowest | 0036h | Port C0, both input edges |

# Architecture

Figure 11 displays a block diagram of the interrupt controller.

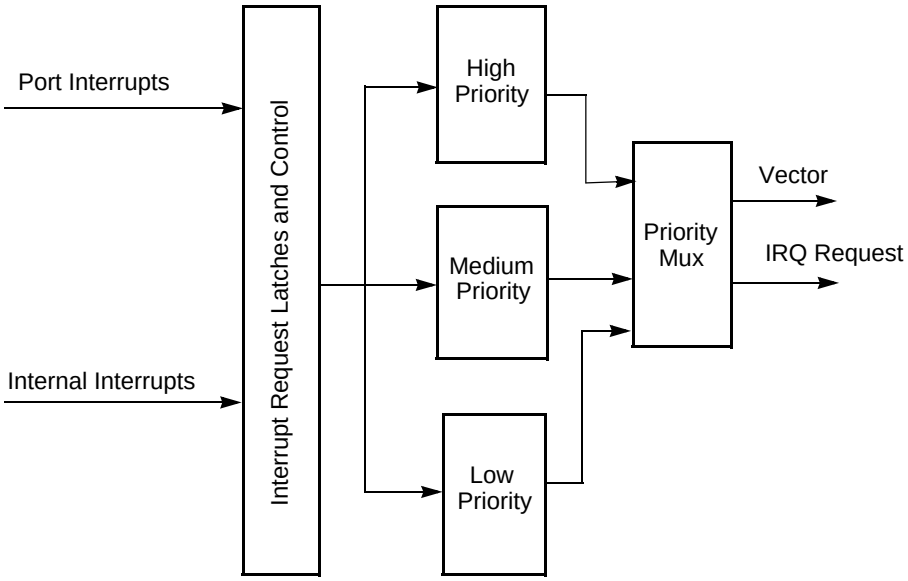



**Figure 11. Interrupt Controller Block Diagram**

# Operation

This section describes the operational aspects of the following functions.

Master Interrupt Enable: see page 49

Interrupt Vectors and Priority: see page 50

Interrupt Assertion: see page 50

Software Interrupt Assertion: see page 51

## 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:

- Executing an Enable Interrupt (EI) instruction
- Executing an Return from Interrupt (IRET) instruction

- Writing a 1 to the IRQE bit in the Interrupt Control Register

Interrupts are globally disabled by any of the following operations:

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

## Interrupt Vectors and Priority

The interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority, and Level 1 is the lowest priority. If all of the interrupts were enabled with identical interrupt priority (all as Level 2 interrupts, for example), then the interrupt priority would be assigned from highest to lowest, as specified in Table 23. 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 23. Resets, Watchdog Timer interrupts (if enabled), and illegal instruction traps always have highest priority.

## Interrupt Assertion

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

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

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

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

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

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

```
ANDX IRQ0, MASK
```

### Software Interrupt Assertion

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

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

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

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

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

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

```
ORX IRQ0, MASK
```

## Interrupt Control Register Definitions

For all interrupts other than the Watchdog 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, shown in Table 24, 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 24. Interrupt Request 0 Register (IRQ0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T2I | T1I | T0I | U0RXI | U0TXI | I2CI | SPII | ADCI |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC0h | | | | | | | |

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

## Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) Register, shown in 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 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.

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 (IRQPS): see page 60.

**Table 25. Interrupt Request 1 Register (IRQ1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PAD7I | PAD6I | PAD5I | PAD4I | PAD3I | PAD2I | PAD1I | PAD0I |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC3h | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>PADxI | **Port A or Port D Pin *x* Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port A or Port D pin *x*.<br>1 = An interrupt request from GPIO Port A or Port D pin *x* is awaiting service. |

Note: x indicates the specific GPIO Port A or D pin in the range [7:0].

## Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register, shown in Table 26, stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the 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 26. Interrupt Request 2 Register (IRQ2)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T3I | U1RXI | U1TXI | DMAI | PC3I | PC2I | PC1I | PC0I |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC6h | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>T3I | **Timer 3 Interrupt Request**<br>0 = No interrupt request is pending for Timer 3.<br>1 = An interrupt request from Timer 3 is awaiting service. |
| [6]<br>U1RXI | **UART 1 Receive Interrupt Request**<br>0 = No interrupt request is pending for the UART1 receiver.<br>1 = An interrupt request from UART1 receiver is awaiting service. |
| [5]<br>U1TXI | **UART 1 Transmit Interrupt Request**<br>0 = No interrupt request is pending for the UART 1 transmitter.<br>1 = An interrupt request from the UART 1 transmitter is awaiting service. |
| [4]<br>DMAI | **DMA Interrupt Request**<br>0 = No interrupt request is pending for the DMA.<br>1 = An interrupt request from the DMA is awaiting service. |
| [3:0]<br>PCxI | **Port C Pin *x* Interrupt Request**<br>0 = No interrupt request is pending for GPIO Port C pin *x*.<br>1 = An interrupt request from GPIO Port C pin *x* is awaiting service. |

Note: x indicates the specific GPIO Port C pin in the range [3:0].

## IRQ0 Enable High and Low Bit Registers

Table 27 describes the priority control for IRQ0. The IRQ0 Enable High and Low Bit registers, shown in Tables 28 and 29, form a priority-encoded enabling for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register.

**Table 27. IRQ0 Enable and Priority Encoding**

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

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

**Table 28. IRQ0 Enable High Bit Register (IRQ0ENH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T2ENH | T1ENH | T0ENH | U0RENH | U0TENH | I2CENH | SPIENH | ADCENH |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC1h | | | | | | | |

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

**Table 29. IRQ0 Enable Low Bit Register (IRQ0ENL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T2ENL | T1ENL | T0ENL | U0RENL | U0TENL | I2CENL | SPIENL | ADCENL |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC2h | | | | | | | |

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

## IRQ1 Enable High and Low Bit Registers

Table 30 describes the priority control for IRQ1. The IRQ1 Enable High and Low Bit registers, shown in Tables 31 and 32, form a priority-encoded enabling for interrupts in the Interrupt Request 1 Register. Priority is generated by setting bits in each register.

**Table 30. IRQ1 Enable and Priority Encoding**

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

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

**Table 31. IRQ1 Enable High Bit Register (IRQ1ENH)**

| Bit | 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 |
| Address | FC4h | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PADxENH | **Port A or Port D Bit[*x*] Interrupt Request Enable High Bit**<br>To select either Port A or Port D as the interrupt source, see the Interrupt Port Select Register on page 60. |

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

**Table 32. IRQ1 Enable Low Bit Register (IRQ1ENL)**

| Bit | 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 |
| Address | FC5h | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PADxENL | **Port A or Port D Bit[*x*] Interrupt Request Enable Low Bit**<br>To select either Port A or Port D as the interrupt source, see the Interrupt Port Select Register on page 60. |

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

## IRQ2 Enable High and Low Bit Registers

Table 33 describes the priority control for IRQ2. The IRQ2 Enable High and Low Bit registers, shown in Tables 34 and 35, form a priority-encoded enabling for interrupts in the Interrupt Request 2 Register. Priority is generated by setting bits in each register.

**Table 33. IRQ2 Enable and Priority Encoding**

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

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

**Table 34. IRQ2 Enable High Bit Register (IRQ2ENH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T3ENH | U1RENH | U1TENH | DMAENH | C3ENH | C2ENH | C1ENH | C0ENH |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC7h | | | | | | | |

| Bit | Description |
|---|---|
| [7] T3ENH | **Timer 3 Interrupt Request Enable High Bit** |
| [6] U1RENH | **UART 1 Receive Interrupt Request Enable High Bit** |
| [5] U1TENH | **UART 1 Transmit Interrupt Request Enable High Bit** |
| [4] DMAENH | **DMA Interrupt Request Enable High Bit** |
| [3] C3ENH | **Port C3 Interrupt Request Enable High Bit** |
| [2] C2ENH | **Port C2 Interrupt Request Enable High Bit** |
| [1] C1ENH | **Port C1 Interrupt Request Enable High Bit** |
| [0] C0ENH | **Port C0 Interrupt Request Enable High Bit** |

**Table 35. IRQ2 Enable Low Bit Register (IRQ2ENL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T3ENL | U1RENL | U1TENL | DMAENL | C3ENL | C2ENL | C1ENL | C0ENL |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC8h | | | | | | | |

| Bit | Description |
|---|---|
| [7] T3ENL | **Timer 3 Interrupt Request Enable Low Bit** |
| [6] U1RENL | **UART 1 Receive Interrupt Request Enable Low Bit** |
| [5] U1TENL | **UART 1 Transmit Interrupt Request Enable Low Bit** |
| [4] DMAENL | **DMA Interrupt Request Enable Low Bit** |
| [3] C3ENL | **Port C3 Interrupt Request Enable Low Bit** |
| [2] C2ENL | **Port C2 Interrupt Request Enable Low Bit** |
| [1] C1ENL | **Port C1 Interrupt Request Enable Low Bit** |
| [0] C0ENL | **Port C0 Interrupt Request Enable Low Bit** |

## Interrupt Edge Select Register

The Interrupt Edge Select (IRQES) Register, shown in Table 36, 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 36. Interrupt Edge Select Register (IRQES)**

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

| Bit | Description |
|---|---|
| [7:0]<br>IESx | **Interrupt Edge Select *x***<br>The minimum pulse width should be greater than 1 system clock to guarantee capture of the edge triggered interrupt. Shorter pulses may be captured but not guaranteed.<br>0 = An interrupt request is generated on the falling edge of the PA*x*/PD*x* input.<br>1 = An interrupt request is generated on the rising edge of the PA*x*/PD*x* input. |

Note: x indicates specific GPIO port pins in the range [7:0].

## Interrupt Port Select Register

The Port Select (IRQPS) Register, shown in Table 37, 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 37. Interrupt Port Select Register (IRQPS)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PAD7S | PAD6S | PAD5S | PAD4S | PAD3S | PAD2S | PAD1S | PAD0S |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FCEh | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>PADxS | **PA*x*/PD*x* Selection**<br>0 = PA*x* is used for the interrupt for PA*x*/PD*x* interrupt request.<br>1 = PD*x* is used for the interrupt for PA*x*/PD*x* interrupt request. |

Note: x indicates specific GPIO port pins in the range [7:0].

## Interrupt Control Register

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

**Table 38. Interrupt Control Register (IRQCTL)**

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

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

# Timers

The Z8 Encore! XP F64xx Series products contain up to four 16-bit reloadable timers that can be used for timing, event counting or generation of pulse-width modulated signals. The timers' features include:

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

In addition to the timers described in this chapter, the baud rate generators for any unused UART, SPI or $I^2C$ peripherals can also be used to provide basic timing functionality. For information about using the baud rate generators as timers, see the respective serial communication peripheral. Timer 3 is unavailable in the 44-pin package devices.

## Architecture

Figure 12 displays the architecture of the timers.

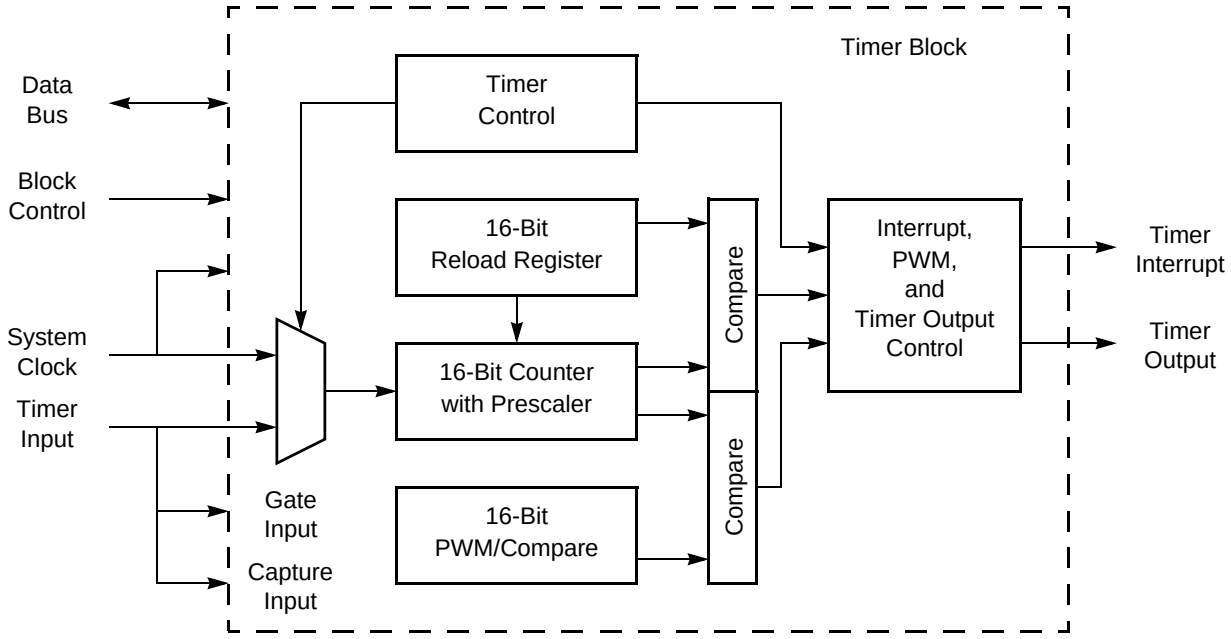



**Figure 12. Timer Block Diagram**

# 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 appropriate to have the timer output make a permanent state change upon a One-Shot time-out, first set the TPOL bit in the Timer Control 1 Register to the start value before beginning One-Shot Mode. Then, after starting the timer, set TPOL to the opposite bit value.

Observe the following procedure for configuring a timer for One-Shot Mode and initiating the count:

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

In One-Shot Mode, the system clock always provides the timer input. The timer period is calculated using the following equation:

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

## Continuous Mode

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

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

1. Write to the Timer Control 1 Register to:

- Disable the timer
- Configure the timer for Continuous Mode
- Set the prescale value
- If using the timer output alternate function, set the initial output level (High or Low)

2. Write to the Timer High and Low Byte registers to set the starting count value (usually 0001h), affecting only the first pass in Continuous Mode. After the first timer reload in Continuous Mode, counting always begins at the reset value of 0001h.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the timer output function, configure the associated GPIO port pin for the timer output alternate function.
6. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In Continuous Mode, the system clock always provides the timer input. The timer period is calculated using the following equation:

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

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

### Counter Mode

In Counter Mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO port pin timer input alternate function. The TPOL bit in the Timer Control 1 Register selects whether the count occurs on the rising edge or the falling edge of the timer input signal. In Counter Mode, the prescaler is disabled.

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

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

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

1. Write to the Timer Control 1 Register to:
   - Disable the timer.
   - Configure the timer for Counter Mode.
   - Select either the rising edge or falling edge of the timer input signal for the count. This also sets the initial logic level (High or Low) for the timer output alternate function. However, the timer output function 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 appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the timer input alternate function.
6. If using the timer output function, configure the associated GPIO port pin for the timer output alternate function.
7. Write to the Timer Control 1 Register to enable the timer.

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

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

### 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 1 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 1 Register is set to 0, the timer output signal begins as a Low (0) and then transitions to a High (1) when the timer value matches the PWM value. The timer output signal returns to a Low (0) after the timer reaches the reload value and is reset to 0001h.

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

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for PWM Mode
   - Set the prescale value
   - Set the initial logic level (High or Low) and PWM High/Low transition for the timer output alternate function
2. Write to the Timer 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 appropriate, 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 1 Register to enable the timer and initiate counting.

The PWM period is calculated using the following equation:

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

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

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

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

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

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

## Capture Mode

In Capture Mode, the current timer count value is recorded when the appropriate external timer input transition occurs. The capture count value is written to the Timer PWM High and Low Byte Registers. The timer input is the system clock. The TPOL bit in the Timer Control 1 Register determines if the capture occurs on a rising edge or a falling edge of the timer input signal. When the capture event occurs, an interrupt is generated and the timer continues counting.

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

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

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Capture Mode
   - Set the prescale value
   - Set the capture edge (rising or falling) for the timer input
2. Write to the Timer 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 the 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 appropriate, 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 1 Register to enable the timer and initiate counting.

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

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

## Compare Mode

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

If the Timer reaches FFFFh, the timer rolls over to 0000h and continue counting.

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

1. Write to the Timer Control 1 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 appropriate
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 appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the timer output function, configure the associated GPIO port pin for the timer output alternate function.
6. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In Compare Mode, the system clock always provides the timer input. The compare time is calculated using the following equation:

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

## Gated Mode

In Gated Mode, the timer counts only when the timer input signal is in its active state (asserted), as determined by the TPOL bit in the Timer Control 1 Register. When the timer input signal is asserted, counting begins. A timer interrupt is generated when the timer input signal is deasserted or a timer reload occurs. To determine if a timer input signal deassertion generated the interrupt, read the associated GPIO input value and compare to the value stored in the TPOL bit.

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

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

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Gated Mode
   - Set the prescale value
2. Write to the Timer High and Low Byte registers to set the starting count value. This only affects the first pass in Gated Mode. After the first timer reset in Gated Mode, counting always begins at the reset value of `0001h`.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the timer input alternate function.
6. Write to the Timer Control 1 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 appropriate transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control 1 Register. The timer input is the system clock.

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

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

Observe the following procedure for configuring a timer for Capture/Compare Mode and initiating the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Capture/Compare Mode
   - Set the prescale value
   - Set the capture edge (rising or falling) for the timer input
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001h).
3. Write to the Timer Reload High and Low Byte registers to set the compare value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the timer input alternate function.
6. Write to the Timer Control 1 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 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

A timer output is a GPIO port pin alternate function. Generally, the timer output is toggled every time the counter is reloaded.

## Timer Control Register Definitions

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

Timer 0–3 High and Low Byte Registers: see page 72

Timer Reload High and Low Byte Registers: see page 74

Timer 0–3 PWM High and Low Byte Registers: see page 75

Timer 0–3 Control 0 Registers: see page 76

Timer 0–3 Control 1 Registers: see page 77

Timers 0–2 are available in all packages. Timer 3 is only available in 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, shown in Tables 39 and 40, contain the current 16-bit timer count value. When the timer is enabled, a read from TxH causes the value in TxL to be stored in a temporary holding register. A read from TMRL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TMRL read 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.

Timer 3 is unavailable in 44-pin packages.

**Table 39. Timer 0–3 High Byte Register (TxH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F00h, F08h, F10h, F18h | | | | | | | |

**Table 40. Timer 0–3 Low Byte Register (TxL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TL | | | | | | | |
| RESET | 0 | | | | | | | 1 |
| R/W | R/W | | | | | | | |
| Address | F01h, F09h, F11h, F19h | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>TH, TL | **Timer High and Low Bytes**<br>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, shown in Tables 41 and 42, store a 16-bit reload value, {TRH[7:0], TRL[7:0]}. Values written to the Timer Reload High Byte Register are stored in a temporary holding register. When a write to the Timer Reload Low Byte Register occurs, the temporary holding register value is written to the Timer High Byte Register. This operation allows simultaneous updates of the 16-bit timer reload value.

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

**Table 41. Timer 0–3 Reload High Byte Register (TxRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F02h, F0Ah, F12h, F1Ah | | | | | | | |

**Table 42. Timer 0–3 Reload Low Byte Register (TxRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F03h, F0Bh, F13h, F1Bh | | | | | | | |

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

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

The Timer 0–3 PWM High and Low Byte (TxPWMH and TxPWML) registers, shown in Tables 43 and 44, are used for Pulse-Width Modulator (PWM) operations. These registers also store the capture values for the Capture and Capture/Compare modes.

**Table 43. Timer 0–3 PWM High Byte Register (TxPWMH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWMH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F04h, F0Ch, F14h, F1Ch | | | | | | | |

**Table 44. Timer 0–3 PWM Low Byte Register (TxPWML)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWML | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F05h, F0Dh, F15h, F1Dh | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] PWMH, PWML | **Pulse-Width Modulator High and Low Bytes**<br>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 1 Register (TxCTL1) Register. The TxPWMH and TxPWML registers also store the 16-bit captured timer value when operating in Capture or Capture/Compare modes. |

## Timer 0–3 Control 0 Registers

The Timer 0–3 Control 0 (TxCTL0) registers, shown in Tables 45 and 46, allow cascading of the timers.

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | CSC | Reserved | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F06h, F0Eh, F16h, F1Eh | | | | | | | |

| Bit | Description |
|---|---|
| [7:5] | **Reserved**<br>These bits are reserved and must be programmed to 000. |
| [4]<br>CSC | **Cascade Timers**<br>0 = Timer input signal comes from the pin.<br>1 = For Timer 0, the input signal is connected to Timer 3 output.<br>For Timer 1, the input signal is connected to the Timer 0 output.<br>For Timer 2, the input signal is connected to the Timer 1 output.<br>For Timer 3, the input signal is connected to the Timer 2 output. |
| [3:0] | **Reserved**<br>These bits are reserved and must be programmed to 0000. |

## Timer 0–3 Control 1 Registers

The Timer 0–3 Control 1 (TxCTL1) registers enable/disable the timers, set the prescaler value, and determine the timer operating mode.

**Table 46. Timer 0–3 Control 1 Register (TxCTL1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | TPOL | PRES | | | TMODE | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F07h, F0Fh, F17h, F1Fh | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TEN | **Timer Enable**<br>0 = Timer is disabled.<br>1 = Timer enabled to count. |
| [6]<br>TPOL | **Timer Input/Output Polarity**<br>Operation of this bit is a function of the current operating mode of the timer.<br>**One-Shot Mode**<br>When the timer is disabled, the timer output signal is set to the value of this bit. When the timer is enabled, the timer output signal is complemented upon timer reload.<br>**Continuous Mode**<br>When the timer is disabled, the timer output signal is set to the value of this bit. When the timer is enabled, the timer output signal is complemented upon timer reload.<br>**Counter Mode**<br>When the timer is disabled, the timer output signal is set to the value of this bit. When the timer is enabled, the timer output signal is complemented upon timer reload.<br>0 = Count occurs on the rising edge of the timer input signal.<br>1 = Count occurs on the falling edge of the timer input signal.<br>**PWM Mode**<br>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.<br>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.<br>**Capture Mode**<br>0 = Count is captured on the rising edge of the timer input signal.<br>1 = Count is captured on the falling edge of the timer input signal.<br>**Compare Mode**<br>When the timer is disabled, the timer output signal is set to the value of this bit. When the timer is enabled, the timer output signal is complemented upon timer reload.<br>**Gated Mode**<br>0 = Timer counts when the timer input signal is High (1) and interrupts are generated on the falling edge of the timer input.<br>1 = Timer counts when the timer input signal is Low (0) and interrupts are generated on the rising edge of the timer input.<br>**Capture/Compare Mode**<br>0 = Counting is started on the first rising edge of the timer input signal. The current count is captured on subsequent rising edges of the timer input signal.<br>1 = Counting is started on the first falling edge of the timer input signal. The current count is captured on subsequent falling edges of the timer input signal.<br><br>**Caution:** When the timer output alternate function TxOUT on a GPIO port pin is enabled, TxOUT will change to whatever state the TPOL bit is in. The timer does not need to be enabled for that to happen. Also, the Port Data Direction Subregister is not needed to be set to output on TxOUT. Changing the TPOL bit with the timer enabled and running does not immediately change the TxOUT. |

| Bit | Description (Continued) |
|---|---|
| [5:3]<br>PRES | **Prescale Value**<br>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 to ensure proper clock division each time the timer is restarted.<br>000 = Divide by 1.<br>001 = Divide by 2.<br>010 = Divide by 4.<br>011 = Divide by 8.<br>100 = Divide by 16.<br>101 = Divide by 32.<br>110 = Divide by 64.<br>111 = Divide by 128. |
| [2:0]<br>TMODE | **TIMER Mode**<br>000 = One-Shot Mode.<br>001 = Continuous Mode.<br>010 = Counter Mode.<br>011 = PWM Mode.<br>100 = Capture Mode.<br>101 = Compare Mode.<br>110 = Gated Mode.<br>111 = Capture/Compare Mode. |

# Watchdog Timer

The Watchdog Timer (WDT) helps protect against corrupt or unreliable software, power faults and other system-level problems which can place the Z8 Encore! XP F64xx Series MCU into unsuitable operating states. The features of the Watchdog Timer include:

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

## Operation

The Watchdog Timer is a retriggerable one-shot timer that resets or interrupts the Z8 Encore! XP F64xx Series devices when the WDT reaches its terminal count. The Watchdog Timer uses its own dedicated on-chip RC oscillator as its clock source. The Watchdog Timer has only two modes of operation: ON and OFF. After it is 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. This WDT_AO bit enables the Watchdog Timer to operate continuously, even if a WDT instruction has not been executed.

The Watchdog 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 calculated using the following equation:

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

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

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

Table 47. Watchdog Timer Approximate Time-Out Delays

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

## Watchdog Timer Refresh

When first enabled, the Watchdog Timer is loaded with the value in the Watchdog Timer Reload registers. The Watchdog 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 Watchdog Timer Reload registers. Counting resumes following the reload operation.

When the Z8 Encore! XP F64xx Series devices are operating in Debug Mode (through the On-Chip Debugger), the Watchdog Timer is continuously refreshed to prevent spurious Watchdog Timer time-outs.

## Watchdog Timer Time-Out Response

The Watchdog Timer times out when the counter reaches 000000h. A time-out of the Watchdog Timer generates either an interrupt or a Reset. The WDT_RES option bit determines the time-out response of the Watchdog Timer. For information about programming of the WDT_RES option bit, see the Option Bits chapter on page 180.

### WDT Interrupt in Normal Operation

If configured to generate an interrupt when a time-out occurs, the Watchdog Timer issues an interrupt request to the interrupt controller and sets the WDT status bit in the Watchdog Timer Control Register. If interrupts are enabled, the eZ8 CPU responds to the interrupt request by fetching the Watchdog Timer interrupt vector and executing code from the vector address. After time-out and interrupt generation, the Watchdog Timer counter rolls over to its maximum value of FFFFFh and continues counting. The Watchdog 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 Z8 Encore! XP F64xx Series devices are in Stop Mode, the Watchdog Timer automatically initiates a Stop Mode Recovery and generates an interrupt request. Both the WDT status bit and the stop bit in the Watchdog Timer Control Register are set to 1 following WDT time-out in Stop

Mode. For more information about Stop Mode Recovery, see the Reset and Stop Mode Recovery chapter on page 28.

If interrupts are enabled, following completion of the Stop Mode Recovery the eZ8 CPU responds to the interrupt request by fetching the Watchdog 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 Watchdog Timer forces the device into the Reset state. The WDT status bit in the Watchdog Timer Control Register is set to 1. For more information about Reset, see the Reset and Stop Mode Recovery chapter on page 28.

### WDT Reset in Stop Mode

If enabled in Stop Mode and configured to generate a Reset when a time-out occurs and the device is in Stop Mode, the Watchdog Timer initiates a Stop Mode Recovery. Both the WDT status bit and the stop bit in the Watchdog Timer Control Register are set to 1 following WDT time-out in Stop Mode. Default operation is for the WDT and its RC oscillator to be enabled during Stop Mode.

### WDT RC Disable in Stop Mode

To minimize power consumption in Stop Mode, the WDT and its RC oscillator can be disabled in Stop Mode. The following sequence configures the WDT to be disabled when the Z8 Encore! XP F64xx Series devices enter Stop Mode following execution of a stop instruction:

1. Write 55h to the Watchdog Timer Control Register (WDTCTL).
2. Write AAh to the Watchdog Timer Control Register (WDTCTL).
3. Write 81h to the Watchdog Timer Control Register (WDTCTL) to configure the WDT and its oscillator to be disabled during Stop Mode. Alternatively, write 00h to the Watchdog Timer Control Register (WDTCTL) as the third step in this sequence to reconfigure the WDT and its oscillator to be enabled during Stop Mode.

This sequence only affects WDT operation in Stop Mode.

## Watchdog Timer Reload Unlock Sequence

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

unlock the Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) for write access.

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

All steps of the Watchdog Timer reload unlock sequence must be written in the sequence described above; 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 Watchdog Timer Reload registers is loaded into the counter when the Watchdog Timer is first enabled and every time a WDT instruction is executed.

# Watchdog Timer Control Register Definitions

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

Watchdog Timer Control Register: see page 83

Watchdog Timer Reload Upper, High and Low Byte Registers: see page 85

## Watchdog Timer Control Register

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

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

**Table 48. Watchdog Timer Control Register (WDTCTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POR | STOP | WDT | EXT | Reserved | | | SM |
| RESET | See Table 49. | | | 0 | | | | |
| R/W | R | | | | | | | |
| Address | FF0h | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>POR | **Power-On Reset Indicator**<br>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. |
| [6]<br>STOP | **Stop Mode Recovery Indicator**<br>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. |
| [5]<br>WDT | **Watchdog Timer Time-Out Indicator**<br>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. |
| [4]<br>EXT | **External Reset Indicator**<br>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. |
| [3:1] | **Reserved**<br>These bits are reserved and must be programmed to 000. |
| [0]<br>SM | **Stop Mode Configuration Indicator**<br>0 = Watchdog Timer and its internal RC oscillator will continue to operate in Stop Mode.<br>1 = Watchdog Timer and its internal RC oscillator will be disabled in Stop Mode. |

Table 49. Watchdog Timer Events

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

## Watchdog Timer Reload Upper, High and Low Byte Registers

The Watchdog Timer Reload Upper, High and Low Byte (WDTU, WDTH, WDTL) registers, shown in Tables 50 through 52, form the 24-bit reload value that is loaded into the Watchdog 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 appropriate reload value. Reading from these registers returns the current Watchdog Timer count value.

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

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | WDTU | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF1h | | | | | | | |

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

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

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | WDTH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF2h | | | | | | | |
| Note: *R/W = Read returns the current WDT count value; write sets the appropriate reload value. | | | | | | | | |

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

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | WDTL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF3h | | | | | | | |
| Note: *R/W = Read returns the current WDT count value; write sets the appropriate reload value. | | | | | | | | |

| Bit | Description |
|---|---|
| [7:0] WDTL | **WDT Reload Low**<br>Least significant byte, bits[7:0] of the 24-bit WDT reload value. |

# Universal Asynchronous Receiver/ Transmitter

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

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

## Architecture

The UART consists of three primary functional blocks: Transmitter, Receiver and Baud Rate Generator. The UART's transmitter and receiver function independently, but employ the same baud rate and data format. Figure 13 displays the UART architecture.

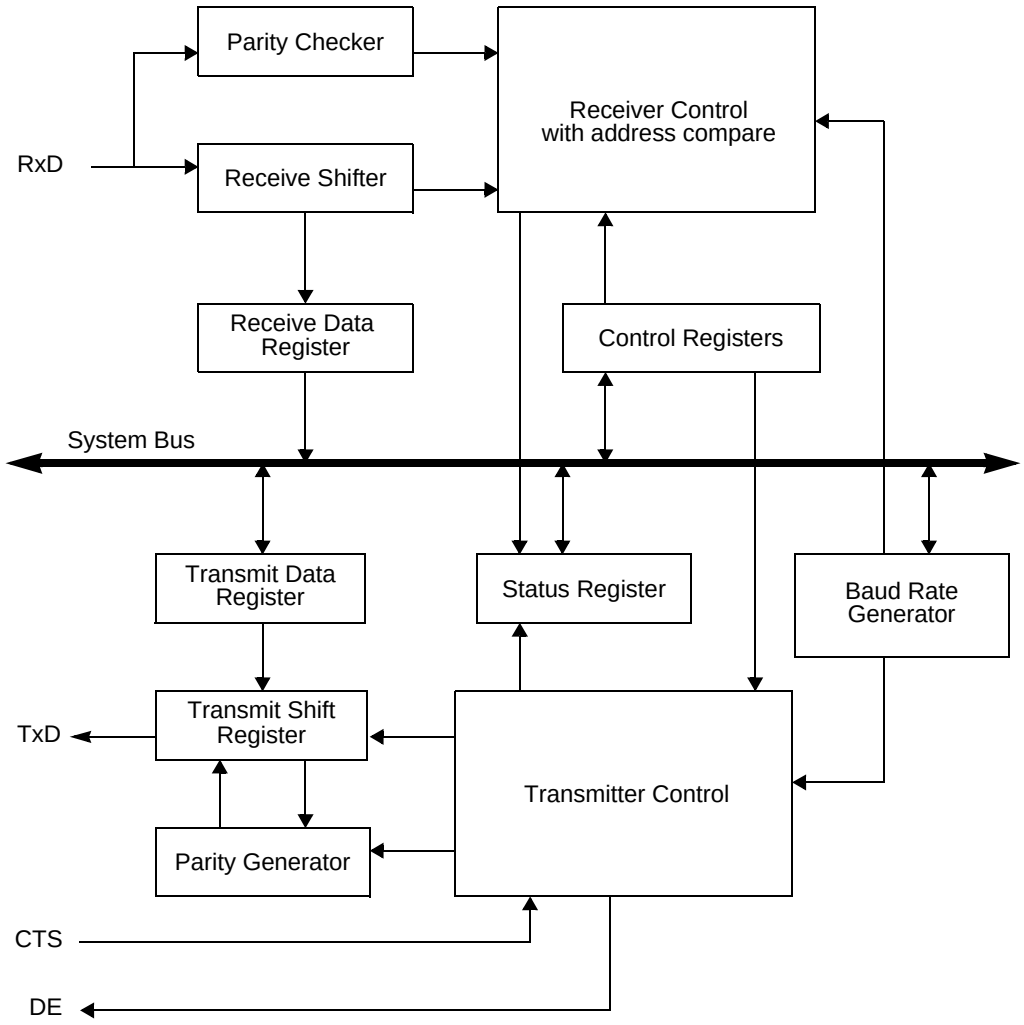



**Figure 13. UART Block Diagram**

# Operation

The UART always transmits and receives data in an 8-bit data format, least significant bit first. An even or odd parity bit 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 14 and 15 display the asynchronous data format employed by the UART without parity and with parity, respectively.

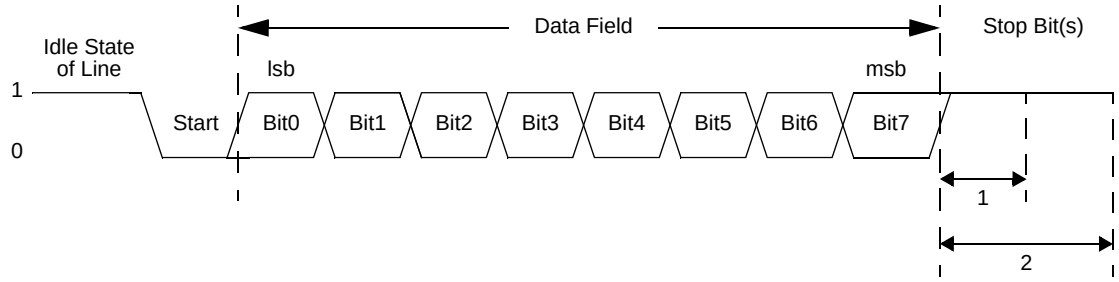



**Figure 14. UART Asynchronous Data Format without Parity**

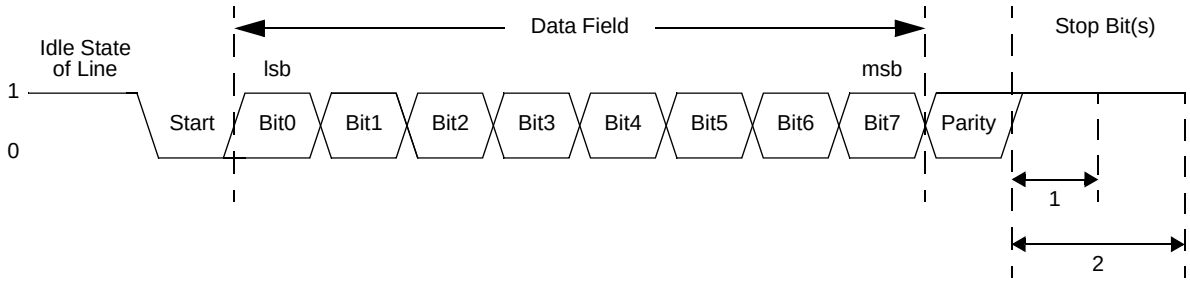



**Figure 15. UART Asynchronous Data Format with Parity**

## Transmitting Data using the Polled Method

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

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

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

5. Check the TDRE bit in the UART Status 0 Register to determine if the Transmit Data Register is empty (indicated by a 1). If empty, continue to Step 6. If the Transmit Data Register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data Register becomes available to receive new data.
6. Write the UART Control 1 Register to select the outgoing address bit.
7. Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte; clear it if sending a data byte.
8. Write the data byte to the UART Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift Register and transmits the data.
9. If appropriate and Multiprocessor Mode is enabled, make any changes to the Multiprocessor Bit Transmitter (MPBT) value.
10. To transmit additional bytes, return to Step 5.

## Transmitting Data using the Interrupt-Driven Method

The UART transmitter interrupt indicates the availability of the Transmit Data Register to accept new data for transmission. Observe the following procedure to configure the UART for interrupt-driven data transmission:

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

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

8. Execute an EI instruction to enable interrupts.

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

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

## Receiving Data using the Polled Method

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

1. Write to the UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Write to the UART Control 1 Register to enable Multiprocessor Mode functions, if appropriate.
4. Write to the UART Control 0 Register to:
   - Set the receive enable bit (REN) to enable the UART for data reception
   - Enable parity, if appropriate and if Multiprocessor Mode is not enabled, and select either even or odd parity
5. Check the RDA bit in the UART Status 0 Register to determine if the Receive Data Register contains a valid data byte (indicated by 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, further actions may be required depending on the Multiprocessor Mode bits MPMD[1:0].
7. Return to Step 5 to receive additional data.

## Receiving Data using the Interrupt-Driven Method

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

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

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

1. Check the UART Status 0 Register to determine the source of the interrupt: error, break, or received data.
2. If the interrupt was caused by data available, read the data from the UART Receive Data Register. If operating in Multiprocessor (9-Bit) Mode, further actions may be required depending on the Multiprocessor Mode bits MPMD[1:0].
3. Clear the UART Receiver interrupt in the applicable Interrupt Request Register.
4. Execute the IRET instruction to return from the interrupt service routine and await more data.

## Clear To Send ($\overline{\text{CTS}}$) Operation

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

## Multiprocessor (9-Bit) Mode

The UART has a Multiprocessor (9-Bit) Mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In Multiprocessor Mode (also referred to as 9-bit mode), the multiprocessor bit (MP) is transmitted immediately following the 8 bits of data and immediately preceding the stop bit(s); the character format is displayed in Figure 16.

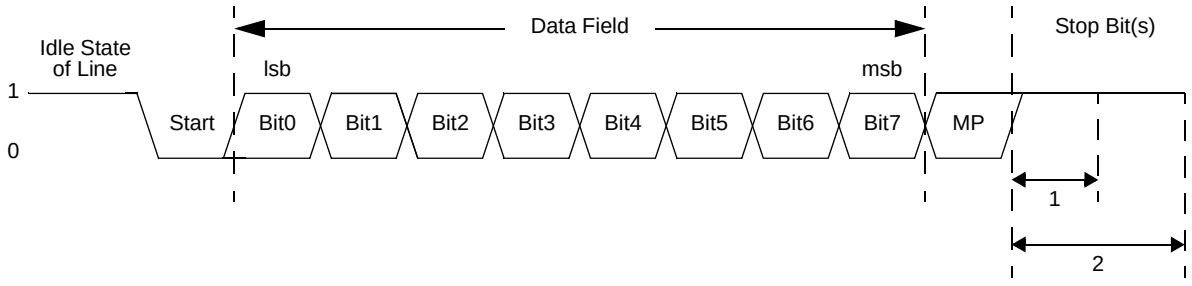



**Figure 16. UART Asynchronous Multiprocessor Mode Data Format**

In Multiprocessor (9-Bit) Mode, the parity bit location (9th bit) becomes the Multiprocessor control bit. The UART Control 1 and Status 1 registers provide Multiprocessor (9-Bit)

Mode control and status information. If an automatic address matching scheme is enabled, the UART Address Compare Register holds the network address of the device.

## Multiprocessor (9-Bit) Mode Receive Interrupts

When Multiprocessor Mode is enabled, the UART only processes frames addressed to it. The determination of whether a frame of data is addressed to the UART can be made in hardware, software or some combination of the two, depending on the multiprocessor configuration bits. In general, the address compare feature reduces the load on the CPU, since it does not need to access the UART when it receives data directed to other devices on the multinode network. The following three Multiprocessor modes are available in hardware:

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

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

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

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

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

## External Driver Enable

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

Driver Enable is an active High signal that envelopes the entire transmitted data frame including parity and stop bits as displayed in Figure 17. The Driver Enable signal asserts when a byte is written to the UART Transmit Data Register. The Driver Enable signal asserts at least one UART bit period and no greater than two UART bit periods before the start bit is transmitted. This timing allows a setup time to enable the transceiver. The Driver Enable signal deasserts one system clock period after the last stop bit is transmitted. This one system clock delay allows both time for data to clear the transceiver before disabling it, as well as the ability to determine if another character follows the current character. In the event of back to back characters (new data must be written to the Transmit Data Register before the previous character is completely transmitted) the DE signal is not deasserted between characters. The DEPOL bit in the UART Control Register 1 sets the polarity of the Driver Enable signal.

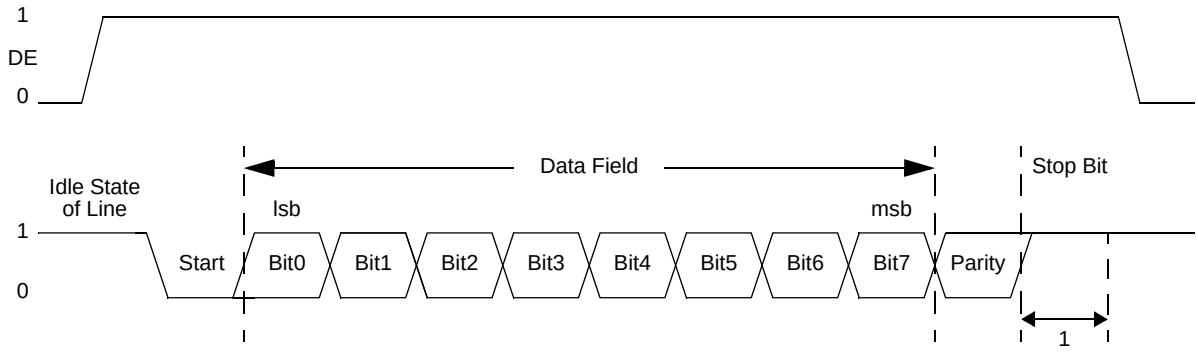



**Figure 17. UART Driver Enable Signal Timing (shown with 1 Stop Bit and Parity)**

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

$$\left(\frac{1}{\text{Baud Rate (Hz)}}\right) \leq \text{DE to Start Bit Setup Time (s)} \leq \left(\frac{2}{\text{Baud Rate (Hz)}}\right)$$

## UART Interrupts

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

### Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register Empty bit (TDRE) is set to 1. This indicates that the transmitter is ready to accept new data for transmission. The TDRE interrupt occurs after the Transmit Shift Register has shifted the first bit of data out. At this point, the Transmit Data Register can be written with the next character to send. This provides 7 bit-periods of latency to load the Transmit Data Register before the Transmit Shift Register completes shifting the current character. Writing to the UART Transmit Data Register clears the TDRE bit to 0.

### Receiver Interrupts

The receiver generates an interrupt when any of the following events 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. The received data interrupt occurs once the receive character has been received and placed in the Receive Data Register. Software must respond to this received data available condition before the next character is completely received to avoid an overrun error.

**Note:** In Multiprocessor Mode (MPEN=1), the receive data interrupts are dependent on the multiprocessor configuration and the most recent address byte.

- A break is received
- An overrun is detected
- A data framing error is detected

### UART Overrun Errors

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

After the valid data has been read, the UART Status 0 Register is updated to indicate the overrun condition (and Break Detect, if applicable). The RDA bit is set to 1 to indicate that the Receive Data Register contains a data byte. However, because the overrun error occurred, this byte may not contain valid data and should be ignored. The BRKD bit indicates if the overrun was caused by a break condition on the line. After reading the status byte indicating an overrun error, the Receive Data Register must be read again to clear the error bits is the UART Status 0 Register. Updates to the Receive Data Register occur only when the next data word is received.

## UART Data and Error Handling Procedure

Figure 18 displays the recommended procedure for use in UART receiver interrupt service routines.

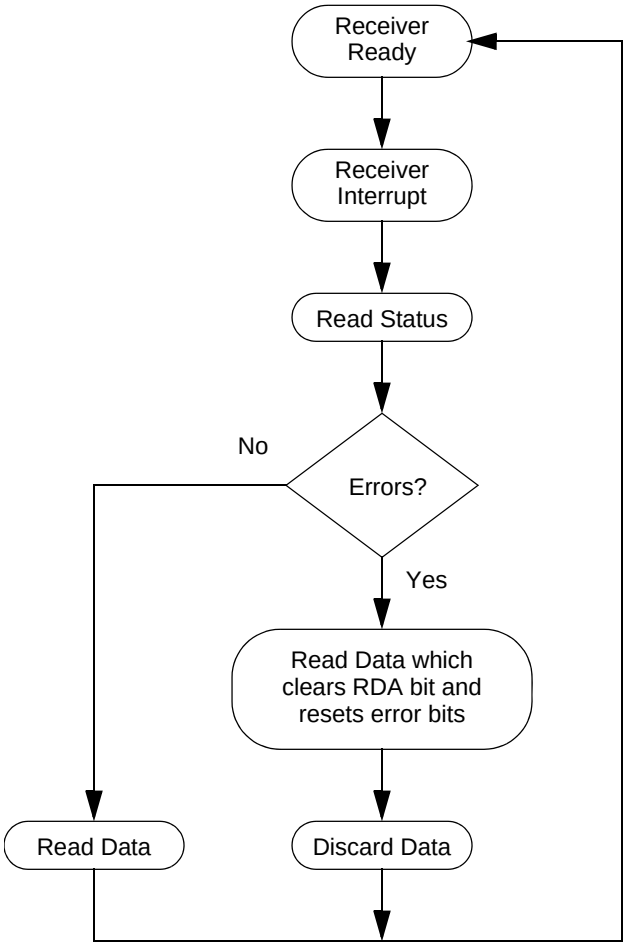



**Figure 18. UART Receiver Interrupt Service Routine Flow**

## Baud Rate Generator Interrupts

If the Baud Rate Generator 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 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 appropriate 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 BRGCTL bit in the UART Control 1 Register to 1.

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

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

# UART Control Register Definitions

The UART control registers support the UART and the associated Infrared Encoder/Decoders. For more information about the infrared operation, see the Infrared Encoder/Decoder chapter on page 109.

## UART Transmit Data Register

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

Table 53. UART Transmit Data Register (U*x*TXD)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TXD | | | | | | | |
| RESET | X | | | | | | | |
| R/W | W | | | | | | | |
| Address | F40h and F48h | | | | | | | |

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

## UART Receive Data Register

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

Table 54. UART Receive Data Register (U*x*RXD)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | RXD | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | F40h and F48h | | | | | | | |

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

## UART Status 0 Register

The UART Status 0 Register, shown in Table 55, identifies the current UART operating configuration and status.

**Table 55. UART Status 0 Register (U*x*STAT0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | RDA | PE | OE | FE | BRKD | TDRE | TXE | CTS |
| RESET | 0 | | | | | | 1 | X |
| R/W | R | | | | | | | |
| Address | F41h and F49h | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>RDA | **Receive Data Available**<br>This bit indicates that the UART Receive Data Register has received data. Reading the UART Receive Data Register clears this bit.<br>0 = The UART Receive Data Register is empty.<br>1 = There is a byte in the UART Receive Data Register. |
| [6]<br>PE | **Parity Error**<br>This bit indicates that a parity error has occurred. Reading the UART Receive Data Register clears this bit.<br>0 = No parity error occurred.<br>1 = A parity error occurred. |
| [5]<br>OE | **Overrun Error**<br>This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the 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.<br>0 = No overrun error occurred.<br>1 = An overrun error occurred. |
| [4]<br>FE | **Framing Error**<br>This bit indicates that a framing error (no stop bit following data reception) was detected. Reading the UART Receive Data Register clears this bit.<br>0 = No framing error occurred.<br>1 = A framing error occurred. |
| [3]<br>BRKD | **Break Detect**<br>This bit indicates that a break occurred. If the data bits, parity/multiprocessor bit, and stop bit(s) are all zeros then this bit is set to 1. Reading the UART Receive Data Register clears this bit.<br>0 = No break occurred.<br>1 = A break occurred. |

| Bit | Description (Continued) |
|---|---|
| [2]<br>TDRE | **Transmitter Data Register Empty**<br>This bit indicates that the UART Transmit Data Register is empty and ready for additional data. Writing to the UART Transmit Data Register resets this bit.<br>0 = Do not write to the UART Transmit Data Register.<br>1 = The UART Transmit Data Register is ready to receive an additional byte to be transmitted. |
| [1]<br>TXE | **Transmitter Empty**<br>This bit indicates that the Transmit Shift Register is empty and character transmission is finished.<br>0 = Data is currently transmitting.<br>1 = Transmission is complete. |
| [0]<br>CTS | **$\overline{\text{CTS}}$ Signal**<br>When this bit is read, it returns the level of the $\overline{\text{CTS}}$ signal. |

## UART Status 1 Register

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

**Table 56. UART Status 1 Register (U*x*STAT1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | | | NEWFRM | MPRX |
| RESET | 0 | | | | | | | |
| R/W | R | | | | R/W | | R | |
| Address | F44h and F4Ch | | | | | | | |

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

## UART Control 0 and Control 1 Registers

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

**Table 57. UART Control 0 Register (U*x*CTL0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | REN | CTSE | PEN | PSEL | SBRK | STOP | LBEN |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F42h and F4Ah | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TEN | **Transmit Enable**<br>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.<br>0 = Transmitter disabled.<br>1 = Transmitter enabled. |
| [6]<br>REN | **Receive Enable**<br>This bit enables or disables the receiver.<br>0 = Receiver disabled.<br>1 = Receiver enabled. |
| [5]<br>CTSE | **CTS Enable**<br>0 = The $\overline{\text{CTS}}$ signal has no effect on the transmitter.<br>1 = The UART recognizes the $\overline{\text{CTS}}$ signal as an enable control from the transmitter. |
| [4]<br>PEN | **Parity Enable**<br>This bit enables or disables parity. Even or odd is determined by the PSEL bit. It is overridden by the MPEN bit.<br>0 = Parity is disabled.<br>1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit. |
| [3]<br>PSEL | **Parity Select**<br>0 = Even parity is transmitted and expected on all received data.<br>1 = Odd parity is transmitted and expected on all received data. |
| [2]<br>SBRK | **Send Break**<br>This bit pauses or breaks data transmission. Sending a break interrupts any transmission in progress, so ensure that the transmitter has finished sending data before setting this bit.<br>0 = No break is sent.<br>1 = The output of the transmitter is zero. |

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

**Table 58. UART Control 1 Register (U*x*CTL1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | MPMD[1] | MPEN | MPMD[0] | MPBT | DEPOL | BRGCTL | RDAIRQ | IREN |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F43h and F4Bh | | | | | | | |

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

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

## UART Address Compare Register

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

**Table 59. UART Address Compare Register (U*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | COMP_ADDR | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F45h and F4Dh | | | | | | | |

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

## UART Baud Rate High and Low Byte Registers

The UART Baud Rate High and Low Byte registers, shown in Tables 60 and 61, combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the UART. 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 appropriate 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 BRGCTL bit in the UART Control 1 Register to 1.

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

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

**Table 60. UART Baud Rate High Byte Register (U*x*BRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F46h and F4Eh | | | | | | | |

**Table 61. UART Baud Rate Low Byte Register (U*x*BRL)**

| Bit7 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F47h and F4Fh | | | | | | | |

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 appropriate baud rate is calculated using the following equation:

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

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

**Table 62. UART Baud Rates**

| 20.0 MHz System Clock | | | | 18.432 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 | 1 | 1250.0 | 0.00 | 1250.0 | 1 | 1152.0 | –7.84% |
| 625.0 | 2 | 625.0 | 0.00 | 625.0 | 2 | 576.0 | –7.84% |
| 250.0 | 5 | 250.0 | 0.00 | 250.0 | 5 | 230.4 | –7.84% |
| 115.2 | 11 | 113.6 | –1.36 | 115.2 | 10 | 115.2 | 0.00 |
| 57.6 | 22 | 56.8 | –1.36 | 57.6 | 20 | 57.6 | 0.00 |
| 38.4 | 33 | 37.9 | –1.36 | 38.4 | 30 | 38.4 | 0.00 |
| 19.2 | 65 | 19.2 | 0.16 | 19.2 | 60 | 19.2 | 0.00 |
| 9.60 | 130 | 9.62 | 0.16 | 9.60 | 120 | 9.60 | 0.00 |
| 4.80 | 260 | 4.81 | 0.16 | 4.80 | 240 | 4.80 | 0.00 |
| 2.40 | 521 | 2.40 | –0.03 | 2.40 | 480 | 2.40 | 0.00 |
| 1.20 | 1042 | 1.20 | –0.03 | 1.20 | 960 | 1.20 | 0.00 |
| 0.60 | 2083 | 0.60 | 0.02 | 0.60 | 1920 | 0.60 | 0.00 |
| 0.30 | 4167 | 0.30 | –0.01 | 0.30 | 3840 | 0.30 | 0.00 |
| **16.667 MHz System Clock** | | | | **11.0592 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 | 1 | 1041.69 | –16.67 | 1250.0 | N/A | N/A | N/A |
| 625.0 | 2 | 520.8 | –16.67 | 625.0 | 1 | 691.2 | 10.59 |
| 250.0 | 4 | 260.4 | 4.17 | 250.0 | 3 | 230.4 | –7.84 |
| 115.2 | 9 | 115.7 | 0.47 | 115.2 | 6 | 115.2 | 0.00 |
| 57.6 | 18 | 57.87 | 0.47 | 57.6 | 12 | 57.6 | 0.00 |
| 38.4 | 27 | 38.6 | 0.47 | 38.4 | 18 | 38.4 | 0.00 |
| 19.2 | 54 | 19.3 | 0.47 | 19.2 | 36 | 19.2 | 0.00 |
| 9.60 | 109 | 9.56 | –0.45 | 9.60 | 72 | 9.60 | 0.00 |
| 4.80 | 217 | 4.80 | –0.83 | 4.80 | 144 | 4.80 | 0.00 |
| 2.40 | 434 | 2.40 | 0.01 | 2.40 | 288 | 2.40 | 0.00 |
| 1.20 | 868 | 1.20 | 0.01 | 1.20 | 576 | 1.20 | 0.00 |
| 0.60 | 1736 | 0.60 | 0.01 | 0.60 | 1152 | 0.60 | 0.00 |
| 0.30 | 3472 | 0.30 | 0.01 | 0.30 | 2304 | 0.30 | 0.00 |

**Table 62. 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 | 0.2 | 0.30 | 1152 | 0.30 | 0.00 |
| **3.579545 MHz System Clock** | | | | **1.8432 MHz System Clock** | | | |
| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) | Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
| 1250.0 | N/A | N/A | N/A | 1250.0 | N/A | N/A | N/A |
| 625.0 | N/A | N/A | N/A | 625.0 | N/A | N/A | N/A |
| 250.0 | 1 | 223.72 | –10.51 | 250.0 | N/A | N/A | N/A |
| 115.2 | 2 | 111.9 | –2.90 | 115.2 | 1 | 115.2 | 0.00 |
| 57.6 | 4 | 55.9 | –2.90 | 57.6 | 2 | 57.6 | 0.00 |
| 38.4 | 6 | 37.3 | –2.90 | 38.4 | 3 | 38.4 | 0.00 |
| 19.2 | 12 | 18.6 | –2.90 | 19.2 | 6 | 19.2 | 0.00 |
| 9.60 | 23 | 9.73 | 1.32 | 9.60 | 12 | 9.60 | 0.00 |
| 4.80 | 47 | 4.76 | –0.83 | 4.80 | 24 | 4.80 | 0.00 |
| 2.40 | 93 | 2.41 | 0.23 | 2.40 | 48 | 2.40 | 0.00 |
| 1.20 | 186 | 1.20 | 0.23 | 1.20 | 96 | 1.20 | 0.00 |
| 0.60 | 373 | 0.60 | –0.04 | 0.60 | 192 | 0.60 | 0.00 |
| 0.30 | 746 | 0.30 | –0.04 | 0.30 | 384 | 0.30 | 0.00 |

# Infrared Encoder/Decoder

The Z8 Encore! XP F64xx Series products contain two fully-functional, high-performance UART-to-infrared encoders/decoders (endecs). Each infrared endec is integrated with an on-chip UART to allow easy communication between the Z8 Encore! XP F64xx Series 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 19 displays the architecture of the infrared endec.

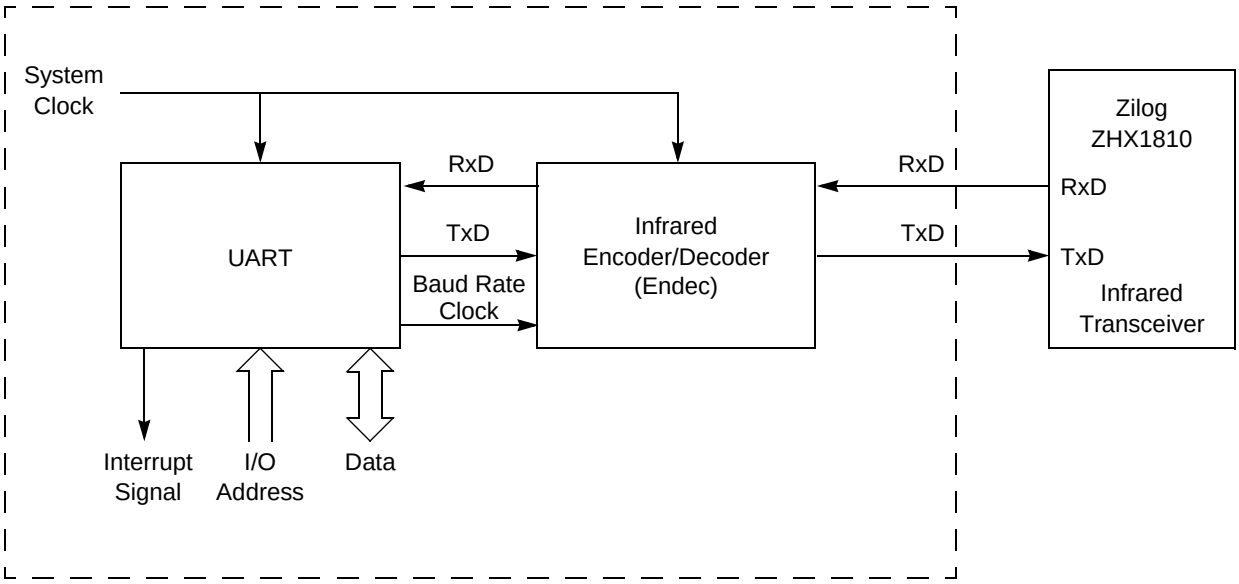



**Figure 19. Infrared Data Communication System Block Diagram**

## 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 115.2KBaud. Higher baud rates are possible, but do not meet IrDA specifications. The UART must be enabled to use the infrared endec. The infrared endec data rate is calculated using the following equation:

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

## 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-clock 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 20 displays IrDA data transmission. When the infrared endec is enabled, the UART's TxD signal is internal to the Z8 Encore! XP F64xx Series products while the IR_TXD signal is output through the TxD pin.

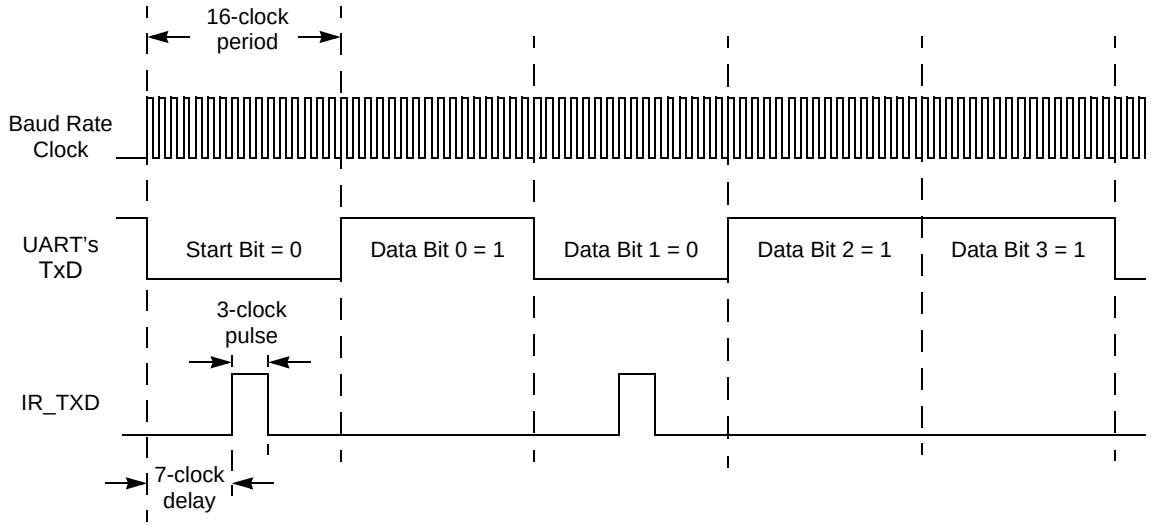



**Figure 20. Infrared Data Transmission**

## 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 21 displays data reception. When the infrared endec is enabled, the UART's RxD signal is internal to the Z8 Encore! XP F64xx Series products while the IR_RXD signal is received through the RxD pin.

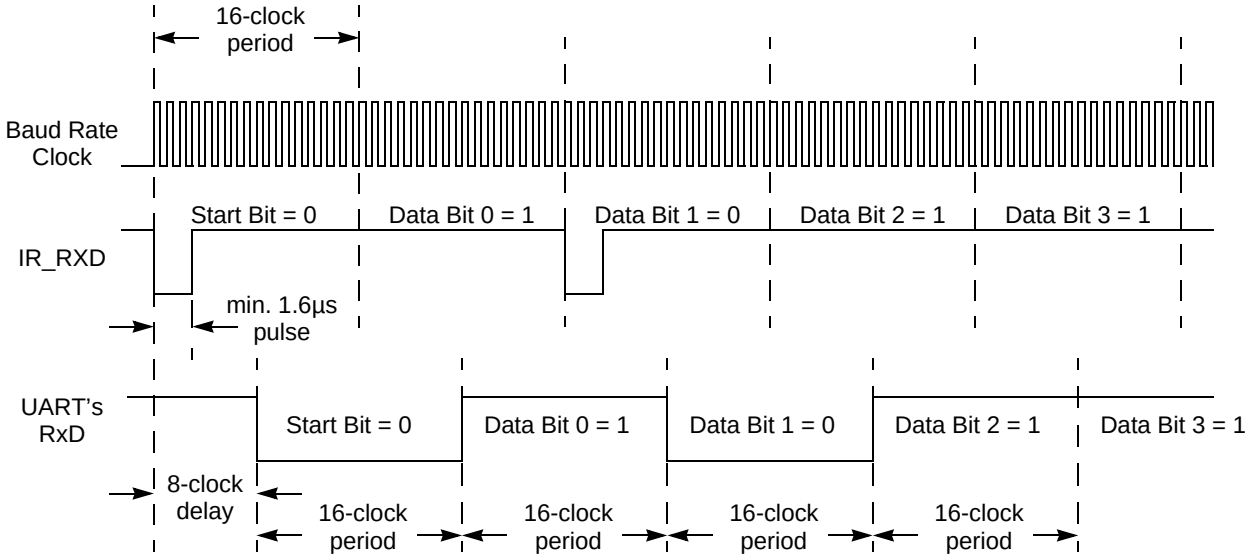



**Figure 21. Infrared Data Reception**

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

### Endec Receiver Synchronization

The IrDA receiver uses a local baud rate clock counter (0 to 15 clock periods) to generate an input stream for the UART and to create a sampling window for detection of incoming pulses. The generated UART input (UART RxD) is delayed by 8 baud rate clock periods with respect to the incoming IrDA data stream. When a falling edge in the input data stream is detected, the endec counter is reset. When the count reaches a value of 8, the UART RxD value is updated to reflect the value of the decoded data. When the count reaches 12 baud clock periods, the sampling window for the next incoming pulse opens. The window remains open until the count again reaches 8 (i.e., 24 baud clock periods

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

## Infrared Encoder/Decoder Control Register Definitions

All infrared endec configuration and status information is set by the UART control registers as defined in the UART Control Register Definitions section on page 98.

**Caution:** To prevent spurious signals during IrDA data transmission, set the IREN bit in the UARTx 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

The Serial Peripheral Interface 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-half the system clock frequency
- Error detection
- Dedicated Baud Rate Generator

## Architecture

The SPI may be configured as either a Master (in single or multimaster systems) or a Slave as displayed in Figures 22 through 24.

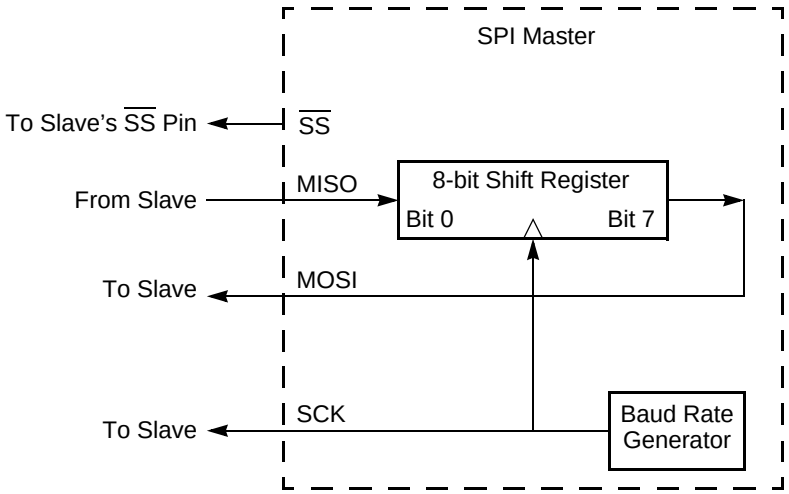



**Figure 22. SPI Configured as a Master in a Single-Master, Single-Slave System**

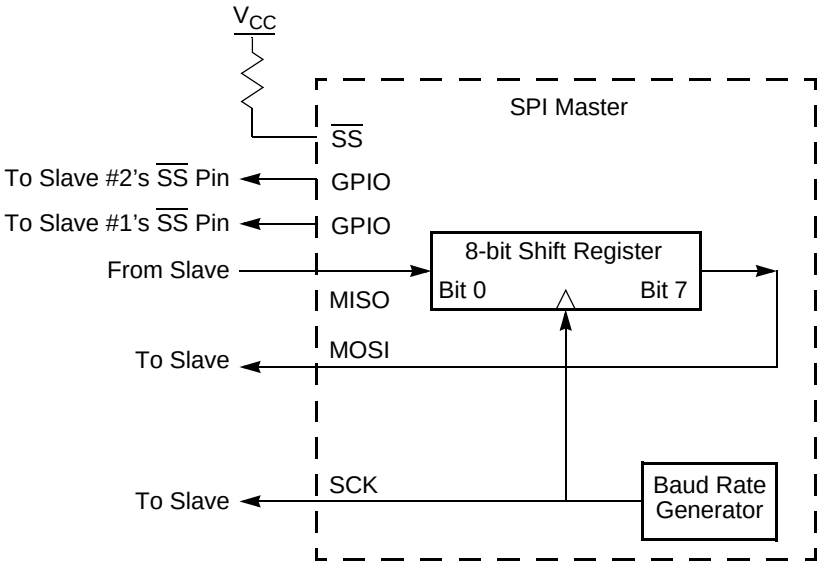



**Figure 23. SPI Configured as a Master in a Single-Master, Multiple-Slave System**

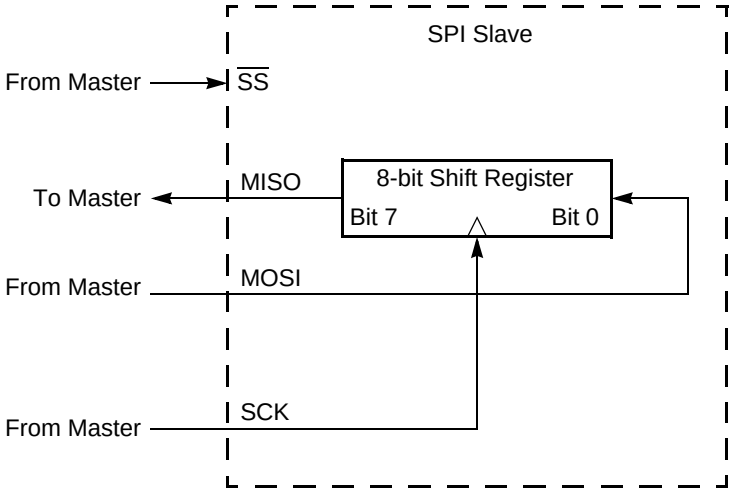



**Figure 24. SPI Configured as a Slave**

# 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 a transmit/receive shift register, a baud rate (clock) generator and a control unit.

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

## SPI Signals

The four basic SPI signals are:

- Master-In/Slave-Out
- Master-Out/Slave-In
- Serial Clock
- Slave Select

Each signal is described in both Master and Slave modes.

### Master-In/Slave-Out

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

### Master-Out/Slave-In

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

### Serial Clock

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

The Master and Slave are each capable of exchanging a character of data during a sequence of NUMBITS clock cycles (see the NUMBITS field in the SPI Mode Register section on page 125). In both Master and Slave SPI devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. Edge polarity is determined by the SPI phase and polarity control.

### Slave Select

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

When the SPI is configured as the only Master in an SPI system, the $\overline{SS}$ pin can be set as either an input or an output. Other GPIO output pins can also be employed to select external SPI Slave devices.

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

## SPI Clock Phase and Polarity Control

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

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

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

### Transfer Format PHASE Equals Zero

Figure 25 displays the timing diagram for an SPI transfer in which PHASE is cleared to 0. The two SCK waveforms show polarity with CLKPOL reset to 0 and with CLKPOL set to one. The diagram may be interpreted as either a Master or Slave timing diagram because the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the Master and the Slave.

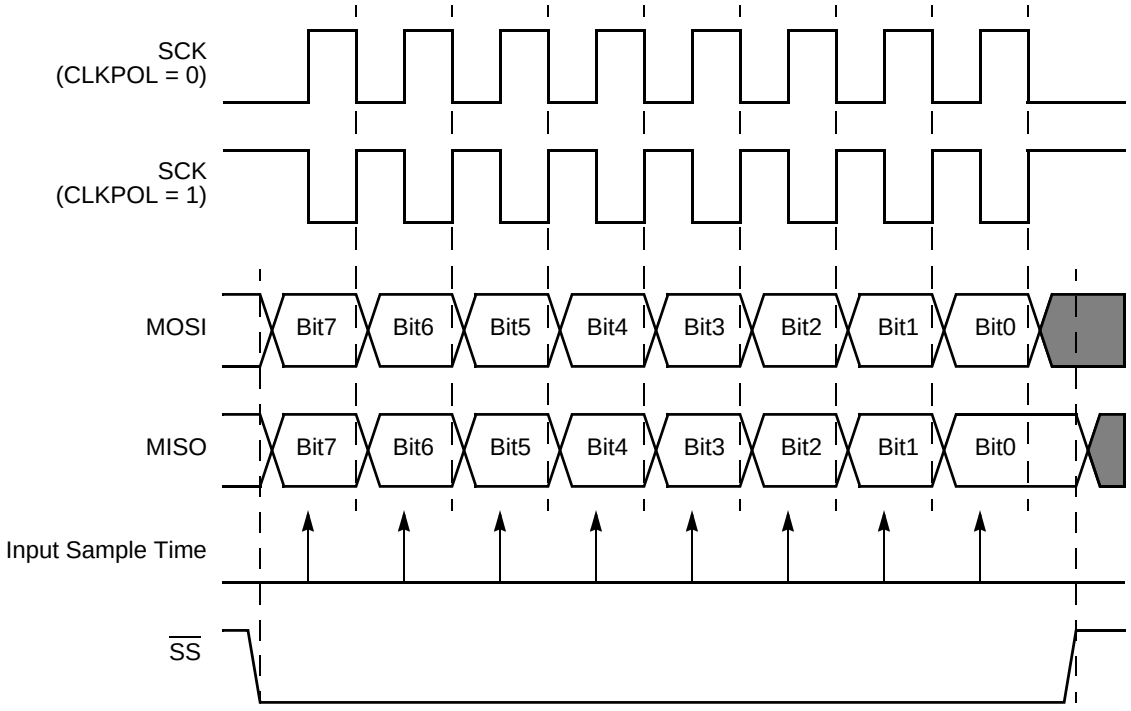



**Figure 25. SPI Timing When PHASE is 0**

### Transfer Format PHASE Equals One

Figure 26 displays the timing diagram for an SPI transfer in which PHASE is 1. Two waveforms are depicted for SCK, one for CLKPOL reset to 0 and another for CLKPOL set to 1.

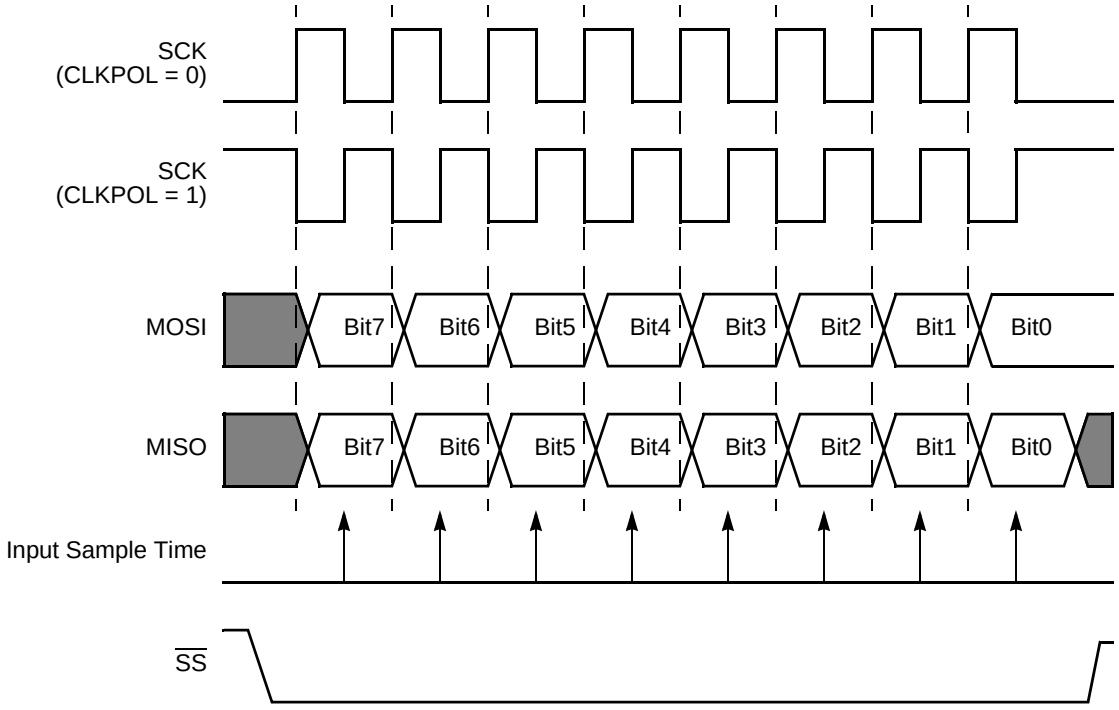



**Figure 26. SPI Timing When PHASE is 1**

## Multimaster Operation

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

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

## Slave Operation

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

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

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

## Error Detection

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

### Overrun (Write Collision)

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

### Mode Fault (Multimaster Collision)

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

### Slave Mode Abort

In the SLAVE Mode of operation, if the $\overline{SS}$ pin deasserts before all bits in a character have been transferred, the transaction is aborted. When this condition occurs, the ABT bit is set in the SPISTAT Register as well as the IRQ bit (indicating the transaction is complete).

The next time $\overline{SS}$ asserts, the MISO pin outputs SPIDAT[7], regardless of where the previous transaction left off. Writing a 1 to ABT clears this error flag.

## SPI Interrupts

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

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

## SPI Baud Rate Generator

In SPI Master Mode, the 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 SPI baud rate is calculated using the following equation:

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

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

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

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

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

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

# SPI Control Register Definitions

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

SPI Data Register: see page 121

SPI Control Register: see page 122

SPI Status Register: see page 123

SPI Mode Register: see page 125

SPI Diagnostic State Register: see page 126

SPI Baud Rate High and Low Byte Registers: see page 126

## SPI Data Register

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

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

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

**Table 64. SPI Data Register (SPIDATA)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DATA | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F60h | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>DATA | **Data**<br>Transmit and/or receive data. |

## SPI Control Register

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

**Table 65. SPI Control Register (SPICTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | IRQE | STR | BIRQ | PHASE | CLKPOL | WOR | MMEN | SPIEN |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F61h | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>IRQE | **Interrupt Request Enable**<br>0 = SPI interrupts are disabled. No interrupt requests are sent to the Interrupt Controller.<br>1 = SPI interrupts are enabled. Interrupt requests are sent to the Interrupt Controller. |
| [6]<br>STR | **Start an SPI Interrupt Request**<br>0 = No effect.<br>1 = Setting this bit to 1 also sets the IRQ bit in the SPI Status Register to 1. Setting this bit forces the SPI to send an interrupt request to the Interrupt Control. This bit can be used by software for a function similar to transmit buffer empty in a UART. Writing a 1 to the IRQ bit in the SPI Status Register clears this bit to 0. |
| [5]<br>BIRQ | **BRG Timer Interrupt Request**<br>If the SPI is enabled, this bit has no effect. If the SPI is disabled:<br>0 = The Baud Rate Generator timer function is disabled.<br>1 = The Baud Rate Generator timer function and time-out interrupt are enabled. |

| Bit | Description (Continued) |
|---|---|
| [4]<br>PHASE | **Phase Select**<br>Sets the phase relationship of the data to the clock. For more information about operation of the PHASE bit, see the SPI Clock Phase and Polarity Control section on page 116. |
| [3]<br>CLKPOL | **Clock Polarity**<br>0 = SCK idles Low (0).<br>1 = SCK idle High (1). |
| [2]<br>WOR | **Wire-OR (Open-Drain) Mode Enabled**<br>0 = SPI signal pins not configured for open-drain.<br>1 = All four SPI signal pins (SCK, SS, MISO, MOSI) configured for open-drain function. This setting is typically used for multimaster and/or multislave configurations. |
| [1]<br>MMEN | **SPI Master Mode Enable**<br>0 = SPI configured in SLAVE Mode.<br>1 = SPI configured in MASTER Mode. |
| [0]<br>SPIEN | **SPI Enable**<br>0 = SPI disabled.<br>1 = SPI enabled. |

## SPI Status Register

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

**Table 66. SPI Status Register (SPISTAT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | IRQ | OVR | COL | ABT | Reserved | | TXST | SLAS |
| RESET | 0 | | | | | | | 1 |
| R/W | R/W* | | | | R | | | |
| Address | F62h | | | | | | | |

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

| Bit | Description |
|---|---|
| [7]<br>IRQ | **Interrupt Request**<br>If SPIEN = 1, this bit is set if the STR bit in the SPICTL Register is set, or upon completion of an SPI master or slave transaction. This bit does not set if SPIEN = 0 and the SPI Baud Rate Generator is used as a timer to generate the SPI interrupt.<br>0 = No SPI interrupt request pending.<br>1 = SPI interrupt request is pending. |
| [6]<br>OVR | **Overrun**<br>0 = An overrun error has not occurred.<br>1 = An overrun error has been detected. |

| Bit | Description (Continued) |
|---|---|
| [5]<br>COL | **Collision**<br>0 = A multimaster collision (mode fault) has not occurred.<br>1 = A multimaster collision (mode fault) has been detected. |
| [4]<br>ABT | **Slave Mode Transaction Abort**<br>This bit is set if the SPI is configured in Slave Mode, a transaction is occurring and $\overline{SS}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the SPIMODE Register. The IRQ bit also sets, indicating the transaction has completed.<br>0 = A Slave Mode transaction abort has not occurred.<br>1 = A Slave Mode transaction abort has been detected. |
| [3:2] | **Reserved**<br>These bits are reserved and must be programmed to 00. |
| [1]<br>TXST | **Transmit Status**<br>0 = No data transmission currently in progress.<br>1 = Data transmission currently in progress. |
| [0]<br>SLAS | **Slave Select**<br>If SPI enabled as a Slave, then the following conditions are true:<br>0 = $\overline{SS}$ input pin is asserted (Low).<br>1 = $\overline{SS}$ input is not asserted (High).<br>If SPI enabled as a Master, this bit is not applicable. |

## SPI Mode Register

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

**Table 67. SPI Mode Register (SPIMODE)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | DIAG | NUMBITS[2:0] | | | SSIO | SSV |
| RESET | 0 | | | | | | | |
| R/W | R | | R/W | | | | | |
| Address | F63h | | | | | | | |

| Bit | Description |
|---|---|
| [7:6] | **Reserved**<br>These bits are reserved and must be programmed to 00. |
| [5]<br>DIAG | **Diagnostic Mode Control bit**<br>This bit is for SPI diagnostics. Setting this bit allows the Baud Rate Generator value to be read using the SPIBRH and SPIBRL Register locations.<br>0 = Reading SPIBRH, SPIBRL returns the value in the SPIBRH and SPIBRL registers.<br>1 = Reading SPIBRH returns bits [15:8] of the SPI Baud Rate Generator; and reading SPIBRL returns bits [7:0] of the SPI Baud Rate Counter. The Baud Rate Counter High and Low byte values are not buffered.<br>**Caution:** Exercise caution if reading the values while the BRG is counting. |
| [4]<br>NUMBITS[2:0] | **Number of Data Bits Per Character to Transfer**<br>This field contains the number of bits to shift for each character transfer. For information about valid bit positions when the character length is less than 8 bits, see the SPI Data Register (SPIDATA) description.<br>000 = 8 bits.<br>001 = 1 bit.<br>010 = 2 bits.<br>011 = 3 bits.<br>100 = 4 bits.<br>101 = 5 bits.<br>110 = 6 bits.<br>111 = 7 bits. |
| [1]<br>SSIO | **Slave Select I/O**<br>0 = $\overline{SS}$ pin configured as an input.<br>1 = $\overline{SS}$ pin configured as an output (Master Mode only). |
| [0]<br>SSV | **Slave Select Value**<br>If $\overline{SSIO}$ = 1 and SPI is configured as a Master, the following conditions are true:<br>0 = $\overline{SS}$ pin driven Low (0).<br>1 = $\overline{SS}$ pin driven High (1).<br>This bit has no effect if SSIO = 0 or if SPI is configured as a Slave. |

## SPI Diagnostic State Register

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

**Table 68. SPI Diagnostic State Register (SPIDST)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SCKEN | TCKEN | SPISTATE | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | |
| Address | F64h | | | | | | | |

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

## SPI Baud Rate High and Low Byte Registers

The SPI Baud Rate High and Low Byte registers, shown in Tables 69 and 70, combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator.

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

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

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F66h | | | | | | | |

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

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F67h | | | | | | | |

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

# I²C Controller

The I²C Controller makes the Z8 Encore! XP F64xx Series products bus-compatible with the I²C protocol. The I²C Controller consists of two bidirectional bus lines: a serial data signal (SDA) and a serial clock signal (SCL). Features of the I²C Controller include:

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

The I²C Controller in the Z8 Encore! XP F64xx Series products does not operate in SLAVE Mode.

## Architecture

Figure 27 displays the architecture of the I²C Controller.

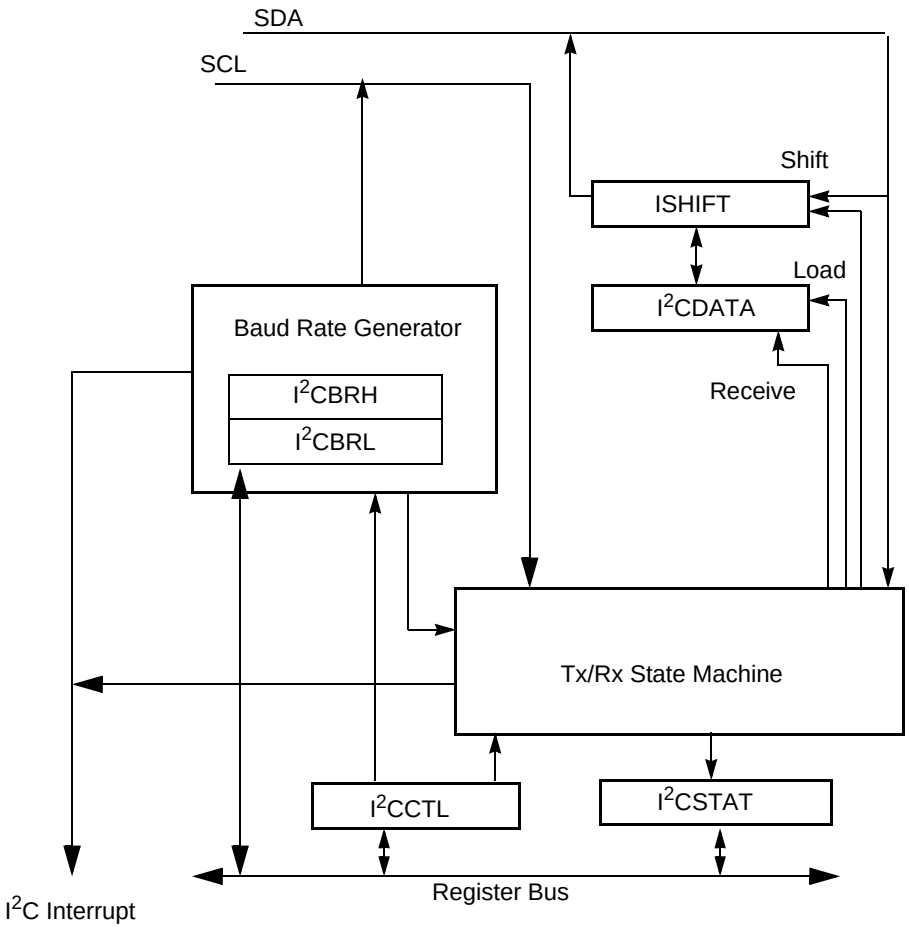



**Figure 27. I²C Controller Block Diagram**

# Operation

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

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

## SDA and SCL Signals

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

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

## I²C Interrupts

The I²C Controller contains four sources of interrupts—Transmit, Receive, Not Acknowledge and baud rate generator. These four interrupt sources are combined into a single interrupt request signal to the Interrupt Controller. The transmit interrupt is enabled by the IEN and TXI bits of the Control Register. The Receive and Not Acknowledge interrupts are enabled by the IEN bit of the Control Register. The baud rate generator interrupt is enabled by the BIRQ and IEN bits of the Control Register.

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

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

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

- The I²C Controller is enabled

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

**Note:** Writing to the I²C Data Register always clears the TRDE bit to 0. When TDRE is asserted, the I²C Controller pauses at the beginning of the Acknowledge cycle of the byte currently shifting out. It does not resume until the Data Register is written with the next value to send or until the stop or start bits are set, indicating that the current byte is the last one to send.

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

## Software Control of I²C Transactions

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

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

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

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

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

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

## Start and Stop Conditions

The master ($I^2C$) drives all start and stop signals and initiates all transactions. To start a transaction, the $I^2C$ Controller generates a start condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the $I^2C$ Controller generates a stop condition by creating a low-to-High transition of the SDA signal while the SCL signal is High. The start and stop bits in the $I^2C$ Control Register control the sending of the start and stop conditions. A master is also allowed to end one transaction and begin a new one by issuing a Restart. This is accomplished by setting the start bit at the end of a transaction, rather than the stop bit. Note that a start condition not sent until the start bit is set and data has been written to the $I^2C$ Data Register.

## Master Write and Read Transactions

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

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

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

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

When reading data from the slave, the $I^2C$ pauses after the data Acknowledge cycle until the receive interrupt is serviced and the RDRF bit of the status register is cleared by reading the $I^2C$ Data Register. Once the $I^2C$ data register has been read, the $I^2C$ reads the next data byte.

## Address Only Transaction with a 7-bit Address

In the situation where software determines if a slave with a 7-bit address is responding without sending or receiving data, a transaction can be done which only consists of an address phase. Figure 28 displays this *address only* transaction to determine if a slave with a 7-bit address will acknowledge. As an example, this transaction can be used after a *write* has been performed to an EEPROM to determine when the EEPROM completes its internal write operation and is again responding to $I^2C$ transactions. If the slave does not Acknowledge, the transaction can be repeated until the slave does Acknowledge.

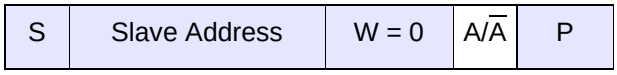



**Figure 28. 7-Bit Address Only Transaction Format**

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

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

## Write Transaction with a 7-Bit Address

Figure 29 displays the data transfer format for a 7-bit addressed slave. 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 | W = 0 | A | Data | A | Data | A | Data | A/A | P/S |
|---|---|---|---|---|---|---|---|---|---|---|

**Figure 29. 7-Bit Addressed Slave Data Transfer Format**

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

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

    If the slave does not acknowledge, the Not Acknowledge interrupt occurs (NCKI bit is set in the Status Register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The $I^2C$ Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore the following steps).
12. The $I^2C$ Controller loads the contents of the $I^2C$ Shift Register with the contents of the $I^2C$ Data Register.
13. The $I^2C$ Controller shifts the data out of using the SDA signal. After the first bit is sent, the transmit interrupt is asserted.
14. If more bytes remain to be sent, return to Step 9.
15. Software responds by setting the stop bit of the $I^2C$ Control Register (or start bit to initiate a new transaction). In the stop case, software clears the TXI bit of the $I^2C$ Control Register at the same time.

16. The I2C Controller completes transmission of the data on the SDA signal.
17. The slave may either Acknowledge or Not Acknowledge the last byte. Because either the stop or start bit is already set, the NCKI interrupt does not occur.
18. The I2C Controller sends the stop (or RESTART) condition to the I2C bus. The stop or start bit is cleared.

## Address Only Transaction with a 10-bit Address

In the situation where software wants to determine if a slave with a 10-bit address is responding without sending or receiving data, a transaction can be done which only consists of an address phase. Figure 30 displays this *address only* transaction to determine if a slave with 10-bit address will acknowledge. As an example, this transaction can be used after a write has been performed to an EEPROM to determine when the EEPROM completes its internal write operation and is again responding to I2C transactions. If the slave does not Acknowledge the transaction can be repeated until the slave is able to Acknowledge.

| S | Slave Address 1st 7 bits | W = 0 | A/A | Slave Address 2nd Byte | A/A | P |
|---|---|---|---|---|---|---|

**Figure 30. 10-Bit Address Only Transaction Format**

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

1. Software asserts the IEN bit in the I2C Control Register.
2. Software asserts the TXI bit of the I2C Control Register to enable transmit interrupts.
3. The I2C interrupt asserts, because the I2C Data Register is empty (TDRE = 1)
4. Software responds to the TDRE interrupt by writing the first slave address byte. The least significant bit must be 0 for the write operation.
5. Software asserts the start bit of the I2C Control Register.
6. The I2C Controller sends the start condition to the I2C slave.
7. The I2C Controller loads the I2C Shift Register with the contents of the I2C Data Register.
8. After one bit of address is shifted out by the SDA signal, the transmit interrupt is asserted.
9. Software responds by writing the second byte of address into the contents of the I2C Data Register.

10. The I2C Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. If the I2C slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL the I2C Controller sets the ACK bit in the I2C Status Register. Continue with Step 12.

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

## Write Transaction with a 10-Bit Address

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

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

**Figure 31. 10-Bit Addressed Slave Data Transfer Format**

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

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

1. Software asserts the IEN bit in the I2C Control Register.

2. Software asserts the TXI bit of the I2C Control Register to enable transmit interrupts.
3. The I2C interrupt asserts because the I2C Data Register is empty.
4. Software responds to the TDRE interrupt by writing the first slave address byte to the I2C Data Register. The least significant bit must be 0 for the write operation.
5. Software asserts the start bit of the I2C Control Register.
6. The I2C Controller sends the start condition to the I2C slave.
7. The I2C Controller loads the I2C Shift Register with the contents of the I2C Data Register.
8. After one bit of address is shifted out by the SDA signal, the transmit interrupt is asserted.
9. Software responds by writing the second byte of address into the contents of the I2C Data Register.
10. The I2C Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. If the I2C slave acknowledges the first address byte by pulling the SDA signal Low during the next High period of SCL, the I2C Controller sets the ACK bit in the I2C Status Register. Continue with Step 12.

    If the slave does not acknowledge the first address byte, the I2C Controller sets the NCKI bit and clears the ACK bit in the I2C Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I2C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore the following steps).
12. The I2C Controller loads the I2C Shift Register with the contents of the I2C Data Register.
13. The I2C Controller shifts the second address byte out the SDA signal. After the first bit has been sent, the transmit interrupt is asserted.
14. Software responds by writing a data byte to the I2C Data Register.
15. The I2C Controller completes shifting the contents of the shift register on the SDA signal.
16. If the I2C slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL, the I2C Controller sets the ACK bit in the I2C Status Register. Continue with Step 17.

    If the slave does not acknowledge the second address byte or one of the data bytes, the I2C Controller sets the NCKI bit and clears the ACK bit in the I2C Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I2C Controller sends the stop condition on the bus and

clears the stop and NCKI bits. The transaction is complete (ignore the following steps).

17. The I²C Controller shifts the data out by the SDA signal. After the first bit is sent, the transmit interrupt is asserted.
18. If more bytes remain to be sent, return to Step 14.
19. If the last byte is currently being sent, software sets the stop bit of the I²C Control Register (or start bit to initiate a new transaction). In the stop case, software also clears the TXI bit of the I²C Control Register at the same time.
20. The I²C Controller completes transmission of the last data byte on the SDA signal.
21. The slave may either Acknowledge or Not Acknowledge the last byte. Because either the stop or start bit is already set, the NCKI interrupt does not occur.
22. The I²C Controller sends the stop (or RESTART) condition to the I²C bus and clears the stop (or start) bit.

## Read Transaction with a 7-Bit Address

Figure 32 displays the data transfer format for a read operation to a 7-bit addressed slave. The shaded regions indicate data transferred from the I²C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I²C Controller.

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

**Figure 32. Receive Data Transfer Format for a 7-Bit Addressed Slave**

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

1. Software writes the I²C Data Register with a 7-bit slave address plus the read bit (=1).
2. Software asserts the start bit of the I²C Control Register.
3. If this is a single byte transfer, Software asserts the NAK bit of the I²C Control Register so that after the first byte of data has been read by the I²C Controller, a Not Acknowledge is sent to the I²C slave.
4. The I²C Controller sends the start condition.
5. The I²C Controller shifts the address and read bit out the SDA signal.
6. If the I²C slave acknowledges the address by pulling the SDA signal Low during the next High period of SCL, the I²C Controller sets the ACK bit in the I²C Status Register. Continue with Step 7.

   If the slave does not acknowledge, the Not Acknowledge interrupt occurs (NCKI bit is

set in the Status Register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the stop bit and clearing the TXI bit. The I²C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore the following steps).

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

## Read Transaction with a 10-Bit Address

Figure 33 displays the read transaction format for a 10-bit addressed slave. The shaded regions indicate data transferred from the I²C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I²C Controller.

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

**Figure 33. Receive Data Format for a 10-Bit Addressed Slave**

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

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

1. Software writes `11110B` followed by the two address bits and a 0 (write) to the I²C Data Register.
2. Software asserts the start and TXI bits of the I²C Control Register.
3. The I²C Controller sends a start condition.
4. The I²C Controller loads the I²C Shift Register with the contents of the I²C Data Register.

5. After the first bit has been shifted out, a transmit interrupt is asserted.
6. Software responds by writing the lower eight bits of address to the I²C Data Register.
7. The I²C Controller completes shifting of the two address bits and a 0 (write).
8. If the I²C slave acknowledges the first address byte by pulling the SDA signal Low during the next High period of SCL, the I²C Controller sets the ACK bit in the I²C Status Register. Continue with Step 9.

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

    If the slave does not acknowledge the second address byte, the I²C Controller sets the NCKI bit and clears the ACK bit in the I²C Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I²C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore the following steps).
15. The I²C Controller sends the repeated start condition.
16. The I²C Controller loads the I²C Shift Register with the contents of the I²C Data Register (third address transfer).
17. The I²C Controller sends `11110B` followed by the two most significant bits of the slave read address and a 1 (read).
18. The I²C slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL

If the slave were to Not Acknowledge at this point (this should not happen because the slave did acknowledge the first two address bytes), software would respond by setting the stop and flush bits and clearing the TXI bit. The I2C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore the following steps).

19. The I2C Controller shifts in a byte of data from the I2C slave on the SDA signal. The I2C Controller sends a Not Acknowledge to the I2C slave if the NAK bit is set (last byte), else it sends an Acknowledge.
20. The I2C Controller asserts the receive interrupt (RDRF bit set in the Status Register).
21. Software responds by reading the I2C Data Register which clears the RDRF bit. If there is only one more byte to receive, set the NAK bit of the I2C Control Register.
22. If there are one or more bytes to transfer, return to Step 19.
23. After the last byte is shifted in, a Not Acknowledge interrupt is generated by the I2C Controller.
24. Software responds by setting the stop bit of the I2C Control Register.
25. A stop condition is sent to the I2C slave and the stop and NCKI bits are cleared.

# I2C Control Register Definitions

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

I2C Data Register: see page 141

I2C Status Register: see page 142

I2C Control Register: see page 144

I2C Baud Rate High and Low Byte Registers: see page 145

I2C Diagnostic State Register: see page 147

I2C Diagnostic Control Register: see page 149

## I2C Data Register

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

Table 71. I²C Data Register (I2CDATA)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DATA | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F50h | | | | | | | |

## I²C Status Register

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

Table 72. I²C Status Register (I2CSTAT)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TDRE | RDRF | ACK | 10B | RD | TAS | DSS | NCKI |
| RESET | 1 | 0 | | | | | | |
| R/W | R | | | | | | | |
| Address | F51h | | | | | | | |

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

| Bit | Description (Continued) |
|---|---|
| [5]<br>ACK | **Acknowledge**<br>This bit indicates the status of the Acknowledge for the last byte transmitted or received. When set, this bit indicates that an Acknowledge occurred for the last byte transmitted or received. This bit is cleared when IEN = 0 or when a Not Acknowledge occurred for the last byte transmitted or received. It is not reset at the beginning of each transaction and is not reset when this register is read.<br>**Caution:** When making decisions based on this bit within a transaction, software cannot determine when the bit is updated by hardware. In the case of write transactions, the $I^2C$ pauses at the beginning of the Acknowledge cycle if the next transmit data or address byte has not been written (TDRE = 1) and stop and start = 0. In this case the ACK bit is not updated until the transmit interrupt is serviced and the Acknowledge cycle for the previous byte completes. For examples of how the ACK bit can be used, see the Address Only Transaction with a 7-bit Address section on page 133 and the Address Only Transaction with a 10-bit Address section on page 135. |
| [4]<br>10B | **10-Bit Address**<br>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. |
| [3]<br>RD | **Read**<br>This bit indicates the direction of transfer of the data. It is active High during a read. The status of this bit is determined by the least significant bit of the $I^2C$ Shift Register after the start bit is set. |
| [2]<br>TAS | **Transmit Address State**<br>This bit is active High while the address is being shifted out of the $I^2C$ Shift Register. |
| [1]<br>DSS | **Data Shift State**<br>This bit is active High while data is being shifted to or from the $I^2C$ Shift Register. |
| [0]<br>NCKI | **NACK Interrupt**<br>This bit is set High when a Not Acknowledge condition is received or sent and neither the start nor the stop bit is active. When set, this bit generates an interrupt that can only be cleared by setting the start or stop bit, allowing you to specify whether to perform a stop or a repeated start. |

# I²C Control Register

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

**Table 73. I²C Control Register (I2CCTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | IEN | START | STOP | BIRQ | TXI | NAK | FLUSH | FILTEN |
| RESET | 0 | | | | | | | |
| R/W | R/W | R/W1 | R/W1 | R/W | R/W | R/W1 | W1 | R/W |
| Address | F52h | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>IEN | **I²C Enable**<br>1 = The I²C transmitter and receiver are enabled.<br>0 = The I²C transmitter and receiver are disabled. |
| [6]<br>START | **Send Start Condition**<br>This bit sends a start condition. Once asserted, it is cleared by the I²C Controller after it sends a start condition or if the IEN bit is deasserted. If this bit is 1, it cannot be cleared to 0 by writing to the register. After this bit is set, a start condition is sent if there is data in the I²C Data Register or I²C Shift Register. If there is no data in one of these registers, the I²C Controller waits until the Data Register is written. If this bit is set while the I²C Controller is shifting out data, it generates a start condition after the byte shifts and the acknowledge phase completes. If the stop bit is also set, it also waits until the stop condition is sent before the sending the start condition. |
| [5]<br>STOP | **Send Stop Condition**<br>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. AFter it is 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. |
| [4]<br>BIRQ | **Baud Rate Generator Interrupt Request**<br>This bit allows the I²C Controller to be used as an additional timer when the I²C Controller is disabled. This bit is ignored when the I²C Controller is enabled.<br>1 = An interrupt occurs every time the baud rate generator counts down to one.<br>0 = No baud rate generator interrupt occurs. |
| [3]<br>TXI | **Enable TDRE Interrupts**<br>This bit enables the transmit interrupt when the I²C Data Register is empty (TDRE = 1).<br>1 = Transmit interrupt (and DMA transmit request) is enabled.<br>0 = Transmit interrupt (and DMA transmit request) is disabled. |
| [2]<br>NAK | **Send NAK**<br>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. If this bit is 1, it cannot be cleared to 0 by writing to the register. |

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

## I²C Baud Rate High and Low Byte Registers

The I²C Baud Rate High and Low Byte registers, shown in Tables 74 and 75, combine to form a 16-bit reload value, BRG[15:0], for the I²C Baud Rate Generator.

When the I²C 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 I²C by clearing the IEN bit in the I²C Control Register to 0.
2. Load the appropriate 16-bit count value into the I²C 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 I²C Control Register to 1.

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

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

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRH | | | | | | | |
| RESET | FFh | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F53h | | | | | | | |

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

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRL | | | | | | | |
| RESET | FFh | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F54h | | | | | | | |

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

## I²C Diagnostic State Register

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

**Table 76. I²C Diagnostic State Register (I2CDST)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SCLIN | SDAIN | STPCNT | TXRXSTATE | | | | |
| RESET | X | | 0 | | | | | |
| R/W | R | | | | | | | |
| Address | F55h | | | | | | | |

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

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

## I²C Diagnostic Control Register

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

**Table 77. I²C Diagnostic Control Register (I2CDIAG)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | | | | DIAG |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | R/W |
| Address | F56h | | | | | | | |

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

# Direct Memory Access Controller

The Z8 Encore! XP F64xx Series 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 ADC operation and transfers SINGLE-SHOT Mode ADC output data to the register file.

## 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 to the eZ8 CPU.
4. If the Current Address equals the End Address, then the following conditions are true:
   - 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 the Current Address does not equal the End Address, then the Current Address increments by 1 (single-byte transfer) or 2 (two-byte word transfer).

### Configuring DMA0 and DMA1 for Data Transfer

Observe the following procedure 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 starting and ending register file addresses. The 12-bit Start Address is provided by {DMA*x*_H[3:0], DMA_START[7:0]}. The 12-bit End Address is provided 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 operations:
   - 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 appropriate
   - 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:
   - The 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, the 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, then the DMA_ADC initiates data conversion in the next higher-numbered ADC analog input.

## Configuring DMA_ADC for Data Transfer

Observe the following procedure to configure and enable the 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 operations:
   - Enable the DMA_ADC interrupt request, if appropriate
   - 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, the Analog-to-Digital Converter must be configured for SINGLE-SHOT Mode. If the ADC_IN field in the DMA_ADC Control Register is greater than 000b, the ADC must be in SINGLE-SHOT Mode.

Continuous Mode operation of the ADC can only be used in conjunction with the 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

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

DMAx Control Register: see page 152

DMAx I/O Address Register: see page 154

DMAx Address High Nibble Register: see page 155

DMAx Start/Current Address Low Byte Register: see page 156

DMAx End Address Low Byte Register: see page 156

DMA_ADC Address Register: see page 157

DMA_ADC Control Register: see page 158

DMA_ADC Status Register: see page 159

## DMA*x* Control Register

The DMA*x* Control Register, shown in Table 78, enables and selects the mode of operation for DMA*x*.

**Table 78. DMA*x* Control Register (DMA*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DEN | DLE | DDIR | IRQEN | WSEL | RSS | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB0h, FB8h | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>DEN | **DMA*x* Enable**<br>0 = DMA*x* is disabled and data transfer requests are disregarded.<br>1 = DMA*x* is enabled and initiates a data transfer upon receipt of a request from the trigger source. |
| [6]<br>DLE | **DMA*x* Loop Enable**<br>0 = DMA*x* reloads the original Start Address and is then disabled after the End Address data is transferred.<br>1 = DMA*x*, after the End Address data is transferred, reloads the original Start Address and continues operating. |
| [5]<br>DDIR | **DMA*x* Data Transfer Direction**<br>0 = Register file → on-chip peripheral control register.<br>1 = On-chip peripheral control → register file. |
| [4]<br>IRQEN | **DMA*x* Interrupt Enable**<br>0 = DMA*x* does not generate any interrupts.<br>1 = DMA*x* generates an interrupt when the End Address data is transferred. |

| Bit | Description (Continued) |
|---|---|
| [3]<br>WSEL | **Word Select**<br>0 = DMA*x* transfers a single byte per request.<br>1 = DMA*x* transfers a two-byte word per request. The address for the on-chip peripheral control register must be an even address. |
| [2:0]<br>RSS | **Request Trigger Source Select**<br>The Request Trigger Source Select field determines the peripheral that can initiate a DMA 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.<br>000 = Timer 0.<br>001 = Timer 1.<br>010 = Timer 2.<br>011 = Timer 3.<br>100 = DMA0 Control Register: UART0 Received Data Register contains valid data. DMA1 Control Register: UART0 Transmit Data Register empty.<br>101 = DMA0 Control Register: UART1 Received Data Register contains valid data. DMA1 Control Register: UART1 Transmit Data Register empty.<br>110 = DMA0 Control Register: $I^2C$ Receiver Interrupt. DMA1 Control Register: $I^2C$ Transmitter Interrupt Register empty.<br>111 = Reserved. |

## DMA*x* I/O Address Register

The DMA*x* I/O Address Register, shown in Table 79, contains the low byte of the on-chip peripheral address for data transfer. The full 12-bit register file address is provided by {FFh, 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 79. DMA*x* I/O Address Register (DMA*x*IO)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_IO | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB1h, FB9h | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>DMA_IO | **DMA On-Chip Peripheral Control Register Address**<br>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, shown in Table 80, specifies the upper four bits of address for the Start/Current and End addresses of DMA*x*.

**Table 80. DMA*x* Address High Nibble Register (DMA*x*H)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_END_H | | | | DMA_START_H | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB2h, FBAh | | | | | | | |

| Bit | Description |
|---|---|
| [7:4]<br>DMA_END_H | **DMA*x* End Address High Nibble**<br>These bits, used with the DMA*x* End Address Low Register, form a 12-bit End Address. The full 12-bit address is provided by {DMA_END_H[3:0], DMA_END[7:0]}. |
| [3:0]<br>DMA_START_H | **DMA*x* Start/Current Address High Nibble**<br>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 provided 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 Byte Register, shown in Table 81, in conjunction with the DMA*x* Address High Nibble Register, shown in Table 80, 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 81. DMA*x* Start/Current Address Low Byte Register (DMA*x*START)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_START | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB3h, FBBh | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>DMA_START | **DMA*x* Start/Current Address Low**<br>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 provided 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, shown in Table 82, forms a 12-bit End Address.

**Table 82. DMA*x* End Address Low Byte Register (DMA*x*END)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_END | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB4h, FBCh | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>DMA_END | **DMA*x* End Address Low**<br>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 provided by {DMA_END_H[3:0], DMA_END[7:0]}. |

## DMA_ADC Address Register

The DMA_ADC Address Register, shown in Table 84, points to a block of the register file to store the ADC conversion values displayed in Table 83. 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 83 provides an example of the register file addresses if the DMA_ADC Address Register contains the value 72h.

**Table 83. DMA_ADC Register File Address Example**

| ADC Analog Input | Register File Address (Hex)* |
|---|---|
| 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 |

Note: *DMAA_ADDR is set to 72h.

**Table 84. DMA_ADC Address Register (DMAA_ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMAA_ADDR | | | | | | | Reserved |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FBDh | | | | | | | |

| Bit | Description |
|---|---|
| [7:1] DMAA_ADDR | **DMA_ADC Address**<br>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}. |
| 0 | **Reserved**<br>This bit is reserved and must be programmed to 0. |

## DMA_ADC Control Register

The DMA_ADC Control Register, shown in Table 85, enables and sets options (DMA enable and interrupt enable) for ADC operation.

**Table 85. DMA_ADC Control Register (DMAACTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DAEN | IRQEN | Reserved | | ADC_IN | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FBEh | | | | | | | |

| Bit | Description |
|---|---|
| [7] DAEN | **DMA_ADC Enable**<br>0 = DMA_ADC is disabled and the ADC analog input Number (ADC_IN) is reset to 0.<br>1 = DMA_ADC is enabled. |
| [6] IRQEN | **Interrupt Enable**<br>0 = DMA_ADC does not generate any interrupts.<br>1 = DMA_ADC generates an interrupt after transferring data from the last ADC analog input specified by the ADC_IN field. |

| Bit | Description (Continued) |
|---|---|
| [5:4] | **Reserved**<br>These bits are reserved and must be programmed to 00. |
| [3:0]<br>ADC_IN | **ADC Analog Input Number**<br>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.<br>0000 = ADC analog input 0 updated.<br>0001 = ADC analog inputs 0–1 updated.<br>0010 = ADC analog inputs 0–2 updated.<br>0011 = ADC analog inputs 0–3 updated.<br>0100 = ADC analog inputs 0–4 updated.<br>0101 = ADC analog inputs 0–5 updated.<br>0110 = ADC analog inputs 0–6 updated.<br>0111 = ADC analog inputs 0–7 updated.<br>1000 = ADC analog inputs 0–8 updated.<br>1001 = ADC analog inputs 0–9 updated.<br>1010 = ADC analog inputs 0–10 updated.<br>1011 = ADC analog inputs 0–11 updated.<br>1100–1111 = Reserved. |

## DMA_ADC Status Register

The DMA Status Register, shown in Table 86, 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 86. DMA_ADC Status Register (DMAA_STAT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | CADC[3:0] | | | | Reserved | IRQA | IRQ1 | IRQ0 |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | |
| Address | FBFh | | | | | | | |

| Bit | Description |
|---|---|
| [7:4]<br>CADC[3:0] | **Current ADC Analog Input**<br>This field identifies the Analog Input that the ADC is currently converting. |
| [3] | **Reserved**<br>This bit is reserved and must be programmed to 0. |

| Bit | Description (Continued) |
|---|---|
| [2]<br>IRQA | **DMA_ADC Interrupt Request Indicator**<br>This bit is automatically reset to 0 each time a read from this register occurs.<br>0 = DMA_ADC is not the source of the interrupt from the DMA Controller.<br>1 = DMA_ADC completed transfer of data from the last ADC analog input and generated an interrupt. |
| [1]<br>IRQ1 | **DMA1 Interrupt Request Indicator**<br>This bit is automatically reset to 0 each time a read from this register occurs.<br>0 = DMA1 is not the source of the interrupt from the DMA Controller.<br>1 = DMA1 completed transfer of data to/from the End Address and generated an interrupt. |
| [0]<br>IRQ0 | **DMA0 Interrupt Request Indicator**<br>This bit is automatically reset to 0 each time a read from this register occurs.<br>0 = DMA0 is not the source of the interrupt from the DMA Controller.<br>1 = DMA0 completed transfer of data to/from the End Address and generated an interrupt. |

# Analog-to-Digital Converter

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

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

## Architecture

Figure 34 displays the three major functional blocks (converter, analog multiplexer, and voltage reference generator) of the ADC. The ADC converts an analog input signal to its digital representation. The 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 $V_{REF}$ pin or generated internally by the voltage reference generator.

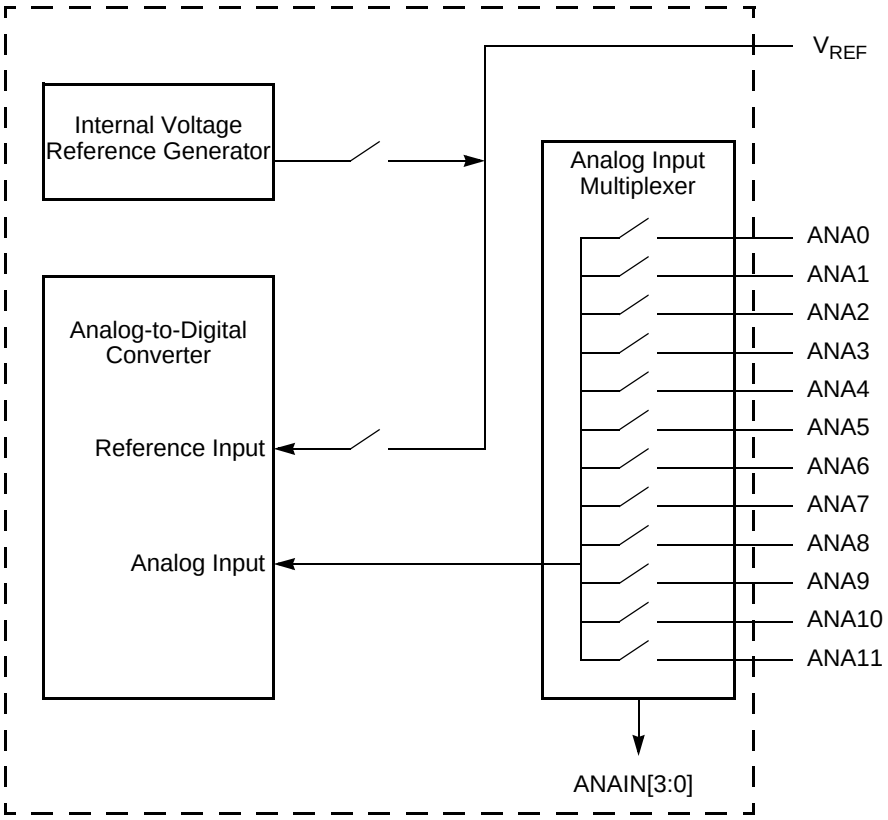



**Figure 34. Analog-to-Digital Converter Block Diagram**

The sigma-delta ADC architecture provides alias and image attenuation below the amplitude resolution of the ADC in the frequency range of DC to one-half the ADC clock rate (one-fourth the system clock rate). The ADC provides alias free conversion for frequencies up to one-half the ADC clock rate. Therefore, the sigma-delta ADC exhibits high noise immunity, which makes it ideal for embedded applications. In addition, monotonicity (no missing codes) is guaranteed by design.

# Operation

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

## Automatic Power-Down

If the ADC is idle (i.e., no conversions are in progress) for 160 consecutive system clock cycles, portions of the ADC are automatically powered down. From this powered-down state, the ADC requires 40 system clock cycles to power up. The ADC powers up when a conversion is requested 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. Observe the following procedure for setting up the ADC and initiating a single-shot conversion:

1. Enable the appropriate 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 the ANAIN[3:0] field to select one of the 12 analog input sources
   - Clear CONT to 0 to select a single-shot conversion
   - Write to the $\overline{\text{VREF}}$ bit to enable or disable the internal voltage reference generator
   - Set CEN to 1 to start the conversion
3. CEN remains 1 while the conversion is in progress. A single-shot conversion requires 5129 system clock cycles to complete. If a single-shot conversion is requested from an ADC powered-down state, the ADC uses 40 additional clock cycles to power up before beginning the 5129 cycle conversion.
4. When the conversion is complete, the ADC control logic performs the following operations:
   - 10-bit data result written to {ADCD_H[7:0], ADCD_L[7:6]}
   - CEN resets to 0 to indicate the conversion is complete
   - An interrupt request is sent to the Interrupt Controller
5. If the ADC remains idle for 160 consecutive system clock cycles, it is automatically powered down.

## Continuous Conversion

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

**Caution:** In Continuous Mode, you 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.

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

1. Enable the appropriate 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 the ANAIN[3:0] field to select one of the 12 analog input sources
   - Set CONT to 1 to select continuous conversion
   - Write to the $\overline{\text{VREF}}$ bit to enable or disable the internal voltage reference generator
   - Set CEN to 1 to start the conversions
3. When the first conversion in continuous operation is complete (after 5129 system clock cycles, plus the 40 cycles required to power up, if necessary), the ADC control logic performs the following operations:
   - CEN resets to 0 to indicate the first conversion is complete. CEN remains 0 for all subsequent conversions in continuous operation
   - An interrupt request is sent to the Interrupt Controller to indicate the conversion is complete
4. Thereafter, the ADC writes a new 10-bit data result to {ADCD_H[7:0], ADCD_L[7:6]} every 256 system clock cycles. An interrupt request is sent to the Interrupt Controller when each conversion is complete.
5. To disable continuous conversion, clear the CONT bit in the ADC Control Register to 0.

## 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 about the DMA and configuring for ADC operations, see the Direct Memory Access Controller chapter on page 150.

# ADC Control Register Definitions

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

ADC Control Register: see page 165

ADC Data High Byte Register: see page 167

ADC Data Low Bits Register: see page 168

## ADC Control Register

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

**Table 87. ADC Control Register (ADCCTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | CEN | Reserved | VREF | CONT | ANAIN[3:0] | | | |
| RESET | 0 | | 1 | 0 | | | | |
| R/W | R/W | | | | | | | |
| Address | F70h | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>CEN | **Conversion Enable**<br>0 = Conversion is complete. Writing a 0 produces no effect. The ADC automatically clears this bit to 0 when a conversion has been completed.<br>1 = Begin conversion. Writing a 1 to this bit starts a conversion. If a conversion is already in progress, the conversion restarts. This bit remains 1 until the conversion is complete. |
| [6] | **Reserved**<br>This bit is reserved and must be programmed to 0. |
| [5]<br>VREF | **Voltage Reference**<br>0 = Internal voltage reference generator enabled. The $V_{REF}$ pin should be left unconnected (or capacitively coupled to analog ground) if the internal voltage reference is selected as the ADC reference voltage.<br>1 = Internal voltage reference generator disabled. An external voltage reference must be provided through the $V_{REF}$ pin. |

| Bit | Description (Continued) |
|---|---|
| [4]<br>CONT | **Conversion**<br>0 = Single-shot conversion. ADC data is output once at completion of the 5129 system clock cycles.<br>1 = Continuous conversion. ADC data updated every 256 system clock cycles. |
| [3:0]<br>ANAIN[3:0] | **Analog Input Select**<br>These bits select the analog input for conversion. For information about the Port pins available with each package style, see the Signal and Pin Descriptions chapter on page 7. Do not enable unavailable analog inputs.<br>0000 = ANA0.<br>0001 = ANA1.<br>0010 = ANA2.<br>0011 = ANA3.<br>0100 = ANA4.<br>0101 = ANA5.<br>0110 = ANA6.<br>0111 = ANA7.<br>1000 = ANA8.<br>1001 = ANA9.<br>1010 = ANA10.<br>1011 = ANA11.<br>11xx = Reserved. |

## ADC Data High Byte Register

The ADC Data High Byte Register, shown in Table 88, contains the upper eight bits of the 10-bit ADC output. During a single-shot conversion, this value is invalid. Access to the ADC Data High Byte Register is read-only. The full 10-bit ADC result is provided by {ADCD_H[7:0], ADCD_L[7:6]}. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

**Table 88. ADC Data High Byte Register (ADCD_H)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | ADCD_H | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | F72h | | | | | | | |

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

## ADC Data Low Bits Register

The ADC Data Low Bits Register, Table 89, contains the lower two bits of the conversion value. The data in the ADC Data Low Bits Register is latched each time the ADC Data High Byte Register is read. Reading this register always returns the lower two bits of the conversion last read into the ADC High Byte Register. Access to the ADC Data Low Bits Register is read-only. The full 10-bit ADC result is provided by {ADCD_H[7:0], ADCD_L[7:6]}.

**Table 89. ADC Data Low Bits Register (ADCD_L)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | ADCD_L | | Reserved | | | | | |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | F73h | | | | | | | |

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

# Flash Memory

The products in the Z8 Encore! XP F64xx Series feature 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 512 byte per page. The 512 byte page is the minimum Flash block size that can be erased. The Flash memory is also divided into 8 sectors which can be protected from programming and erase operations on a per sector basis.

Table 90 describes the Flash memory configuration for each device in the Z8 Encore! XP F64xx Series. Table 91 lists the sector address ranges. Figure 35 displays the Flash memory arrangement.

**Table 90. Flash Memory Configurations**

| Part Number | Flash Size | Number of Pages | Flash Memory Addresses | Sector Size | Number of Sectors | Pages per Sector |
|---|---|---|---|---|---|---|
| Z8F162x | 16K (16,384) | 32 | 0000h–3FFFh | 2K (2048) | 8 | 4 |
| Z8F242x | 24K (24,576) | 48 | 0000h–5FFFh | 4K (4096) | 6 | 8 |
| Z8F322x | 32K (32,768) | 64 | 0000h–7FFFh | 4K (4096) | 8 | 8 |
| Z8F482x | 48K (49,152) | 96 | 0000h–BFFFh | 8K (8192) | 6 | 16 |
| Z8F642x | 64K (65,536) | 128 | 0000h–FFFFh | 8K (8192) | 8 | 16 |

**Table 91. Flash Memory Sector Addresses**

| Sector Number | Flash Sector Address Ranges | | | | |
|---|---|---|---|---|---|
| | Z8F162x | Z8F242x | Z8F322x | Z8F482x | Z8F642x |
| 0 | 0000h–07FFh | 0000h–0FFFh | 0000h–0FFFh | 0000h–1FFFh | 0000h–1FFFh |
| 1 | 0800h–0FFFh | 1000h–1FFFh | 1000h–1FFFh | 2000h–3FFFh | 2000h–3FFFh |
| 2 | 1000h–17FFh | 2000h–2FFFh | 2000h–2FFFh | 4000h–5FFFh | 4000h–5FFFh |
| 3 | 1800h–1FFFh | 3000h–3FFFh | 3000h–3FFFh | 6000h–7FFFh | 6000h–7FFFh |
| 4 | 2000h–27FFh | 4000h–4FFFh | 4000h–4FFFh | 8000h–9FFFh | 8000h–9FFFh |
| 5 | 2800h–2FFFh | 5000h–5FFFh | 5000h–5FFFh | A000h–BFFFh | A000h–BFFFh |
| 6 | 3000h–37FFh | N/A | 6000h–6FFFh | N/A | C000h–DFFFh |
| 7 | 3800h–3FFFh | N/A | 7000h–7FFFh | N/A | E000h–FFFFh |

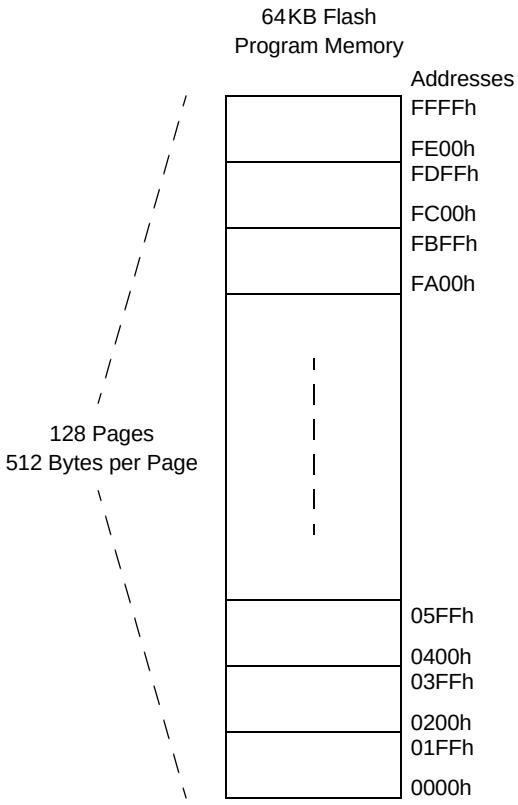



**Figure 35. Flash Memory Arrangement**

# Information Area

Table 92 describes the Z8 Encore! XP F64xx Series Information Area. This 512-byte Information Area is accessed by setting bit 7 of the Page Select Register to 1. When access is enabled, the Information Area is mapped into Flash memory and overlays the 512 bytes at addresses FE00h to FFFFh. When the Information Area access is enabled, LDC instructions return data from the Information Area. CPU instruction fetches always comes from Flash memory regardless of the Information Area access bit. Access to the Information Area is read-only.

**Table 92. Z8 Encore! XP F64xx Series Information Area Map**

| Flash Memory Address (Hex) | Function |
|---|---|
| FE00h–FE3Fh | Reserved |
| FE40h–FE53h | Part Number<br>20-character ASCII alphanumeric code<br>Left-justified and filled with zeros |
| FE54h–FFFFh | Reserved |

# Operation

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

## Timing Using the Flash Frequency Registers

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

The Flash Frequency High and Low Byte registers combine to form a 16-bit value, FFREQ, to control timing for Flash program and erase operations. The 16-bit Flash Frequency value must contain the system clock frequency in kHz. This value is calculated using the following equation:.

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

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

## Flash Read Protection

The user code contained within Flash memory can be protected from external access. Programming the Flash Read Protect option bit prevents reading of user code by the On-Chip Debugger or by using the Flash Controller Bypass Mode. For more information, see the Option Bits chapter on page 180 and the On-Chip Debugger chapter on page 183.

## Flash Write/Erase Protection

The Z8 Encore! XP F64xx Series provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by the Flash Controller unlock mechanism, the Flash Sector Protect Register, and the Flash Write Protect option bit.

### Flash Controller Unlock Mechanism

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

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

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

### Flash Sector Protection

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

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

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

1. Write 00h to the Flash Control Register to reset the Flash Controller.
2. Write 5Eh to the Flash Control Register to select the Flash Sector Protect Register.
3. Read and/or write the Flash Sector Protect Register which is now at register file address FF9h.
4. Write 00h to the Flash Control Register to return the Flash Controller to its reset state.

### Flash Write Protection Option Bit

The Flash Write Protect option bit can be enabled to block all program and erase operations from user code. For more information, see the Option Bits chapter on page 180.

## Byte Programming

When the Flash Controller is unlocked, writes to Flash memory from user code will program a byte into the Flash if the address is located in the unlocked page. An erased Flash byte contains all ones (FFh). The programming operation can only be used to change bits from one to zero. To change a Flash bit (or multiple bits) from zero to one requires a Page Erase or Mass Erase operation.

Byte programming can be accomplished using the eZ8 CPU's LDC or LDCI instructions. For a description of the LDC and LDCI instructions, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download on www.zilog.com.

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

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

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

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

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

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

## Page Erase

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

Observe the following procedure to perform a Page Erase operation:

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

## Mass Erase

The Flash memory cannot be mass-erased by user code.

## Flash Controller Bypass

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

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

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

### Flash Controller Behavior in Debug Mode

The following changes in Flash Controller behavior occur when the Flash Controller is accessed using the On-Chip Debugger:

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

**Caution:** For security reasons, the Flash Controller allows only a single page to be opened for write/erase operations. When writing multiple Flash pages, the Flash Controller must go through the unlock sequence again to select another page.

## Flash Control Register Definitions

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

Flash Control Register: see page 175

Flash Status Register: see page 177

Page Select Register: see page 177

Flash Sector Protect Register: see page 178

Flash Frequency High and Low Byte Registers: see page 179

### Flash Control Register

The Flash Control Register, shown in Table 93, unlocks the Flash Controller for programming and erase operations, or to select the Flash Sector Protect Register.

The write-only Flash Control Register shares its register file address with the read-only Flash Status Register.

**Table 93. Flash Control Register (FCTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | FCMD | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | W | | | | | | | |
| Address | FF8h | | | | | | | |

| Bit | Description |
|---|---|
| [7:0]<br>FCMD | **Flash Command***<br>73h = First unlock command.<br>8Ch = Second unlock command.<br>95h = Page erase command.<br>63h = Mass erase command<br>5Eh = Flash Sector Protect Register select. |

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

## Flash Status Register

The Flash Status Register, shown in Table 94, 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 94. Flash Status Register (FSTAT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | FSTAT | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | |
| Address | FF8h | | | | | | | |

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

## Page Select Register

The Page Select (FPS) Register, shown in Table 95, selects one of the 128 available Flash memory pages to be erased or programmed. Each Flash page contains 512 bytes of Flash memory. During a Page Erase operation, all Flash memory locations with the 7 most significant bits of the address provided by the PAGE field are erased to FFh.

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

**Table 95. Page Select Register (FPS)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | INFO_EN | PAGE | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FF9h | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>INFO_EN | **Information Area Enable**<br>0 = Information Area is not selected.<br>1 = Information Area is selected. The Information area is mapped into the Flash memory address space at addresses FE00h through FFFFh. |
| [6:0]<br>PAGE | **Page Select**<br>This 7-bit field selects the Flash memory page for programming and Page Erase operations. Flash Memory Address[15:9] = PAGE[6:0]. |

## Flash Sector Protect Register

The Flash Sector Protect Register, shown in Table 96, protects Flash memory sectors from being programmed or erased from user code. The Flash Sector Protect Register shares its register file address with the Page Select Register. The Flash Sector Protect Register can be accessed only after writing the Flash Control Register with 5Eh.

User code can only write bits in this register to 1 (bits cannot be cleared to 0 by user code). To determine the appropriate Flash memory sector address range and sector number for your Z8F64xx Series product, please refer to Table 91 on page 169.

**Table 96. Flash Sector Protect Register (FPROT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SECT7 | SECT6 | SECT5 | SECT4 | SECT3 | SECT2 | SECT1 | SECT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF9h | | | | | | | |

Note: *R/W = This register is accessible for read operations; it can be written to 1 only via user code.

| Bit | Description |
|---|---|
| [7:0]<br>SECT*n* | **Sector Protect****<br>0 = Sector *n* can be programmed or erased from user code.<br>1 = Sector *n* is protected and cannot be programmed or erased from user code. |

Note: **User code can only write bits from 0 to 1.

## Flash Frequency High and Low Byte Registers

The Flash Frequency High and Low Byte registers, shown in Tables 97 and 98, combine to form a 16-bit value, FFREQ, to control timing for Flash program and erase operations. The 16-bit Flash Frequency registers must be written with the system clock frequency in kHz for Program and Erase operations. Calculate the Flash Frequency value using the following equation:

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

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

**Table 97. Flash Frequency High Byte Register (FFREQH)**

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

**Table 98. Flash Frequency Low Byte Register (FFREQL)**

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

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

# Option Bits

Option bits allow user configuration of certain aspects of the Z8 Encore! XP F64xx Series operation. The feature configuration data is stored in the Flash memory and read during Reset. The features available for control via the option bits are:

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

## Operation

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

### Option Bit Configuration By Reset

Each time the option bits are programmed or erased, the device must be Reset for the change to take place. During any reset operation (System Reset, Reset, or Stop Mode Recovery), the option bits are automatically read from the Flash memory and written to Option Configuration registers. The Option Configuration registers control operation of the devices within the Z8 Encore! XP F64xx Series. Option bit control is established before the device exits Reset and the eZ8 CPU begins code execution. The Option Configuration registers are not part of the register file and are not accessible for read or write access.

## Option Bit Address Space

The first two bytes of Flash memory at addresses 0000h (see Table 99) and 0001h (see Table 100) are reserved for the user option bits. The byte at Flash memory address 0000h configures user options. The byte at Flash memory address 0001h is reserved for future use and must remain unprogrammed.

## Flash Memory Address 0000h

**Table 99. Flash Option Bits At Flash Memory Address 0000h**

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

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

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

| Bit | Description (Continued) |
|---|---|
| [1] | **Reserved**<br>This bit is reserved and must be programmed to 0. |
| [0]<br>FWP | **Flash Write Protect (Flash version only)**<br>0 = Programming, Page Erase, and Mass Erase through User Code is disabled. Mass Erase is available through the On-Chip Debugger.<br>1 = Programming, and Page Erase are enabled for all of Flash program memory. |

## Flash Memory Address 0001h

**Table 100. Options Bits at Flash Memory Address 0001h**

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

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

| Bit | Description |
|---|---|
| [7:0] | **Reserved**<br>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

The Z8 Encore! XP F64xx Series products contain 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
- Execution of eZ8 CPU instructions

## Architecture

The On-Chip Debugger consists of four primary functional blocks: transmitter, receiver, autobaud generator, and debug controller. Figure 36 displays the architecture of the On-Chip Debugger.

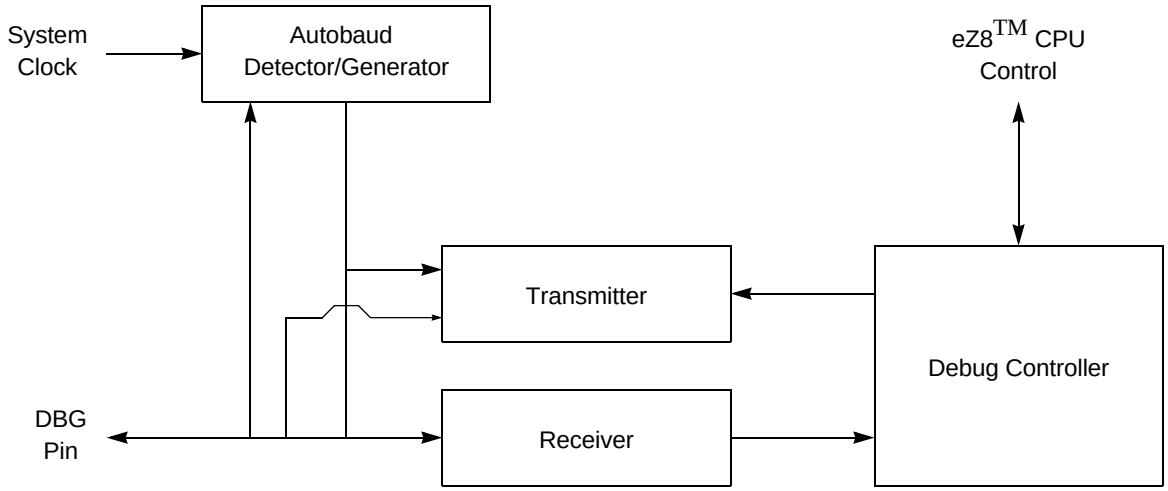



**Figure 36. On-Chip Debugger Block Diagram**

# Operation

The following section describes the operation of the OCD.

## OCD Interface

The On-Chip Debugger uses the DBG pin for communication with an external host. This one-pin interface is a bidirectional open-drain interface that transmits and receives data. Data transmission is half-duplex, meaning that transmit and receive operations cannot occur simultaneously. The serial data on the DBG pin is sent using the standard asynchronous data format defined in RS-232. This pin can interface the Z8 Encore! XP F64xx Series products to the serial port of a host PC using minimal external hardware.Two different methods for connecting the DBG pin to an RS-232 interface are depicted in Figures 37 and 38.

**Caution:** For proper 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 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.

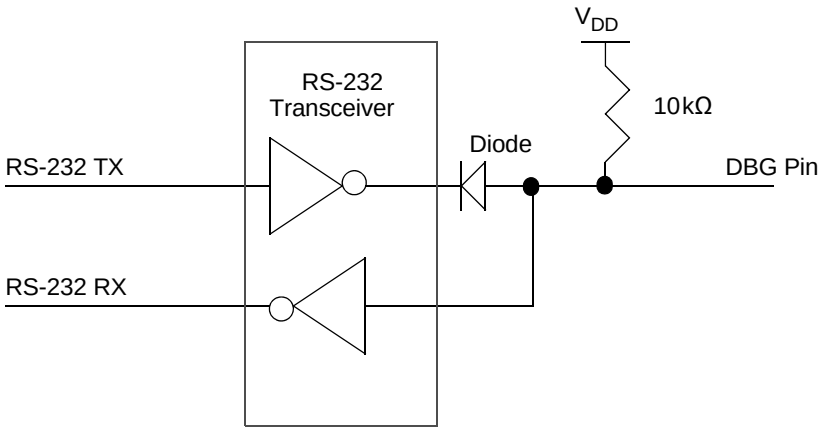



**Figure 37. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #1 of 2**

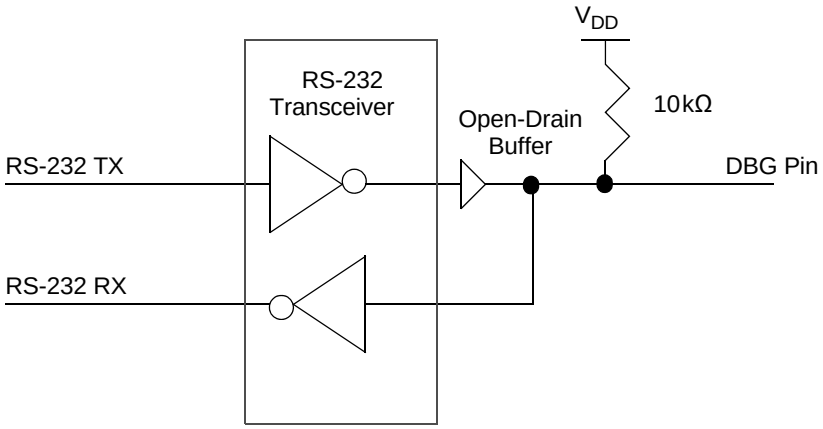



**Figure 38. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #2 of 2**

## Debug Mode

The operating characteristics of the Z8 Encore! XP F64xx Series devices in Debug Mode are:

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

### Entering Debug Mode

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

- Writing the DBGMODE bit in the OCD Control Register to 1 using the OCD interface
- eZ8 CPU execution of a breakpoint (BRK) instruction (when enabled)
- If the DBG pin is Low when the 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 Brown-Out reset
- Asserting the $\overline{\text{RESET}}$ pin Low to initiate a Reset
- Driving the DBG pin Low while the device is in Stop Mode initiates a system reset

## OCD Data Format

The OCD interface uses the asynchronous data format defined for RS-232. Each character is transmitted as 1 start bit, 8 data bits (least significant bit first), and 1 stop bit, as shown in Figure 39.

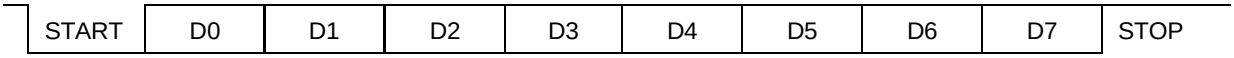



**Figure 39. OCD Data Format**

## OCD Autobaud Detector/Generator

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

The Autobaud Detector/Generator is clocked by the system clock. The minimum baud rate is the system clock frequency divided by 512. For optimal operation, the maximum recommended baud rate is the system clock frequency divided by 8. The theoretical maximum baud rate is the system clock frequency divided by 4. This theoretical maximum is possible for low noise designs with clean signals. Table 101 lists minimum and recommended maximum baud rates for sample crystal frequencies.

**Table 101. 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 Autobaud Detector/Generator resets. The Autobaud 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 serial break 4096 system clock cycles long back to the host, and resets the Autobaud 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 back to the host only extends the length of the serial break if the host releases the serial break early.

The host transmits a serial break on the DBG pin when first connecting to the Z8 Encore! XP F64xx Series devices or when recovering from an error. A serial break from the host resets the Autobaud Generator/Detector but does not reset the OCD Control Register. A serial break leaves the device in Debug Mode if that is the current mode. The OCD is held in Reset until the end of the serial break when the DBG pin returns High. Because of the open-drain nature of the DBG pin, the host can send a serial break to the OCD even if the OCD is transmitting a character.

## Breakpoints

Execution breakpoints are generated using the BRK instruction (op code 00h). When the eZ8 CPU decodes a BRK instruction, it signals the On-Chip Debugger. If breakpoints are enabled, the OCD idles the eZ8 CPU and enters Debug Mode. If breakpoints are not enabled, the OCD ignores the BRK signal and the BRK instruction operates as an NOP.

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

The loop on BRK instruction can be used to service interrupts in the background. For interrupts to be serviced in the background, there cannot be any breakpoints in the interrupt service routine. Otherwise, the CPU stops on the breakpoint in the interrupt routine. For interrupts to be serviced in the background, interrupts must also be enabled. Debugging software should not automatically enable interrupts when using this feature, since

interrupts are typically disabled during critical sections of code where interrupts should not occur (such as adjusting the stack pointer or modifying shared data).

Software can poll the IDLE bit of the OCDSTAT Register to determine if the OCD is looping on a BRK instruction. When software stops the CPU on the BRK instruction that it is looping on, it should not set the DBGMODE bit of the OCDCTL Register. The CPU may have vectored to and be in the middle of an interrupt service routine when this bit gets set. Instead, software must clear the BRKLP bit. This action allows the CPU to finish the interrupt service routine it may be in and return the BRK instruction. When the CPU returns to the BRK instruction it was previously looping on, it automatically sets the DBGMODE bit and enters Debug Mode.

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

### Breakpoints in Flash Memory

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

## On-Chip Debugger Commands

The host communicates to the On-Chip Debugger by sending OCD commands using the DBG interface. During normal operation, only a subset of the OCD commands are available. In Debug Mode, all OCD commands become available unless the user code and control registers are protected by programming the Read Protect option bit (RP). The Read Protect option bit prevents the code in memory from being read out of the Z8 Encore! XP F64xx Series products. When this option is enabled, several of the OCD commands are disabled.

Table 102 contains a summary of the On-Chip Debugger commands. Table 102 lists those commands that operate when the device is not in Debug Mode (normal operation) and those commands that are disabled by programming the Read Protect option bit.

Each OCD command is further described in the list that follows the table.

**Table 102. 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 | — |
| 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 |
| Write Program Memory | 0Ah | — | Disabled |
| Read Program Memory | 0Bh | — | Disabled |
| Write Data Memory | 0Ch | — | Disabled |
| Read Data Memory | 0Dh | — | Disabled |
| Read Program Memory CRC | 0Eh | — | — |
| Reserved | 0Fh | — | — |
| Step Instruction | 10h | — | Disabled |
| Stuff Instruction | 11h | — | Disabled |
| Execute Instruction | 12h | — | Disabled |
| Reserved | 13h–FFh | — | — |

In the following 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 determines 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 reads the OCDSTAT Register.

```
DBG ← 02h
DBG OCDSTAT[7:0]
```

**Write OCD Control Register (04h).** The Write OCD Control Register command writes the data that follows to the OCDCTL Register. When the Read Protect option bit is enabled, the DBGMODE bit (OCDCTL[7]) can only be set to 1, it cannot be cleared to 0 and the only method of putting the device back into normal operating mode is to reset the device.

```
DBG ← 04h
DBG ← OCDCTL[7:0]
```

**Read OCD Control Register (05h).** The Read OCD Control Register command reads the value of the OCDCTL Register.

```
DBG ← 05h
DBG OCDCTL[7:0]
```

**Write Program Counter (06h).** The Write Program Counter command writes the data that follows to the eZ8 CPU's program counter (PC). If the device is not in Debug Mode or if the Read Protect option bit is enabled, the PC values are discarded.

```
DBG ← 06h
DBG ← ProgramCounter[15:8]
DBG ← ProgramCounter[7:0]
```

**Read Program Counter (07h).** The Read Program Counter command reads the value in the eZ8 CPU's program counter (PC). If the device is not in Debug Mode or if the Read Protect option bit is enabled, this command returns FFFFh.

```
DBG ← 07h
DBG ProgramCounter[15:8]
DBG ProgramCounter[7:0]
```

**Write Register (08h).** The Write Register command writes data to the register file. Data can be written 1-256 bytes at a time (256 bytes can be written by setting size to zero). If the device is not in Debug Mode, the address and data values are discarded. If the Read Protect option bit is enabled, then only writes to the Flash Control Registers are allowed and all other register write data values are discarded.

```
DBG ← 08h
DBG ← {4'h0,Register Address[11:8]}
DBG ← Register Address[7:0]
DBG ← Size[7:0]
DBG ← 1-256 data bytes
```

**Read Register (09h).** The Read Register command reads data from the register file. Data can be read 1-256 bytes at a time (256 bytes can be read by setting size to zero). If the device is not in Debug Mode or if the Read Protect option bit is enabled, this command returns FFh for all the data values.

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

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

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

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

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

**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 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 device is not in Debug Mode, this command returns FFh for the data.

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

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

```
DBG ← 0Eh
DBG CRC[15:8]
DBG CRC[7:0]
```

**Step Instruction (10h).** The Step Instruction command steps one assembly instruction at the current program counter (PC) location. If the device is not in Debug Mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 10h
```

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

```
DBG ← 11h
DBG ← opcode[7:0]
```

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

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

# On-Chip Debugger Control Register Definitions

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

## OCD Control Register

The OCD Control Register, shown in Table 103, controls the state of the On-Chip Debugger. This register enters or exits Debug Mode and enables the BRK instruction.

A *reset and stop* function can be achieved by writing 81h to this register. A *reset and go* function can be achieved by writing 41h to this register. If the device is operating in Debug Mode, a *run* function can be implemented by writing 40h to this register.

**Table 103. OCD Control Register (OCDCTL)**

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

| Bit | Description |
|---|---|
| [7]<br>DBGMODE | **Debug Mode**<br>Setting this bit to 1 causes the device to enter Debug Mode. When in Debug Mode, the eZ8 CPU stops fetching new instructions. Clearing this bit causes the eZ8 CPU to start running again. This bit is automatically set when a BRK instruction is decoded and breakpoints are enabled. If the Read Protect option bit is enabled, this bit can only be cleared by resetting the device, it cannot be written to 0.<br>0 = TheZ8 Encore! XP F64xx Series device is operating in Normal Mode.<br>1 = The Z8 Encore! XP F64xx Series device is in Debug Mode. |
| [6]<br>BRKEN | **Breakpoint Enable**<br>This bit controls the behavior of the BRK instruction (op code 00h). By default, breakpoints are disabled and the BRK instruction behaves like a NOP. If this bit is set to 1 and a BRK instruction is decoded, the OCD takes action dependent upon the BRKLOOP bit.<br>0 = BRK instruction is disabled.<br>1 = BRK instruction is enabled. |
| [5]<br>DBGACK | **Debug Acknowledge**<br>This bit enables the debug acknowledge feature. If this bit is set to 1, then the OCD sends an Debug Acknowledge character (FFh) to the host when a breakpoint occurs.<br>0 = Debug Acknowledge is disabled.<br>1 = Debug Acknowledge is enabled. |

| Bit | Description (Continued) |
|---|---|
| [4]<br>BRKLOOP | **Breakpoint Loop**<br>This bit determines what action the OCD takes when a BRK instruction is decoded if breakpoints are enabled (BRKEN is 1). If this bit is 0, then the DBGMODE bit is automatically set to 1 and the OCD entered Debug Mode. If BRKLOOP is set to 1, then the eZ8 CPU loops on the BRK instruction.<br>0 = BRK instruction sets DBGMODE to 1.<br>1 = eZ8 CPU loops on BRK instruction. |
| [3:1] | **Reserved**<br>These bits are reserved and must be programmed to 000. |
| [0]<br>RST | **Reset**<br>Setting this bit to 1 resets the Z8 Encore! XP F64xx Series devices. The devices go 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.<br>0 = No effect.<br>1 = Reset the Z8 Encore! XP F64xx Series device. |

## OCD Status Register

The OCD Status Register, shown in Table 104, reports status information about the current state of the debugger and the system.

**Table 104. OCD Status Register (OCDSTAT)**

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

| Bit | Description |
|---|---|
| [7]<br>IDLE | **CPU Idle**<br>This bit is set if the part is in Debug Mode (DBGMODE is 1), or if a BRK instruction occurred since the last time OCDCTL was written. This can be used to determine if the CPU is running or if it is idling.<br>0 = The eZ8 CPU is running.<br>1 = The eZ8 CPU is either stopped or looping on a BRK instruction. |
| [6]<br>HALT | **Halt Mode**<br>0 = The device is not in Halt Mode.<br>1 = The device is in Halt Mode. |

| Bit | Description (Continued) |
|---|---|
| [5]<br>RPEN | **Read Protect Option Bit Enabled**<br>0 = The Read Protect option bit is disabled (1).<br>1 = The Read Protect option bit is enabled (0), disabling many OCD commands. |
| [4:0] | **Reserved**<br>These bits are reserved and must be programmed to 00000. |

# On-Chip Oscillator

The products in the Z8 Encore! XP F64xx Series feature an on-chip oscillator for use with external crystals with frequencies from 32kHz to 20MHz. In addition, the oscillator can support external RC networks with oscillation frequencies up to 4MHz or ceramic resonators with oscillation frequencies up to 20MHz. This oscillator generates the primary system clock for the internal eZ8 CPU and the majority of the on-chip peripherals. Alternatively, the $X_{IN}$ input pin can also accept a CMOS-level clock input signal (32kHz–20MHz). If an external clock generator is used, the $X_{OUT}$ pin must be left unconnected.

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

## Operating Modes

The Z8 Encore! XP F64xx Series products support four different oscillator modes:

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

The oscillator mode is selected through user-programmable option bits. For more information, see the Option Bits chapter on page 180.

## Crystal Oscillator Operation

Figure 40 displays a recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 20MHz. Recommended 20MHz crystal specifications are provided in Table 105. Resistor R1 is optional and limits total power dissipation by the crystal. The printed circuit board layout must add no more than 4 pF of stray capacitance to either the $X_{IN}$ or $X_{OUT}$ pins. If oscillation does not occur, reduce the values of capacitors C1 and C2 to decrease loading.

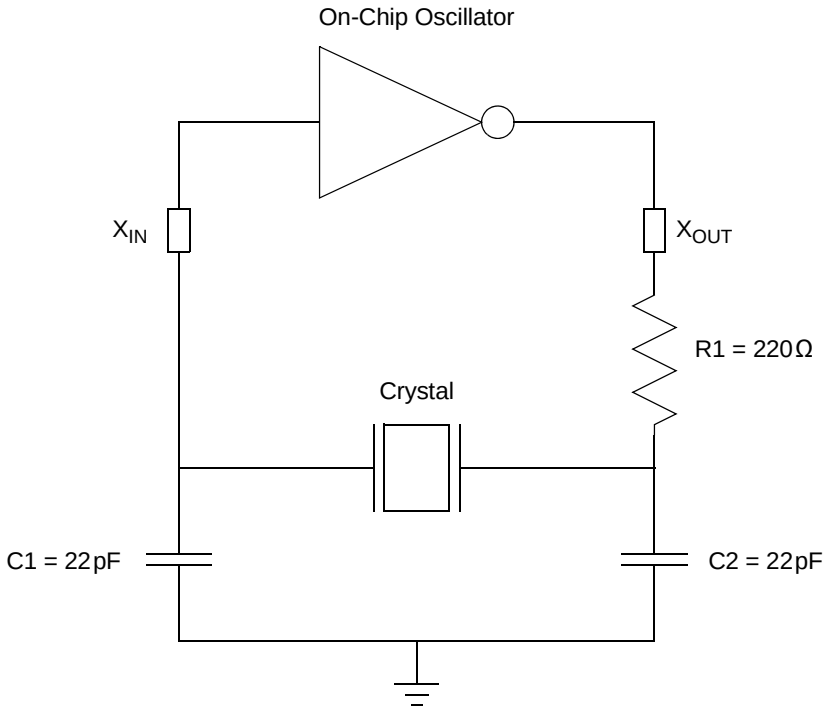



**Figure 40. Recommended 20MHz Crystal Oscillator Configuration**

**Table 105. Recommended Crystal Oscillator Specifications (20MHz Operation)**

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

# Oscillator Operation with an External RC Network

The External RC Oscillator Mode is applicable to timing-insensitive applications. Figure 41 displays a recommended configuration for connection with an external resistor-capacitor (RC) network.

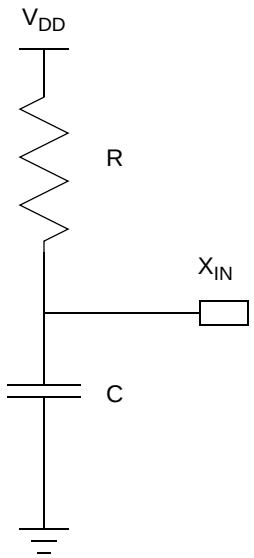



**Figure 41. Connecting the On-Chip Oscillator to an External RC Network**

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

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

Figure 42 displays the typical (3.3V and 25°C) oscillator frequency as a function of the capacitor (*C* in pF) employed in the RC network assuming a 45kΩ external resistor. For very small values of C, the parasitic capacitance of the oscillator $X_{IN}$ pin and the printed circuit board should be included in the estimation of the oscillator frequency.

It is possible to operate the RC oscillator using only the parasitic capacitance of the package and printed circuit board. To minimize sensitivity to external parasitics, external capacitance values in excess of 20pF are recommended.

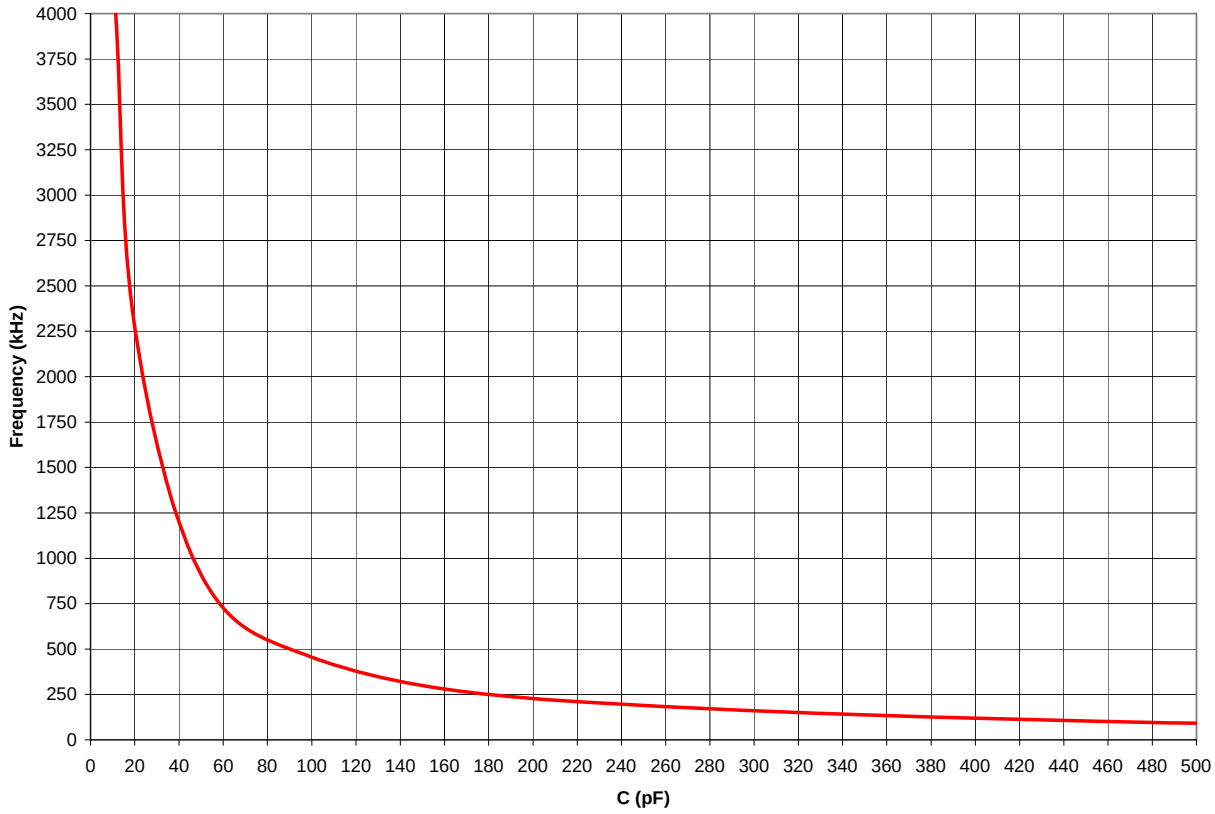



**Figure 42. Typical RC Oscillator Frequency as a Function of the External Capacitance with a 45kΩ Resistor**

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

# Electrical Characteristics

The data in this chapter represents all known data prior to qualification and characterization of the Z8 Encore! XP F64xx Series of products, and is therefore subject to change. Additional electrical characteristics may be found in the individual chapters of this document.

## Absolute Maximum Ratings

Stresses greater than those listed in Table 106 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 106. Absolute Maximum Ratings**

| Parameter | Minimum | Maximum | Units | Notes |
|---|---|---|---|---|
| Ambient temperature under bias | –40 | +125 | 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 125°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 | |
| 68-pin PLCC maximum ratings at 70°C to 125°C | | | | |
| Total power dissipation | | 500 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 140 | mA | |

**Table 106. Absolute Maximum Ratings (Continued)**

| Parameter | Minimum | Maximum | Units | Notes |
|---|---|---|---|---|
| 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°C to 125°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°C to 125°C | | | | |
| Total power dissipation | | 295 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 83 | mA | |
| **44-pin LQFP maximum ratings at –40°C to 70°C** | | | | |
| Total power dissipation | | 750 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 200 | mA | |
| **44-pin LQFP maximum ratings at 70°C to 125°C** | | | | |
| Total power dissipation | | 360 | mW | |
| Maximum current into $V_{DD}$ or out of $V_{SS}$ | | 100 | mA | |

Note: This voltage applies to all pins, with the exception of $V_{DD}$, $AV_{DD}$, pins supporting analog input (ports B and H), $\overline{\text{RESET}}$, and where noted otherwise.

# DC Characteristics

Table 107 lists the DC characteristics of the Z8 Encore! XP F64xx Series products. All voltages are referenced to $V_{SS}$, the primary system ground.

**Table 107. DC Characteristics**

| | | $T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical** | **Maximum** | **Units** | **Conditions** |
| $V_{DD}$ | 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 $\overline{RESET}$, DBG, $X_{IN}$ |
| $V_{IL2}$ | Low Level Input Voltage | –0.3 | – | $0.2*V_{DD}$ | V | For $\overline{RESET}$, DBG, and $X_{IN}$. |
| $V_{IH1}$ | 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 | $\overline{RESET}$, DBG, and $X_{IN}$ pins |
| $V_{OL1}$ | Low Level Output Voltage Standard Drive | – | – | 0.4 | V | $I_{OL}$ = 2 mA; $V_{DD}$ = 3.0V High Output Drive disabled. |
| $V_{OH1}$ | High Level Output Voltage Standard Drive | 2.4 | – | – | V | $I_{OH}$ = –2 mA; $V_{DD}$ = 3.0V High Output Drive disabled. |
| $V_{OL2}$ | Low Level Output Voltage High Drive | – | – | 0.6 | V | $I_{OL}$ = 20 mA; $V_{DD}$ = 3.3V High Output Drive enabled $T_A$ = –40°C to +70°C |
| $V_{OH2}$ | High Level Output Voltage High Drive | 2.4 | – | – | V | $I_{OH}$ = –20 mA; $V_{DD}$ = 3.3V High Output Drive enabled; $T_A$ = –40°C to +70°C |
| $V_{OL3}$ | Low Level Output Voltage High Drive | – | – | 0.6 | V | $I_{OL}$ = 15 mA; $V_{DD}$ = 3.3V High Output Drive enabled; $T_A$ = +70°C to +105°C |

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

**Table 107. DC Characteristics (Continued)**

| | | $T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical** | **Maximum** | **Units** | **Conditions** |
| $V_{OH3}$ | High Level Output Voltage High Drive | 2.4 | – | – | V | $I_{OH}$ = 15 mA; $V_{DD}$ = 3.3V High Output Drive enabled; $T_A$ = +70°C to +105°C |
| $V_{RAM}$ | RAM Data Retention | 0.7 | – | – | V | |
| $I_{IL}$ | Input Leakage Current | –5 | – | +5 | µA | $V_{DD}$ = 3.6V; $V_{IN}$ = $V_{DD}$ or $V_{SS}$[1] |
| $I_{TL}$ | Tri-State Leakage Current | –5 | – | +5 | µA | $V_{DD}$ = 3.6V |
| $C_{PAD}$ | GPIO Port Pad Capacitance | – | 8.0[2] | – | pF | |
| $C_{XIN}$ | $X_{IN}$ Pad Capacitance | – | 8.0[2] | – | pF | |
| $C_{XOUT}$ | $X_{OUT}$ Pad Capacitance | – | 9.5[2] | – | pF | |
| $I_{PU}$ | Weak Pull-up Current | 30 | 100 | 350 | µA | $V_{DD}$ = 3.0–3.6V |
| $I_{DDA}$ | Active Mode Supply Current; GPIO pins are configured as outputs (see Figure 43 on page 205 and Figure 44 on page 206) | – | 11 | 16 | mA | $V_{DD}$ = 3.6V, $F_{SYSCLK}$ = 20MHz |
| | | – | – | 12 | mA | $V_{DD}$ = 3.3V |
| | | – | 9 | 11 | mA | $V_{DD}$ = 3.6V, $F_{SYSCLK}$ = 10MHz |
| | | – | – | 9 | mA | $V_{DD}$ = 3.3V |
| $I_{DDH}$ | Halt Mode Supply Current; GPIO pins configured as outputs (see Figure 45 on page 207 and Figure 46 on page 208) | – | 4 | 7 | mA | $V_{DD}$ = 3.6V, $F_{SYSCLK}$ = 20MHz |
| | | – | – | 5 | mA | $V_{DD}$ = 3.3V |
| | | – | 3 | 5 | mA | $V_{DD}$ = 3.6V, $F_{SYSCLK}$ = 10MHz |
| | | – | – | 4 | mA | $V_{DD}$ = 3.3V |

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

**Table 107. DC Characteristics (Continued)**

| | | $T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical** | **Maximum** | **Units** | **Conditions** |
| $I_{DDS}$ | Stop Mode Supply Current; GPIO pins configured as outputs (see Figure 47 on page 209 and Figure 48 on page 210) | – | 520 | | µA | VBO and WDT enabled |
| | | | | 700 | | $V_{DD}$ = 3.6V |
| | | | | 650 | | $V_{DD}$ = 3.3V |
| | | – | 10 | | µA | VBO disabled, WDT enabled, $T_A$ = 0 to 70ºC |
| | | | | 25 | | $V_{DD}$ = 3.6V |
| | | | | 20 | | $V_{DD}$ = 3.3V |
| | | – | – | | µA | VBO disabled, WDT enabled, $T_A$ = –40 to +105ºC |
| | | | | 80 | | $V_{DD}$ = 3.6V |
| | | | | 70 | | $V_{DD}$ = 3.3V |
| | | – | – | | µA | VBO disabled, WDT enabled, $T_A$ = –40 to +125ºC |
| | | | | 250 | | $V_{DD}$ = 3.6V |
| | | | | 150 | | $V_{DD}$ = 3.3V |

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

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

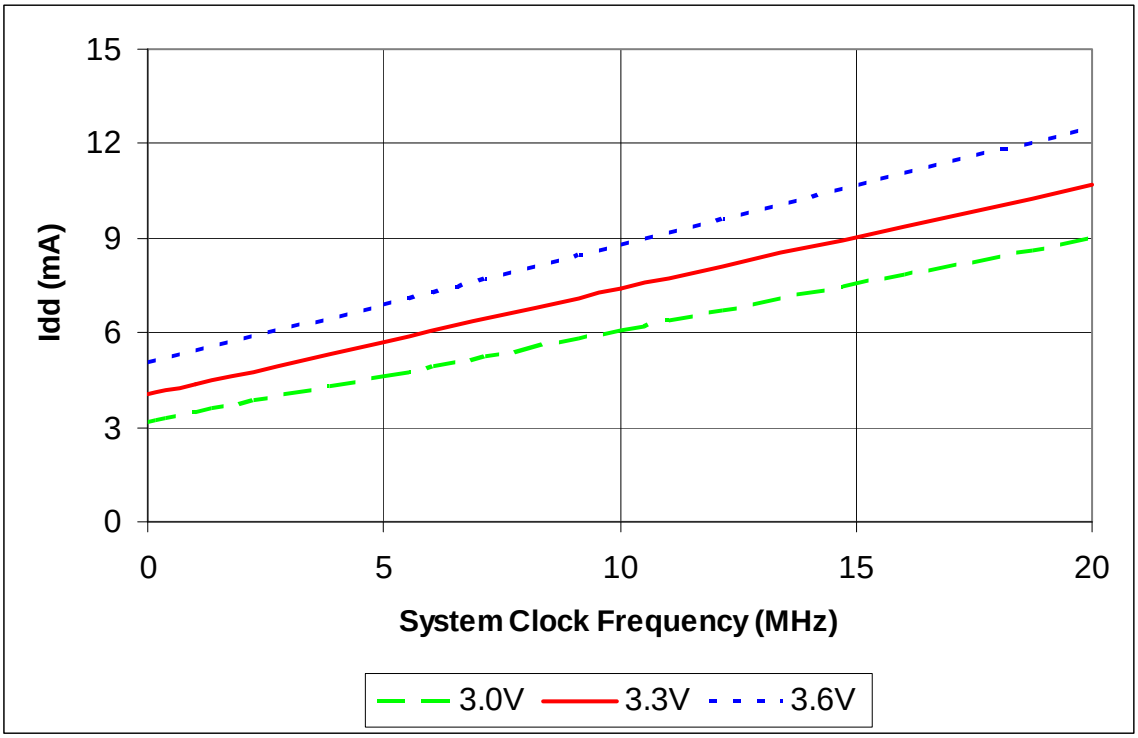



**Figure 43. Typical Active Mode $I_{DD}$ vs. System Clock Frequency**

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

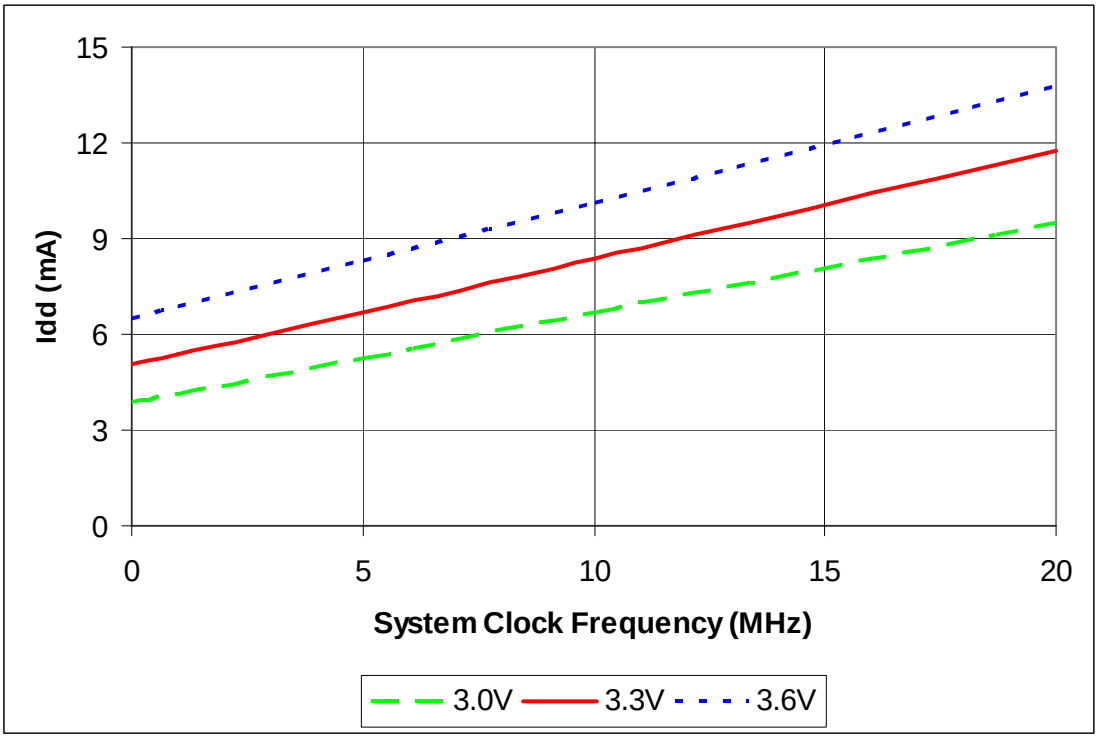



**Figure 44. Maximum Active Mode $I_{DD}$ vs. System Clock Frequency**

Figure 45 displays the typical current consumption in Halt Mode while operating at 25ºC plotted opposite the system clock frequency. All GPIO pins are configured as outputs and driven High.

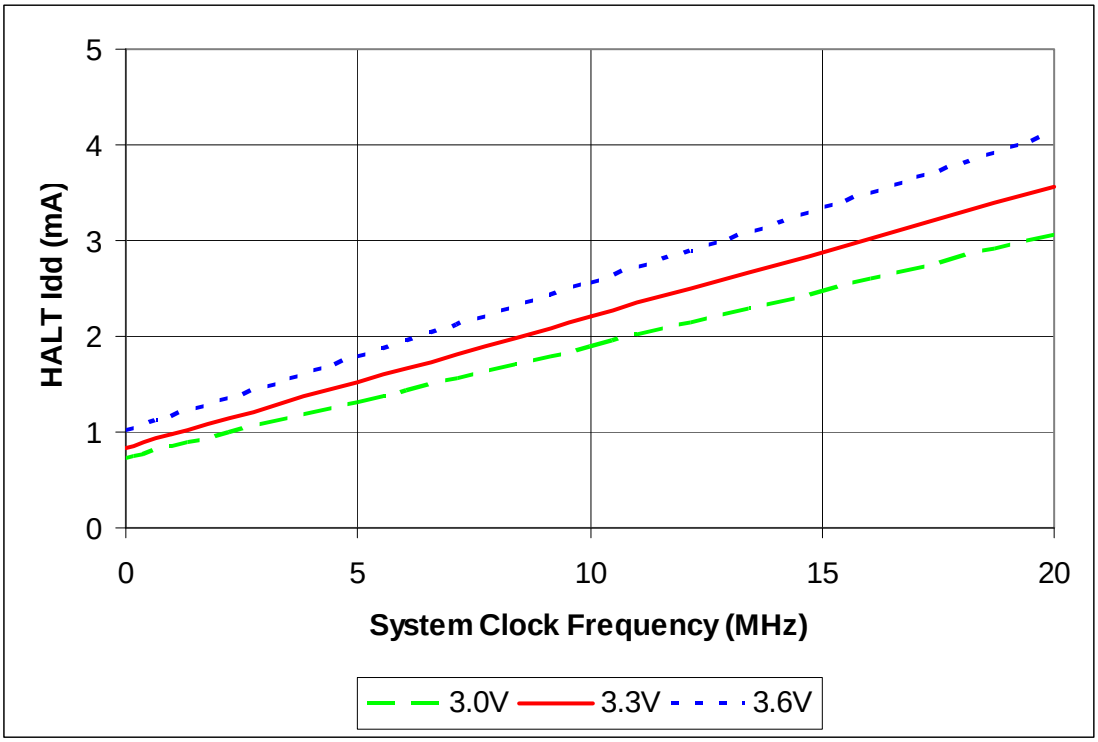



**Figure 45. Typical Halt Mode $I_{DD}$ vs. System Clock Frequency**

Figure 46 displays the maximum Halt Mode current consumption across the full operating temperature range of the device and plotted opposite the system clock frequency. All GPIO pins are configured as outputs and driven High.

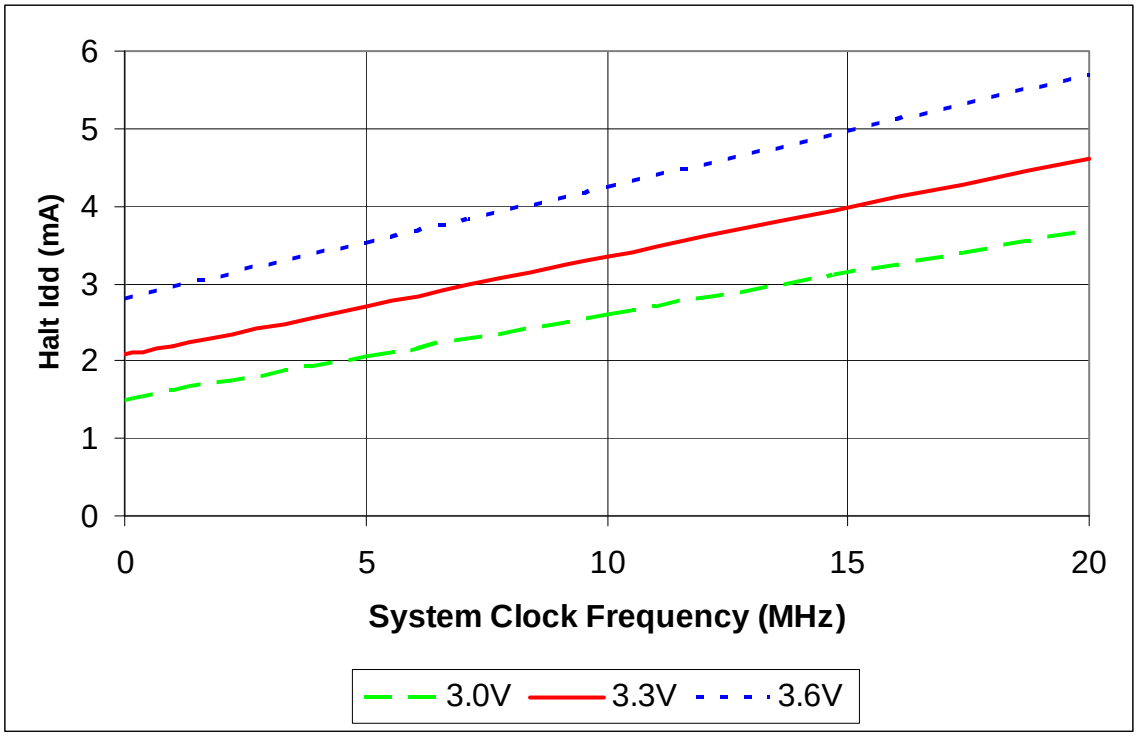



**Figure 46. Maximum Halt Mode $I_{CC}$ vs. System Clock Frequency**

Figure 47 displays the maximum current consumption in Stop Mode with the VBO and Watchdog Timer enabled plotted opposite the power supply voltage. All GPIO pins are configured as outputs and driven High.

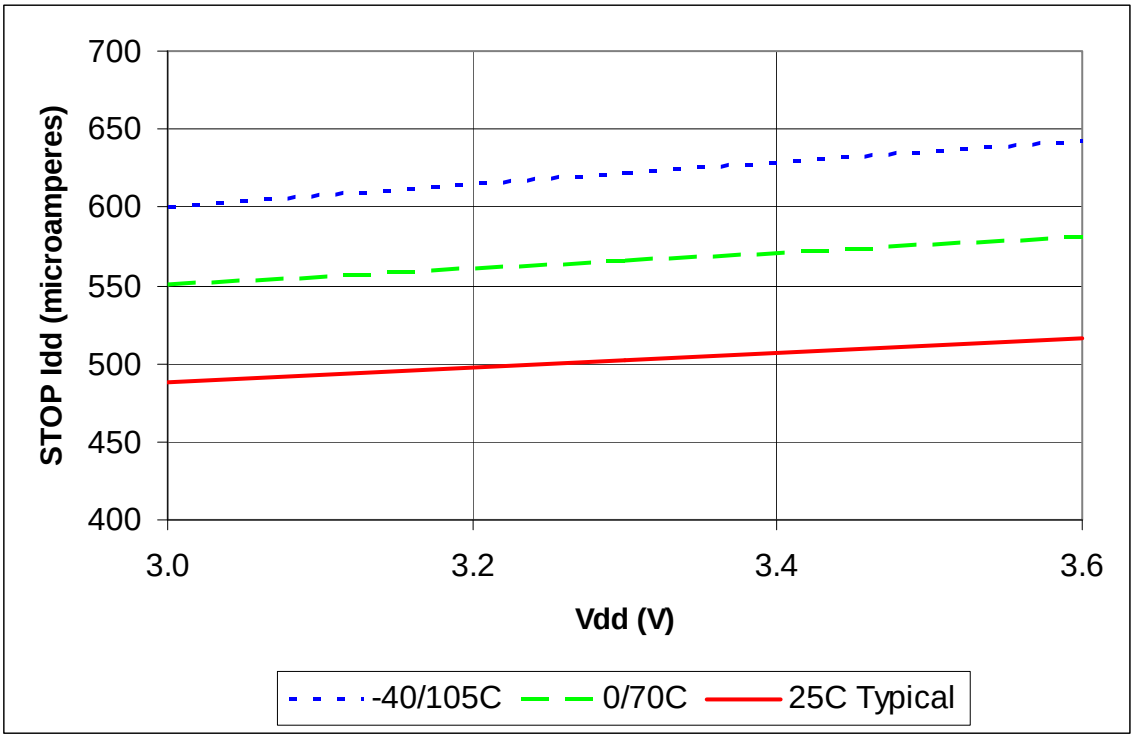



**Figure 47. Maximum Stop Mode $I_{DD}$ with VBO Enabled vs. Power Supply Voltage**

Figure 48 displays the maximum current consumption in Stop Mode with the VBO disabled and Watchdog Timer enabled plotted opposite the power supply voltage. All GPIO pins are configured as outputs and driven High. Disabling the Watchdog Timer and its internal RC oscillator in Stop Mode will provide some additional reduction in Stop Mode current consumption. This small current reduction would be indistinguishable on the scale shown in the figure.

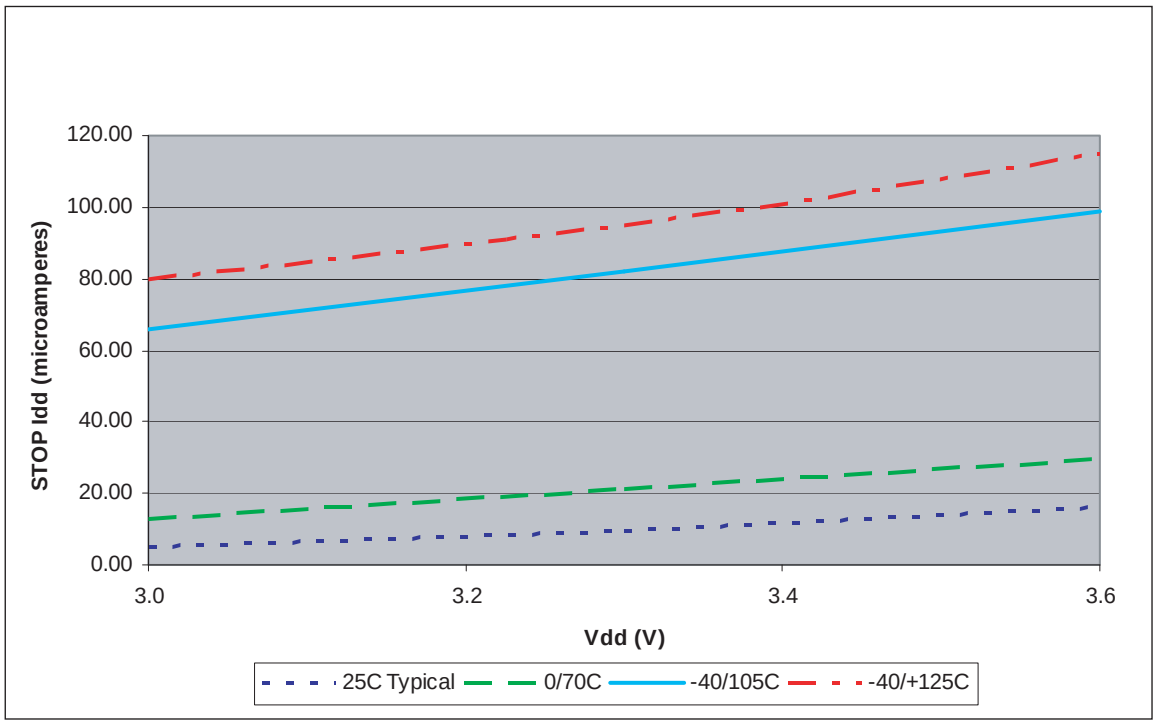



**Figure 48. Maximum Stop Mode $I_{DD}$ with VBO Disabled vs. Power Supply Voltage**

# On-Chip Peripheral AC and DC Electrical Characteristics

**Table 108. Power-On Reset and Voltage Brown-Out Electrical Characteristics and Timing**

| | | $T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical*** | **Maximum** | **Units** | **Conditions** |
| $V_{POR}$ | 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 | |
| $T_{ANA}$ | 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 | – | 6.6 | – | ms | 66 WDT Oscillator cycles (10kHz) + 16 System Clock cycles (20MHz) |
| $T_{VBO}$ | Voltage Brown-Out Pulse Rejection Period | – | 10 | – | µs | $V_{DD} < V_{VBO}$ to generate a Reset. |
| $T_{RAMP}$ | Time for $V_{DD}$ to transition from $V_{SS}$ to $V_{POR}$ to ensure valid Reset | 0.10 | – | 100 | ms | |

Note: *Data in the typical column is from characterization at 3.3V and 0°C. These values are provided for design guidance only and are not tested in production.

**Table 109. External RC Oscillator Electrical Characteristics and Timing**

| | | $T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical*** | **Maximum** | **Units** | **Conditions** |
| $V_{DD}$ | Operating Voltage Range | 2.70[1] | – | – | V | |
| $R_{EXT}$ | External Resistance from $X_{IN}$ to $V_{DD}$ | 40 | 45 | 200 | kΩ | $V_{DD} = V_{VBO}$ |
| $C_{EXT}$ | External Capacitance from $X_{IN}$ to $V_{SS}$ | 0 | 20 | 1000 | pF | |
| $F_{OSC}$ | External RC Oscillation Frequency | – | – | 4 | MHz | |

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

**Table 110. Reset and Stop Mode Recovery Pin Timing**

| | | $T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical** | **Maximum** | **Units** | **Conditions** |
| $T_{RESET}$ | $\overline{\text{RESET}}$ pin assertion to initiate a system reset. | 4 | – | – | $T_{CLK}$ | Not in Stop Mode. $T_{CLK}$ = System Clock period. |
| $T_{SMR}$ | Stop Mode Recovery pin Pulse Rejection Period | 10 | 20 | 40 | ns | RESET, DBG, and GPIO pins configured as SMR sources. |

Table 111 list the Flash memory electrical characteristics and timing.

**Table 111. Flash Memory Electrical Characteristics and Timing**

| | $V_{DD}$ = 3.0–3.6V<br>$T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|
| **Parameter** | **Minimum** | **Typical** | **Maximum** | **Units** | **Notes** |
| Flash Byte Read Time | 50 | – | – | ns | |
| Flash Byte Program Time | 20 | – | 40 | µs | |
| Flash Page Erase Time | 10 | – | – | ms | |
| Flash Mass Erase Time | 200 | – | – | ms | |
| Writes to Single Address Before Next Erase | – | – | 2 | | |
| Flash Row Program Time | – | – | 8 | ms | Cumulative program time for single row cannot exceed limit before next erase. This parameter is only an issue when bypassing the Flash Controller. |
| Data Retention | 100 | – | – | years | 25°C |
| Endurance, –40°C to 105°C | 10,000 | – | – | cycles | Program/erase cycles |
| Endurance, 106°C to 125°C | 1,000 | – | – | cycles | Program/erase cycles |

Table 112 lists the Watchdog Timer electrical characteristics and timing.

**Table 112. Watchdog Timer Electrical Characteristics and Timing**

| | | $V_{DD}$ = 3.0–3.6V<br>$T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical** | **Maximum** | **Units** | **Conditions** |
| $F_{WDT}$ | WDT Oscillator Frequency | 5 | 10 | 20 | kHz | |
| $I_{WDT}$ | WDT Oscillator Current including internal RC Oscillator | – | <1 | 5 | µA | |

Table 113 provides electrical characteristics and timing information for the Analog-to-Digital Converter. Figure 49 displays the input frequency response of the ADC.

**Table 113. Analog-to-Digital Converter Electrical Characteristics and Timing**

| | | $V_{DD}$ = 3.0V–3.6V<br>$T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical** | **Maximum** | **Units** | **Conditions** |
| | Resolution | 10 | – | – | bits | External $V_{REF}$ = 3.0V; |
| | Differential Nonlinearity (DNL) | –1.0 | | +1.0 | lsb | Guaranteed by design |
| | Integral Nonlinearity (INL) | –3.0 | ±1.0 | 3.0 | lsb | External $V_{REF}$ = 3.0V |
| | DC Offset Error | –35 | – | 25 | mV | 80-pin QFP and 64-pin LQFP packages. |
| | DC Offset Error | –50 | – | 25 | mV | 44-pin LQFP, 44-pin PLCC, and 68-pin PLCC packages. |
| $V_{REF}$ | Internal Reference Voltage | 1.9 | 2.0 | 2.4 | V | $V_{DD}$ = 3.0V–3.6V<br>$T_A$ = –40°C to 105°C |
| $VC_{REF}$ | Voltage Coefficient of Internal Reference Voltage | – | 78 | – | mV/V | $V_{REF}$ variation as a function of $AV_{DD}$. |
| $TC_{REF}$ | Temperature Coefficient of Internal Reference Voltage | – | 1 | – | mV/°C | |
| | Single-Shot Conversion Period | – | 5129 | – | cycles | System clock cycles |
| | Continuous Conversion Period | – | 256 | – | cycles | System clock cycles |
| $R_S$ | Analog Source Impedance | – | – | 150 | W | Recommended |
| Zin | Input Impedance | | 150 | | kΩ | 20MHz system clock. Input impedance increases with lower system clock frequency. |
| $V_{REF}$ | External Reference Voltage | | | $AV_{DD}$ | V | $AV_{DD}$ <= $V_{DD}$. When using an external reference voltage, decoupling capacitance should be placed from $V_{REF}$ to $AV_{SS}$. |

**Table 113. Analog-to-Digital Converter Electrical Characteristics and Timing (Continued)**

| | | $V_{DD}$ = 3.0V–3.6V<br>$T_A$ = –40°C to 125°C | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Typical** | **Maximum** | **Units** | **Conditions** |
| $I_{REF}$ | Current draw into $V_{REF}$ pin when driving with external source. | | 25.0 | 40.0 | µA | |

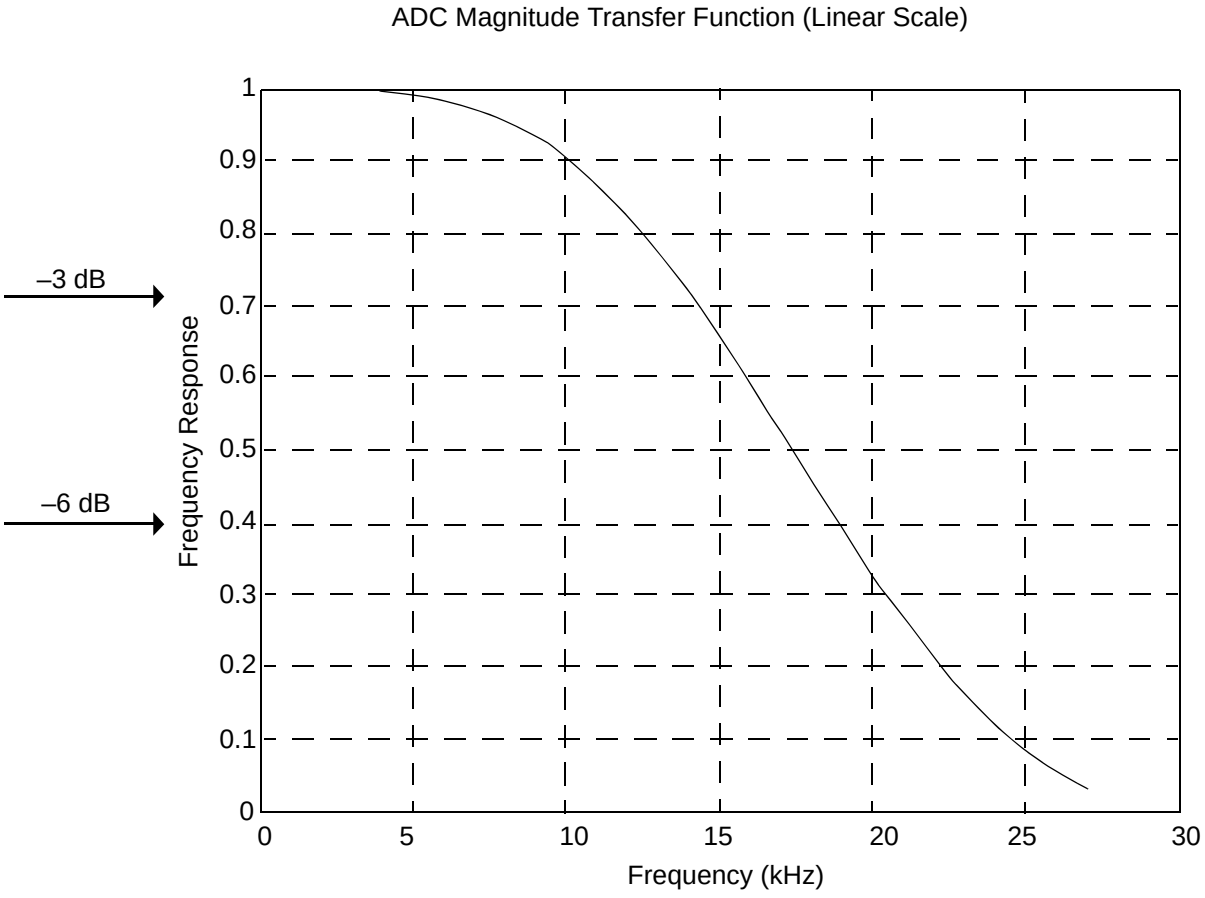



**Figure 49. Analog-to-Digital Converter Frequency Response**

# AC Characteristics

This section provides AC characteristics and timing data which assumes a standard load of 50 pF on all outputs. Table 114 lists the Z8 Encore! XP F64xx Series AC characteristics and timing.

**Table 114. AC Characteristics**

| | | $V_{DD}$ = 3.0 V–3.6 V<br>$T_A$ = –40°C to 125°C | | | |
|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Minimum** | **Maximum** | **Units** | **Conditions** |
| $F_{SYSCLK}$ | System Clock Frequency | – | 20.0 | MHz | Read-only from Flash memory. |
| | | 0.032768 | 20.0 | MHz | Program or erasure of Flash memory. |
| $F_{XTAL}$ | Crystal Oscillator Frequency | 0.032768 | 20.0 | MHz | System clock frequencies below the crystal oscillator minimum require an external clock driver. |
| $T_{XIN}$ | Crystal Oscillator Clock Period | 50 | – | ns | $T_{CLK} = 1/F_{SYSCLK}$ |
| $T_{XINH}$ | System Clock High Time | 20 | | ns | |
| $T_{XINL}$ | System Clock Low Time | 20 | | ns | |
| $T_{XINR}$ | System Clock Rise Time | – | 3 | ns | $T_{CLK}$ = 50 ns. Slower rise times can be tolerated with longer clock periods. |
| $T_{XINF}$ | System Clock Fall Time | – | 3 | ns | $T_{CLK}$ = 50 ns. Slower fall times can be tolerated with longer clock periods. |

## General-Purpose I/O Port Input Data Sample Timing

Figure 50 displays timing of the GPIO Port input sampling. Table 115 lists the GPIO port input timing.

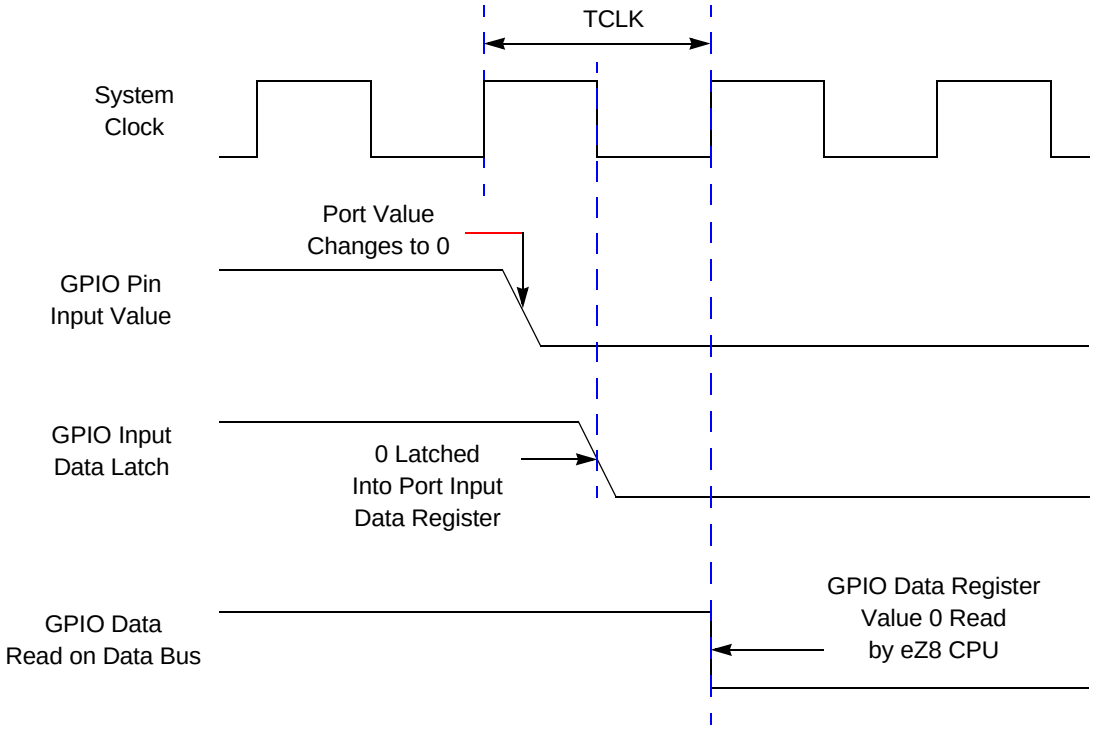



**Figure 50. Port Input Sample Timing**

**Table 115. GPIO Port Input Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Min** | **Max** |
| $T_{S_PORT}$ | Port Input Transition to $X_{IN}$ Fall Setup Time (not pictured) | 5 | – |
| $T_{H_PORT}$ | $X_{IN}$ Fall to Port Input Transition Hold Time (not pictured) | 6 | – |
| $T_{SMR}$ | GPIO Port Pin Pulse Width to Insure Stop Mode Recovery (for GPIO Port pins enabled as SMR sources) | 1 µs | |

## General-Purpose I/O Port Output Timing

Figure 51 and Table 116 provide timing information for GPIO port pins.

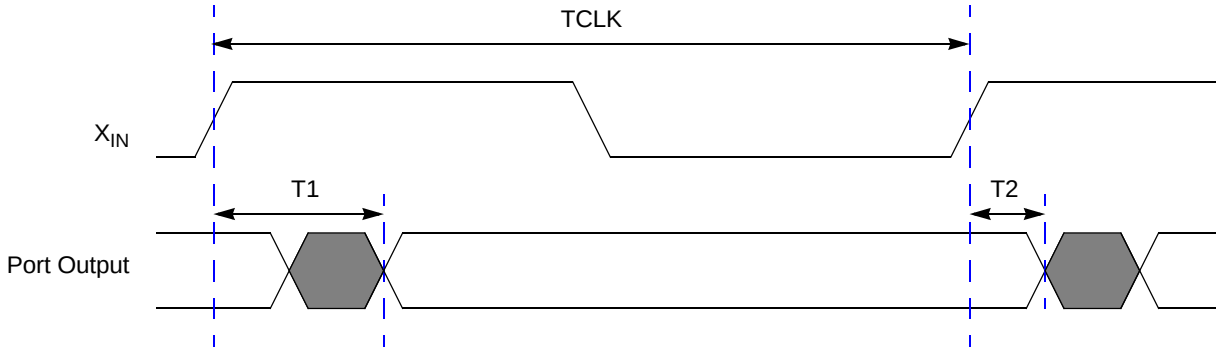



**Figure 51. GPIO Port Output Timing**

**Table 116. GPIO Port Output Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| **GPIO port pins** | | | |
| $T_1$ | $X_{IN}$ Rise to Port Output Valid Delay | – | 20 |
| $T_2$ | $X_{IN}$ Rise to Port Output Hold Time | 2 | – |

## On-Chip Debugger Timing

Figure 52 and Table 117 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4µs maximum rise and fall time.

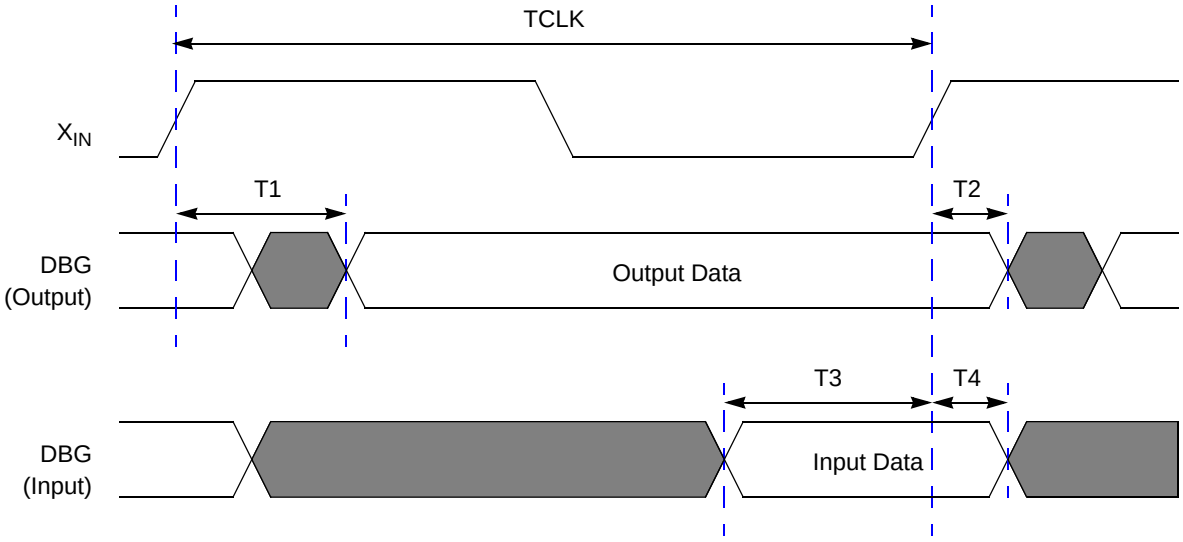



**Figure 52. On-Chip Debugger Timing**

**Table 117. On-Chip Debugger Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| **DBG** | | | |
| $T_1$ | $X_{IN}$ Rise to DBG Valid Delay | – | 30 |
| $T_2$ | $X_{IN}$ Rise to DBG Output Hold Time | 2 | – |
| $T_3$ | DBG to $X_{IN}$ Rise Input Setup Time | 10 | – |
| $T_4$ | DBG to $X_{IN}$ Rise Input Hold Time | 5 | – |
| | DBG frequency | | System Clock/4 |

## SPI Master Mode Timing

Figure 53 and Table 118 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.

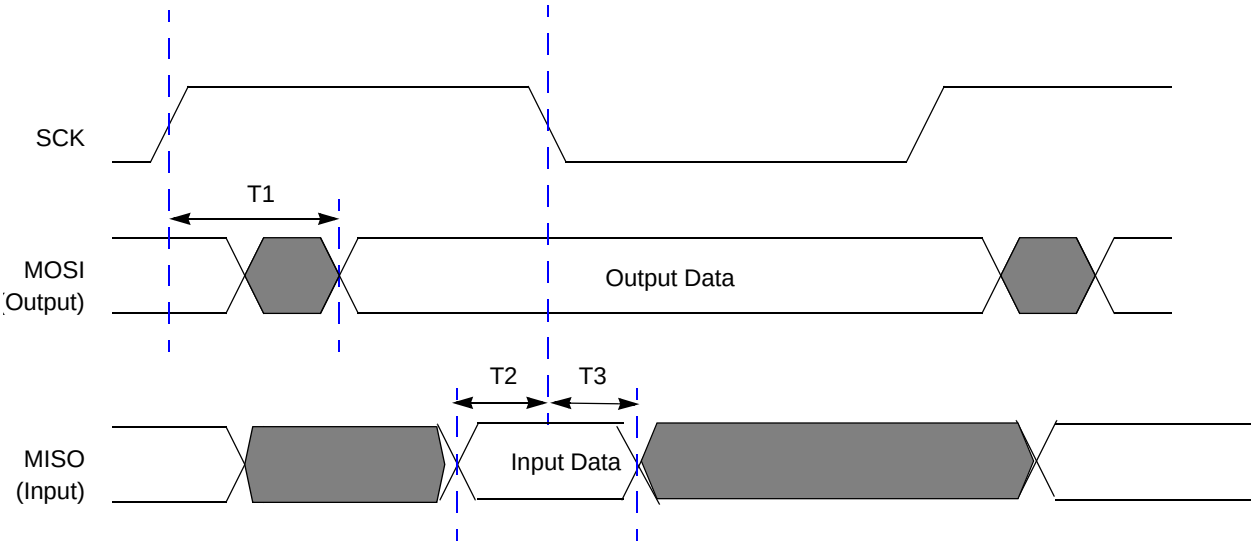



**Figure 53. SPI Master Mode Timing**

**Table 118. SPI Master Mode Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Min** | **Max** |
| **SPI Master** | | | |
| $T_1$ | SCK Rise to MOSI output Valid Delay | –5 | +5 |
| $T_2$ | MISO input to SCK (receive edge) Setup Time | 20 | |
| $T_3$ | MISO input to SCK (receive edge) Hold Time | 0 | |

## SPI Slave Mode Timing

Figure 54 and Table 119 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.

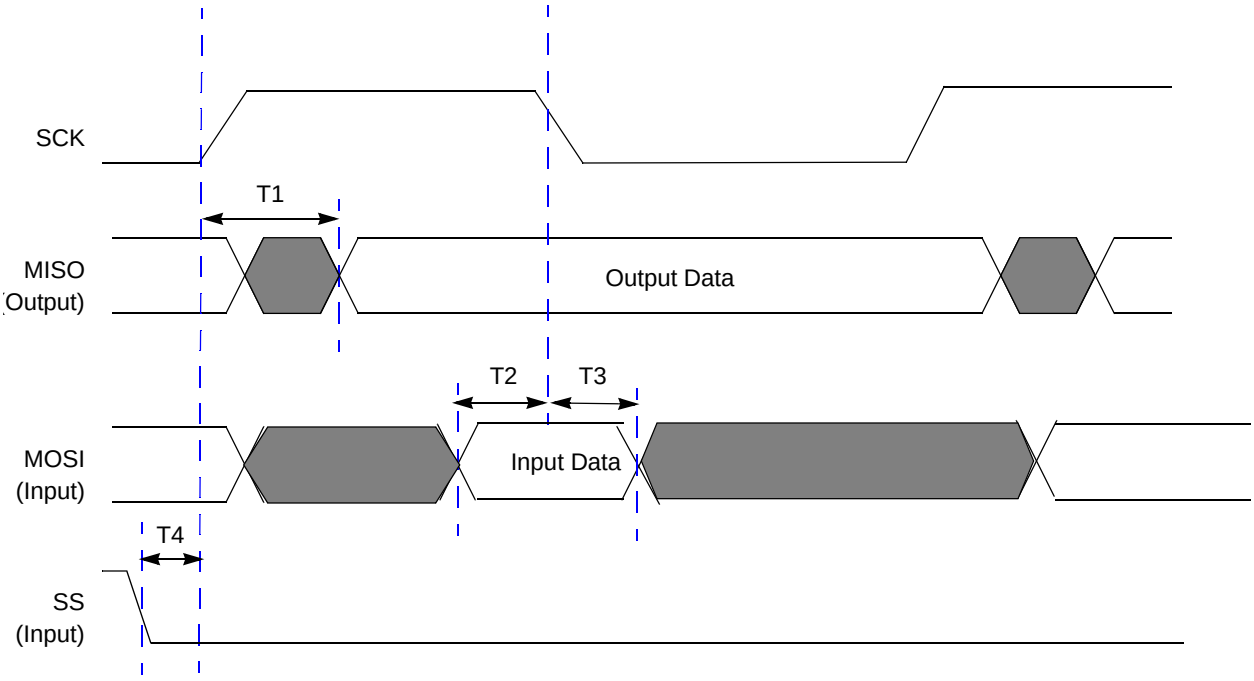



**Figure 54. SPI Slave Mode Timing**

**Table 119. SPI Slave Mode Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| **SPI Slave** | | | |
| $T_1$ | SCK (transmit edge) to MISO output Valid Delay | 2 * $X_{IN}$ period | 3 * $X_{IN}$ period + 20 nsec |
| $T_2$ | MOSI input to SCK (receive edge) Setup Time | 0 | |
| $T_3$ | MOSI input to SCK (receive edge) Hold Time | 3 * $X_{IN}$ period | |
| $T_4$ | SS input assertion to SCK setup | 1 * $X_{IN}$ period | |

# I²C Timing

Figure 55 and Table 120 provide timing information for I²C pins.

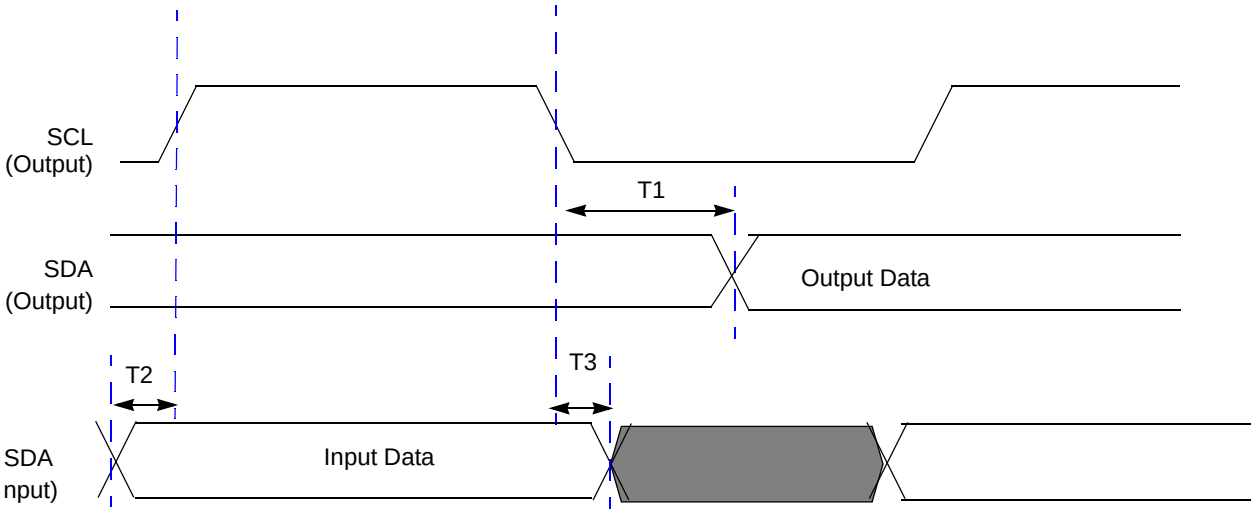



**Figure 55. I²C Timing**

**Table 120. I²C Timing**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| **I²C** | | | |
| $T_1$ | SCL Fall to SDA output delay | | SCL period/4 |
| $T_2$ | SDA Input to SCL rising edge Setup Time | 0 | |
| $T_3$ | SDA Input to SCL falling edge Hold Time | 0 | |

## UART Timing

Figure 56 and Table 121 provide timing information for UART pins for the case where the Clear To Send input pin ($\overline{CTS}$) is used for flow control. In this example, it is assumed that the Driver Enable polarity has been configured to be Active Low and is represented here by $\overline{DE}$. The $\overline{CTS}$ to $\overline{DE}$ assertion delay (T1) assumes the UART Transmit Data Register has been loaded with data prior to $\overline{CTS}$ assertion.

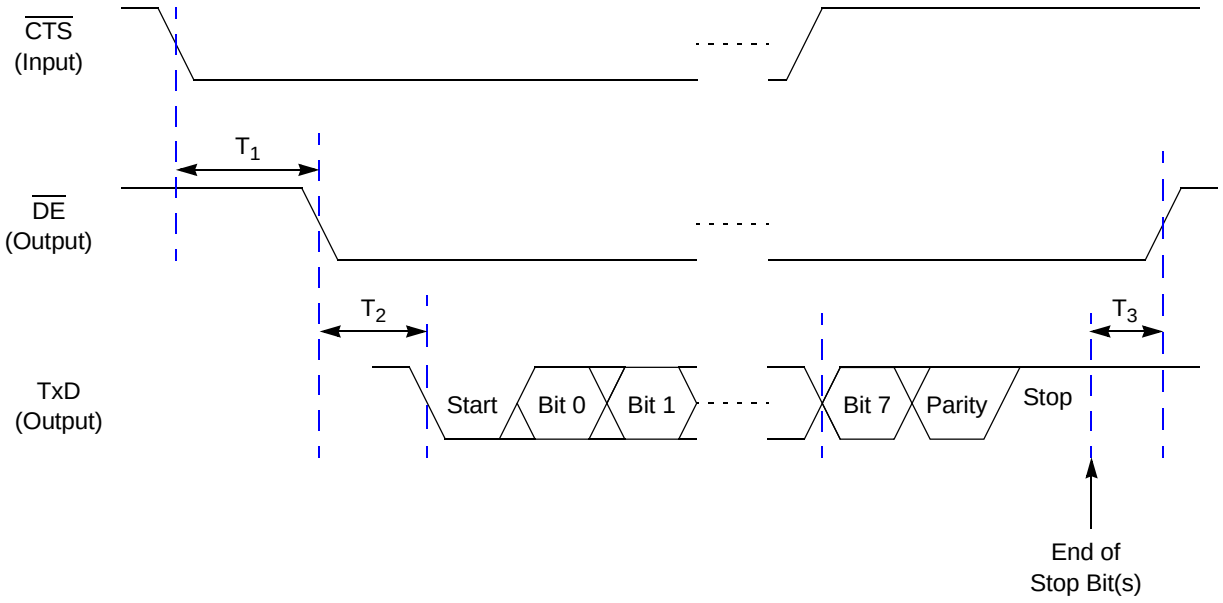



**Figure 56. UART Timing with $\overline{CTS}$**

**Table 121. UART Timing with $\overline{CTS}$**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_1$ | $\overline{CTS}$ Fall to $\overline{DE}$ Assertion Delay | 2 * $X_{IN}$ period | 2 * $X_{IN}$ period + 1 bit period |
| $T_2$ | $\overline{DE}$ Assertion to TxD Falling Edge (Start) Delay | 1 bit period | 1 bit period + 1 * $X_{IN}$ period |
| $T_3$ | End of stop bit(s) to $\overline{DE}$ Deassertion Delay | 1 * $X_{IN}$ period | 2 * $X_{IN}$ period |

Figure 57 and Table 122 provide timing information for UART pins for the case where the Clear To Send input signal ($\overline{CTS}$) is not used for flow control. In this example, it is assumed that the Driver Enable polarity has been configured to be Active Low and is represented here by $\overline{DE}$. $\overline{DE}$ asserts after the UART Transmit Data Register has been written. $\overline{DE}$ remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

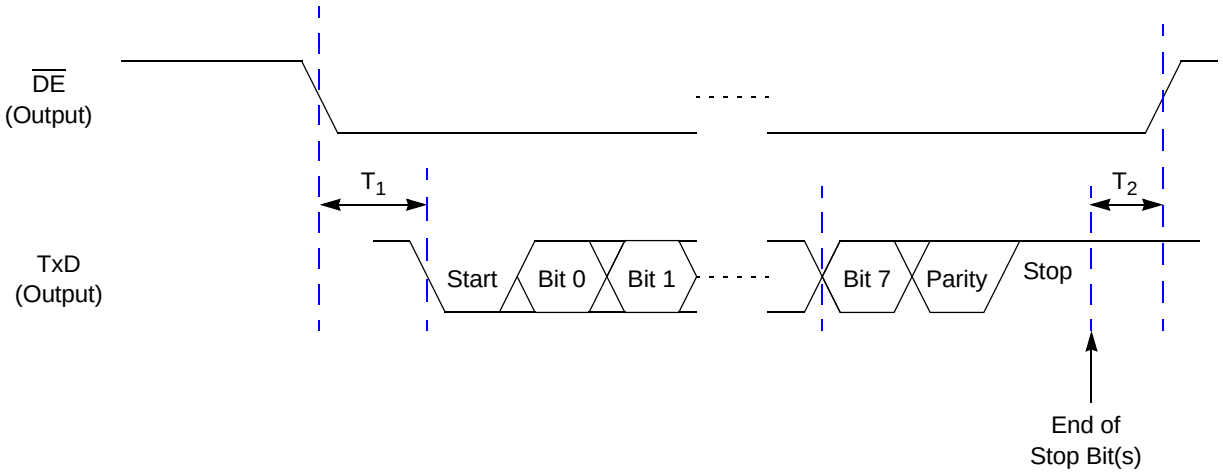



**Figure 57. UART Timing without $\overline{CTS}$**

**Table 122. UART Timing without $\overline{CTS}$**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_1$ | $\overline{DE}$ Assertion to TxD Falling Edge (Start) Delay | 1 bit period | 1 bit period + 1 * $X_{IN}$ period |
| $T_2$ | End of stop bit(s) to $\overline{DE}$ Deassertion Delay | 1 * $X_{IN}$ period | 2 * $X_{IN}$ period |

# eZ8 CPU Instruction Set

This chapter describes the following features of the eZ8 CPU instruction set:

Assembly Language Programming Introduction: see page 225

Assembly Language Syntax: see page 226

eZ8 CPU Instruction Notation: see page 227

eZ8 CPU Instruction Classes: see page 230

eZ8 CPU Instruction Summary: see page 234

## Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without having to be concerned with actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (op codes and operands) to represent the instructions themselves. The op codes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement can contain labels, operations, operands and comments.

Labels can be assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or pseudo-ops, are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the following example.

### Assembly Language Source Program Example

```
JP START        ; Everything after the semicolon is a comment.
START:          ; A label called “START”. The first instruction
                ; (JP START) in this example causes program
                ; execution to jump to the point within the
                ; program where the START label occurs.
LD R4, R7       ; A Load (LD) instruction with two operands. The
                ; first operand, Working Register R4, is the
                ; destination. The second operand, Working
                ; Register R7, is the source. The contents of R7
                ; is written into R4.
LD 234h, #%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 presents the operands in the *source, destination* order; however, ordering is op code-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. This binary format must be followed if you prefer manual program coding or intend to implement your own assembler.

**Example 1.** If the contents of Registers 43h and 08h are added and the result is stored in 43h, the assembly syntax and resulting object code result is shown in Table 123.

**Table 123. Assembly Language Syntax Example 1**

| | | | | |
|---|---|---|---|---|
| Assembly Language Code | ADD | 43h, | 08h | (ADD dst, src) |
| Object Code | 04 | 08 | 43 | (OPC src, dst) |

**Example 2.** In general, when an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or, using Escaped Mode Addressing, a Working Register R0–R15. If the contents of Register 43h and Working Register R8 are added and the result is stored in 43h, the assembly syntax and resulting object code result is shown in Table 124.

**Table 124. Assembly Language Syntax Example 2**

| | | | | |
|---|---|---|---|---|
| Assembly Language Code | ADD | 43h, | R8 | (ADD dst, src) |
| Object Code | 04 | E8 | 43 | (OPC src, dst) |

Refer to 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 125.

**Table 125. Notational Shorthand**

| Notation | Description | Operand | Range |
|---|---|---|---|
| b | Bit | b | b represents a value from 0 to 7 (000B to 111B). |
| cc | Condition Code | — | Refer to Condition Codes overview in the eZ8 CPU User Manual. |
| DA | Direct Address | Addrs | Addrs. represents a number in the range of 0000h to FFFFh. |
| ER | Extended Addressing Register | Reg | Reg. represents a number in the range of 000h to FFFh. |
| IM | Immediate Data | #Data | Data is a number between 00h to FFh. |
| Ir | Indirect Working Register | @Rn | n = 0 –15. |
| IR | Indirect Register | @Reg | Reg. represents a number in the range of 00h to FFh. |
| Irr | Indirect Working Register Pair | @RRp | p = 0, 2, 4, 6, 8, 10, 12, or 14. |
| IRR | Indirect Register Pair | @Reg | Reg. represents an even number in the range 00h to FEh. |
| p | Polarity | p | Polarity is a single bit binary value of either 0B or 1B. |
| r | Working Register | Rn | n = 0 – 15. |
| R | Register | Reg | Reg. represents a number in the range of 00h to FFh. |

**Table 125. Notational Shorthand (Continued)**

| Notation | Description | Operand | Range |
|---|---|---|---|
| RA | Relative Address | X | X represents an index in the range of +127 to –128 which is an offset relative to the address of the next instruction. |
| rr | Working Register Pair | RRp | p = 0, 2, 4, 6, 8, 10, 12, or 14. |
| RR | Register Pair | Reg | Reg. represents an even number in the range of 00h to FEh. |
| Vector | Vector Address | Vector | Vector represents a number in the range of 00h to FFh. |
| X | Indexed | #Index | The register or register pair to be indexed is offset by the signed Index value (#Index) in a +127 to –128 range. |

Table 126 contains additional symbols that are used throughout the Instruction Summary and Instruction Set Description sections.

**Table 126. Additional Symbols**

| Symbol | Definition |
|---|---|
| dst | Destination Operand |
| src | Source Operand |
| @ | Indirect Address Prefix |
| SP | Stack Pointer |
| PC | Program Counter |
| FLAGS | Flags Register |
| RP | Register Pointer |
| # | Immediate Operand Prefix |
| B | Binary Number Suffix |
| % | Hexadecimal Number Prefix |
| H | Hexadecimal Number Suffix |

Assignment of a value is indicated by an arrow, as shown in the following example.

dst ← dst + src

This example indicates that the source data is added to the destination data; the result is stored in the destination location.

## Condition Codes

The C, Z, S and V flags control the operation of the conditional jump (JP cc and JR cc) instructions. Sixteen frequently useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms bits 7:4 of the conditional jump instructions. The condition codes are summarized in Table 127. Some binary condition codes can be created using more than one assembly code mnemonic. The result of the flag test operation decides if the conditional jump is executed.

**Table 127. Condition Codes**

| Binary | Hex | Assembly Mnemonic | Definition | Flag Test Operation |
|---|---|---|---|---|
| 0000 | 0 | F | Always False | – |
| 0001 | 1 | LT | Less Than | (S XOR V) = 1 |
| 0010 | 2 | LE | Less Than or Equal | (Z OR (S XOR V)) = 1 |
| 0011 | 3 | ULE | Unsigned Less Than or Equal | (C OR Z) = 1 |
| 0100 | 4 | OV | Overflow | V = 1 |
| 0101 | 5 | MI | Minus | S = 1 |
| 0110 | 6 | Z | Zero | Z = 1 |
| 0110 | 6 | EQ | Equal | Z = 1 |
| 0111 | 7 | C | Carry | C = 1 |
| 0111 | 7 | ULT | Unsigned Less Than | C = 1 |
| 1000 | 8 | T (or blank) | Always True | – |
| 1001 | 9 | GE | Greater Than or Equal | (S XOR V) = 0 |
| 1010 | A | GT | Greater Than | (Z OR (S XOR V)) = 0 |
| 1011 | B | UGT | Unsigned Greater Than | (C = 0 AND Z = 0) = 1 |
| 1100 | C | NOV | No Overflow | V = 0 |
| 1101 | D | PL | Plus | S = 0 |
| 1110 | E | NZ | Non-Zero | Z = 0 |
| 1110 | E | NE | Not Equal | Z = 0 |
| 1111 | F | NC | No Carry | C = 0 |
| 1111 | F | UGE | Unsigned Greater Than or Equal | C = 0 |

# eZ8 CPU Instruction Classes

eZ8 CPU instructions can be divided functionally into the following groups:

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Tables 128 through 135 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table; these instructions can be considered to be 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 128. Arithmetic Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| ADC | dst, src | Add with Carry |
| ADCX | dst, src | Add with Carry using Extended Addressing |
| ADD | dst, src | Add |
| ADDX | dst, src | Add using Extended Addressing |
| CP | dst, src | Compare |
| CPC | dst, src | Compare with Carry |
| CPCX | dst, src | Compare with Carry using Extended Addressing |
| CPX | dst, src | Compare using Extended Addressing |
| DA | dst | Decimal Adjust |
| DEC | dst | Decrement |
| DECW | dst | Decrement Word |
| INC | dst | Increment |
| INCW | dst | Increment Word |

**Table 128. Arithmetic Instructions (Continued)**

| Mnemonic | Operands | Instruction |
|---|---|---|
| MULT | dst | Multiply |
| SBC | dst, src | Subtract with Carry |
| SBCX | dst, src | Subtract with Carry using Extended Addressing |
| SUB | dst, src | Subtract |
| SUBX | dst, src | Subtract using Extended Addressing |

**Table 129. 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 130. 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 131. CPU Control Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| ATM | — | Atomic Execution |
| CCF | — | Complement Carry Flag |
| DI | — | Disable Interrupts |
| EI | — | Enable Interrupts |
| HALT | — | Halt Mode |
| NOP | — | No Operation |
| RCF | — | Reset Carry Flag |
| SCF | — | Set Carry Flag |
| SRP | src | Set Register Pointer |
| STOP | — | Stop Mode |
| WDT | — | Watchdog Timer Refresh |

**Table 132. Load Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| CLR | dst | Clear |
| LD | dst, src | Load |
| LDC | dst, src | Load Constant to/from program memory |
| LDCI | dst, src | Load Constant to/from program memory and Auto-Increment addresses |
| LDE | dst, src | Load External Data to/from Data Memory |
| LDEI | dst, src | Load External Data to/from Data Memory and Auto-Increment addresses |
| LDWX | dst, src | Load Word using Extended Addressing |
| LDX | dst, src | Load using Extended Addressing |
| LEA | dst, X(src) | Load Effective Address |
| POP | dst | Pop |
| POPX | dst | Pop using Extended Addressing |
| PUSH | src | Push |
| PUSHX | src | Push using Extended Addressing |

**Table 133. 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 134. 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 135. 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 136 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 136. 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 |

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 136. 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 |
| ATM | Block all interrupt and DMA requests during execution of the next 3 instructions | | | 2F | – | – | – | – | – | – | 1 | 2 |
| BCLR bit, dst | dst[bit] ← 0 | r | | E2 | – | – | – | – | – | – | 2 | 2 |
| BIT p, bit, dst | dst[bit] ← p | r | | E2 | – | – | – | – | – | – | 2 | 2 |
| BRK | Debugger Break | | | 00 | – | – | – | – | – | – | 1 | 1 |
| BSET bit, dst | dst[bit] ← 1 | r | | E2 | – | – | – | – | – | – | 2 | 2 |
| BSWAP dst | dst[7:0] ← dst[0:7] | R | | D5 | X | * | * | 0 | – | – | 2 | 2 |
| BTJ p, bit, src, dst | if src[bit] = p PC ← PC + X | | r | F6 | – | – | – | – | – | – | 3 | 3 |
| | | | Ir | F7 | | | | | | | 3 | 4 |

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 136. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BTJNZ bit, src, dst | if src[bit] = 1<br>PC ← PC + X | | r | F6 | – | – | – | – | – | – | 3 | 3 |
| | | | Ir | F7 | | | | | | | 3 | 4 |
| 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 |

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 136. 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 |

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 136. 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 | 4 |
| | | R | IM | E6 | | | | | | | 3 | 2 |
| | | IR | IM | E7 | | | | | | | 3 | 3 |
| | | Ir | r | F3 | | | | | | | 2 | 3 |
| | | IR | R | F5 | | | | | | | 3 | 3 |
| LDC dst, src | dst ← src | r | Irr | C2 | – | – | – | – | – | – | 2 | 5 |
| | | Ir | Irr | C5 | | | | | | | 2 | 9 |
| | | Irr | r | D2 | | | | | | | 2 | 5 |
| LDCI dst, src | dst ← src<br>r ← r + 1<br>rr ← rr + 1 | Ir | Irr | C3 | – | – | – | – | – | – | 2 | 9 |
| | | Irr | Ir | D3 | | | | | | | 2 | 9 |
| LDE dst, src | dst ← src | r | Irr | 82 | – | – | – | – | – | – | 2 | 5 |
| | | Irr | r | 92 | | | | | | | 2 | 5 |
| LDEI dst, src | dst ← src<br>r ← r + 1<br>rr ← rr + 1 | Ir | Irr | 83 | – | – | – | – | – | – | 2 | 9 |
| | | Irr | Ir | 93 | | | | | | | 2 | 9 |
| LDWX dst, src | dst ← src | ER | ER | 1F E8 | – | – | – | – | – | – | 5 | 4 |

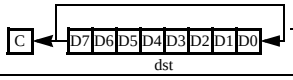
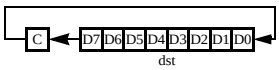
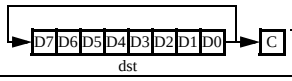
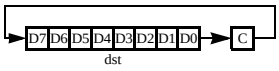
Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 136. 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 |
| POP dst | dst ← @SP SP ← SP + 1 | R | | 50 | – | – | – | – | – | – | 2 | 2 |
| | | IR | | 51 | | | | | | | 2 | 3 |

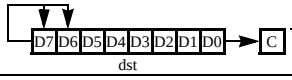
Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 136. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POPX dst | dst ← @SP<br>SP ← SP + 1 | ER | | D8 | – | – | – | – | – | – | 3 | 2 |
| PUSH src | SP ← SP – 1<br>@SP ← src | R | | 70 | – | – | – | – | – | – | 2 | 2 |
| | | IR | | 71 | | | | | | | 2 | 3 |
| | | IM | | 1F 70 | | | | | | | 3 | 2 |
| PUSHX src | SP ← SP – 1<br>@SP ← src | ER | | C8 | – | – | – | – | – | – | 3 | 2 |
| RCF | C ← 0 | | | CF | 0 | – | – | – | – | – | 1 | 2 |
| RET | PC ← @SP<br>SP ← SP + 2 | | | AF | – | – | – | – | – | – | 1 | 4 |
| RL dst |  | R | | 90 | * | * | * | * | – | – | 2 | 2 |
| | | IR | | 91 | | | | | | | 2 | 3 |
| RLC dst |  | R | | 10 | * | * | * | * | – | – | 2 | 2 |
| | | IR | | 11 | | | | | | | 2 | 3 |
| RR dst |  | R | | E0 | * | * | * | * | – | – | 2 | 2 |
| | | IR | | E1 | | | | | | | 2 | 3 |
| RRC dst |  | R | | C0 | * | * | * | * | – | – | 2 | 2 |
| | | IR | | C1 | | | | | | | 2 | 3 |
| SBC dst, src | dst ← dst – src – C | r | r | 32 | * | * | * | * | 1 | * | 2 | 3 |
| | | r | Ir | 33 | | | | | | | 2 | 4 |
| | | R | R | 34 | | | | | | | 3 | 3 |
| | | R | IR | 35 | | | | | | | 3 | 4 |
| | | R | IM | 36 | | | | | | | 3 | 3 |
| | | IR | IM | 37 | | | | | | | 3 | 4 |
| SBCX dst, src | dst ← dst – src – C | ER | ER | 38 | * | * | * | * | 1 | * | 4 | 3 |
| | | ER | IM | 39 | | | | | | | 4 | 3 |
| SCF | C ← 1 | | | DF | 1 | – | – | – | – | – | 1 | 2 |

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 136. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRA dst |  | R | | D0 | * | * | * | 0 | – | – | 2 | 2 |
| | | IR | | D1 | | | | | | | 2 | 3 |
| SRL dst |  | R | | 1F C0 | * | * | 0 | * | – | – | 3 | 2 |
| | | IR | | 1F C1 | | | | | | | 3 | 3 |
| SRP src | RP ← src | | IM | 01 | – | – | – | – | – | – | 2 | 2 |
| STOP | Stop Mode | | | 6F | – | – | – | – | – | – | 1 | 2 |
| SUB dst, src | dst ← dst – src | r | r | 22 | * | * | * | * | 1 | * | 2 | 3 |
| | | r | Ir | 23 | | | | | | | 2 | 4 |
| | | R | R | 24 | | | | | | | 3 | 3 |
| | | R | IR | 25 | | | | | | | 3 | 4 |
| | | R | IM | 26 | | | | | | | 3 | 3 |
| | | IR | IM | 27 | | | | | | | 3 | 4 |
| SUBX dst, src | dst ← dst – src | ER | ER | 28 | * | * | * | * | 1 | * | 4 | 3 |
| | | ER | IM | 29 | | | | | | | 4 | 3 |
| SWAP dst | dst[7:4] ↔ dst[3:0] | R | | F0 | X | * | * | X | – | – | 2 | 2 |
| | | IR | | F1 | | | | | | | 2 | 3 |
| TCM dst, src | (NOT dst) AND src | r | r | 62 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 63 | | | | | | | 2 | 4 |
| | | R | R | 64 | | | | | | | 3 | 3 |
| | | R | IR | 65 | | | | | | | 3 | 4 |
| | | R | IM | 66 | | | | | | | 3 | 3 |
| | | IR | IM | 67 | | | | | | | 3 | 4 |
| TCMX dst, src | (NOT dst) AND src | ER | ER | 68 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 69 | | | | | | | 4 | 3 |

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

**Table 136. eZ8 CPU Instruction Summary (Continued)**

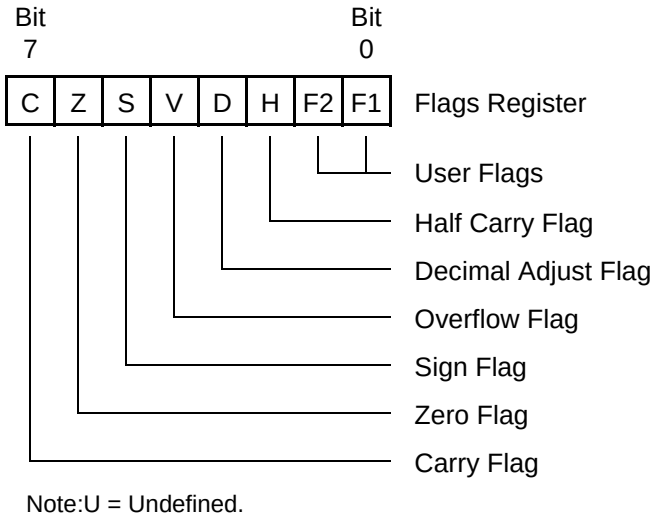
| Assembly Mnemonic | Symbolic Operation | Address Mode dst | Address Mode src | Opcode(s) (Hex) | Flags C | Z | S | V | D | H | Fetch Cycles | Instr. Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TM dst, src | dst AND src | r | r | 72 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | 73 | | | | | | | 2 | 4 |
| | | R | R | 74 | | | | | | | 3 | 3 |
| | | R | IR | 75 | | | | | | | 3 | 4 |
| | | R | IM | 76 | | | | | | | 3 | 3 |
| | | IR | IM | 77 | | | | | | | 3 | 4 |
| TMX dst, src | dst AND src | ER | ER | 78 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | 79 | | | | | | | 4 | 3 |
| TRAP Vector | SP ← SP – 2<br>@SP ← PC<br>SP ← SP – 1<br>@SP ← FLAGS<br>PC ← @Vector | | Vector | F2 | – | – | – | – | – | – | 2 | 6 |
| WDT | | | | 5F | – | – | – | – | – | – | 1 | 2 |
| XOR dst, src | dst ← dst XOR src | r | r | B2 | – | * | * | 0 | – | – | 2 | 3 |
| | | r | Ir | B3 | | | | | | | 2 | 4 |
| | | R | R | B4 | | | | | | | 3 | 3 |
| | | R | IR | B5 | | | | | | | 3 | 4 |
| | | R | IM | B6 | | | | | | | 3 | 3 |
| | | IR | IM | B7 | | | | | | | 3 | 4 |
| XORX dst, src | dst ← dst XOR src | ER | ER | B8 | – | * | * | 0 | – | – | 4 | 3 |
| | | ER | IM | B9 | | | | | | | 4 | 3 |

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

# Flags Register

The Flags Register contains the status information regarding the most recent arithmetic, logical, bit manipulation or rotate and shift operation. The Flags Register contains six bits of status information that are set or cleared by CPU operations. Four of the bits (C, V, Z and S) can be tested 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 58 displays the flags and their bit positions in the Flags Register.




Note:U = Undefined.

**Figure 58. Flags Register**

Interrupts, the software trap (TRAP) instruction, and illegal instruction traps all write the value of the Flags Register to the stack. Executing an interrupt return (IRET) instruction restores the value saved on the stack into the Flags Register.

# Op Code Maps

A description of the op code map data and the abbreviations are provided in Figure 59 and Table 137. Figures 60 and 61 provide information about each of the eZ8 CPU instructions.

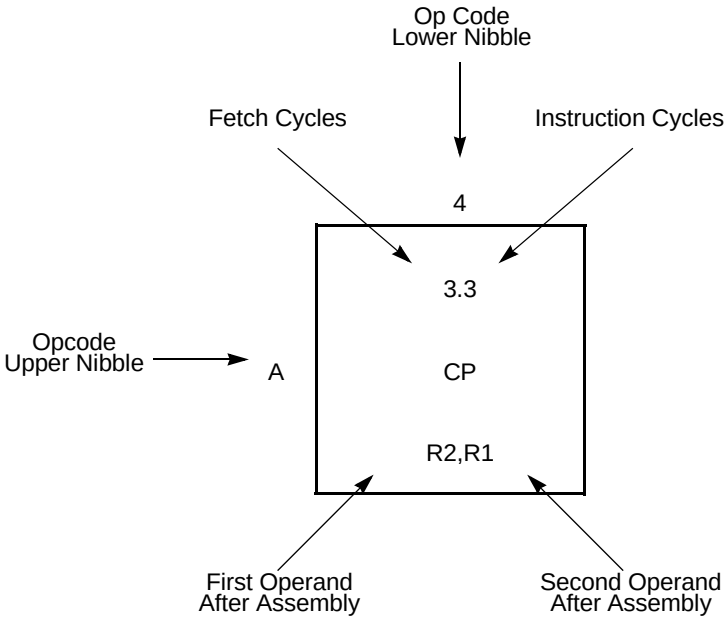



**Figure 59. Op Code Map Cell Description**

**Table 137. Op Code Map Abbreviations**

| Abbreviation | Description | Abbreviation | Description |
|---|---|---|---|
| b | Bit position | IRR | Indirect register pair |
| cc | Condition code | p | Polarity (0 or 1) |
| X | 8-bit signed index or displacement | r | 4-bit working register |
| DA | Destination address | R | 8-bit register |
| ER | Extended addressing register | r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1 | Destination address |

**Table 137. Op Code Map Abbreviations (Continued)**

| Abbreviation | Description | Abbreviation | Description |
|---|---|---|---|
| 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 | | | | | | 1,2 **ATM** |
| 3 | 2.2 **DEC** R1 | 2.3 **DEC** IR1 | 2.3 **SBC** r1,r2 | 2.4 **SBC** r1,Ir2 | 3.3 **SBC** R2,R1 | 3.4 **SBC** IR2,R1 | 3.3 **SBC** R1,IM | 3.4 **SBC** IR1,IM | 4.3 **SBCX** ER2,ER1 | 4.3 **SBCX** IM,ER1 | | | | | | |
| 4 | 2.2 **DA** R1 | 2.3 **DA** IR1 | 2.3 **OR** r1,r2 | 2.4 **OR** r1,Ir2 | 3.3 **OR** R2,R1 | 3.4 **OR** IR2,R1 | 3.3 **OR** R1,IM | 3.4 **OR** IR1,IM | 4.3 **ORX** ER2,ER1 | 4.3 **ORX** IM,ER1 | | | | | | |
| 5 | 2.2 **POP** R1 | 2.3 **POP** IR1 | 2.3 **AND** r1,r2 | 2.4 **AND** r1,Ir2 | 3.3 **AND** R2,R1 | 3.4 **AND** IR2,R1 | 3.3 **AND** R1,IM | 3.4 **AND** IR1,IM | 4.3 **ANDX** ER2,ER1 | 4.3 **ANDX** IM,ER1 | | | | | | 1.2 **WDT** |
| 6 | 2.2 **COM** R1 | 2.3 **COM** IR1 | 2.3 **TCM** r1,r2 | 2.4 **TCM** r1,Ir2 | 3.3 **TCM** R2,R1 | 3.4 **TCM** IR2,R1 | 3.3 **TCM** R1,IM | 3.4 **TCM** IR1,IM | 4.3 **TCMX** ER2,ER1 | 4.3 **TCMX** IM,ER1 | | | | | | 1.2 **STOP** |
| 7 | 2.2 **PUSH** R2 | 2.3 **PUSH** IR2 | 2.3 **TM** r1,r2 | 2.4 **TM** r1,Ir2 | 3.3 **TM** R2,R1 | 3.4 **TM** IR2,R1 | 3.3 **TM** R1,IM | 3.4 **TM** IR1,IM | 4.3 **TMX** ER2,ER1 | 4.3 **TMX** IM,ER1 | | | | | | 1.2 **HALT** |
| 8 | 2.5 **DECW** RR1 | 2.6 **DECW** IRR1 | 2.5 **LDE** r1,Irr2 | 2.9 **LDEI** Ir1,Irr2 | 3.2 **LDX** r1,ER2 | 3.3 **LDX** Ir1,ER2 | 3.4 **LDX** IRR2,R1 | 3.5 **LDX** IRR2,IR1 | 3.4 **LDX** r1,rr2,X | 3.4 **LDX** rr1,r2,X | | | | | | 1.2 **DI** |
| 9 | 2.2 **RL** R1 | 2.3 **RL** IR1 | 2.5 **LDE** r2,Irr1 | 2.9 **LDEI** Ir2,Irr1 | 3.2 **LDX** r2,ER1 | 3.3 **LDX** Ir2,ER1 | 3.4 **LDX** R2,IRR1 | 3.5 **LDX** IR2,IRR1 | 3.3 **LEA** r1,r2,X | 3.5 **LEA** rr1,rr2,X | | | | | | 1.2 **EI** |
| A | 2.5 **INCW** RR1 | 2.6 **INCW** IRR1 | 2.3 **CP** r1,r2 | 2.4 **CP** r1,Ir2 | 3.3 **CP** R2,R1 | 3.4 **CP** IR2,R1 | 3.3 **CP** R1,IM | 3.4 **CP** IR1,IM | 4.3 **CPX** ER2,ER1 | 4.3 **CPX** IM,ER1 | | | | | | 1.4 **RET** |
| B | 2.2 **CLR** R1 | 2.3 **CLR** IR1 | 2.3 **XOR** r1,r2 | 2.4 **XOR** r1,Ir2 | 3.3 **XOR** R2,R1 | 3.4 **XOR** IR2,R1 | 3.3 **XOR** R1,IM | 3.4 **XOR** IR1,IM | 4.3 **XORX** ER2,ER1 | 4.3 **XORX** IM,ER1 | | | | | | 1.5 **IRET** |
| C | 2.2 **RRC** R1 | 2.3 **RRC** IR1 | 2.5 **LDC** r1,Irr2 | 2.9 **LDCI** Ir1,Irr2 | 2.3 **JP** IRR1 | 2.9 **LDC** Ir1,Irr2 | | 3.4 **LD** r1,r2,X | 3.2 **PUSHX** ER2 | | | | | | | 1.2 **RCF** |
| D | 2.2 **SRA** R1 | 2.3 **SRA** IR1 | 2.5 **LDC** r2,Irr1 | 2.9 **LDCI** Ir2,Irr1 | 2.6 **CALL** IRR1 | 2.2 **BSWAP** R1 | 3.3 **CALL** DA | 3.4 **LD** r2,r1,X | 3.2 **POPX** ER1 | | | | | | | 1.2 **SCF** |
| E | 2.2 **RR** R1 | 2.3 **RR** IR1 | 2.2 **BIT** p,b,r1 | 2.3 **LD** r1,Ir2 | 3.2 **LD** R2,R1 | 3.3 **LD** IR2,R1 | 3.2 **LD** R1,IM | 3.3 **LD** IR1,IM | 4.2 **LDX** ER2,ER1 | 4.2 **LDX** IM,ER1 | | | | | | 1.2 **CCF** |
| F | 2.2 **SWAP** R1 | 2.3 **SWAP** IR1 | 2.6 **TRAP** Vector | 2.3 **LD** Ir1,r2 | 2.8 **MULT** RR1 | 3.3 **LD** R2,IR1 | 3.3 **BTJ** p,b,r1,X | 3.4 **BTJ** p,b,Ir1,X | | | | | | | | |

**Figure 60. First Op Code Map**

| Upper Nibble (Hex) \ Lower Nibble (Hex) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | |
| 7 | 3,2 **PUSH** IM | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | |
| A | | | 3.3 **CPC** r1,r2 | 3.4 **CPC** r1,Ir2 | 4.3 **CPC** R2,R1 | 4.4 **CPC** IR2,R1 | 4.3 **CPC** R1,IM | 4.4 **CPC** IR1,IM | 5.3 **CPCX** ER2,ER1 | 5.3 **CPCX** IM,ER1 | | | | | | |
| B | | | | | | | | | | | | | | | | |
| C | 3.2 **SRL** R1 | 3.3 **SRL** IR1 | | | | | | | | | | | | | | |
| D | | | | | | | | | | | | | | | | |
| E | | | | | | | | | 5,4 **LDWX** ER2,ER1 | | | | | | | |
| F | | | | | | | | | | | | | | | | |

**Figure 61. Second Op Code Map after 1Fh**

# Appendix B. Register Tables

For the reader's convenience, this appendix lists all F64xx Series registers numerically by hexadecimal address.

## General Purpose RAM

In the F64xx Series, the 000–FFF hexadecimal address range is partitioned for general-purpose random access memory, as follows.

### Hex Addresses: 000–7FF

This address range is reserved for 2KB general-purpose register file RAM devices. For more details, see the Register File section on page 18.

### Hex Addresses: 000–FFF

This address range is reserved for 4KB general-purpose register file RAM devices. For more details, see the Register File section on page 18.

## Timer 0

For more information about these Timer Control registers, see the Timer Control Register Definitions section on page 72.

### Hex Address: F00

**Table 138. Timer 0–3 High Byte Register (TxH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F00h, F08h, F10h, F18h | | | | | | | |

### Hex Address: F01

**Table 139. Timer 0–3 Low Byte Register (TxL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TL | | | | | | | |
| RESET | 0 | | | | | | | 1 |
| R/W | R/W | | | | | | | |
| Address | F01h, F09h, F11h, F19h | | | | | | | |

### Hex Address: F02

**Table 140. Timer 0–3 Reload High Byte Register (TxRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F02h, F0Ah, F12h, F1Ah | | | | | | | |

### Hex Address: F03

**Table 141. Timer 0–3 Reload Low Byte Register (TxRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F03h, F0Bh, F13h, F1Bh | | | | | | | |

### Hex Address: F04

**Table 142. Timer 0–3 PWM High Byte Register (TxPWMH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWMH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F04h, F0Ch, F14h, F1Ch | | | | | | | |

### Hex Address: F05

**Table 143. Timer 0–3 PWM Low Byte Register (TxPWML)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWML | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F05h, F0Dh, F15h, F1Dh | | | | | | | |

### Hex Address: F06

**Table 144. Timer 0–3 Control 0 Register (TxCTL0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | CSC | Reserved | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F06h, F0Eh, F16h, F1Eh | | | | | | | |

### Hex Address: F07

**Table 145. Timer 0–3 Control 1 Register (TxCTL1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | TPOL | PRES | | | TMODE | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F07h, F0Fh, F17h, F1Fh | | | | | | | |

### Hex Address: F08

**Table 146. Timer 0–3 High Byte Register (TxH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F00h, F08h, F10h, F18h | | | | | | | |

### Hex Address: F09

**Table 147. Timer 0–3 Low Byte Register (TxL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TL | | | | | | | |
| RESET | 0 | | | | | | | 1 |
| R/W | R/W | | | | | | | |
| Address | F01h, F09h, F11h, F19h | | | | | | | |

### Hex Address: F0A

**Table 148. Timer 0–3 Reload High Byte Register (TxRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F02h, F0Ah, F12h, F1Ah | | | | | | | |

### Hex Address: F0B

**Table 149. Timer 0–3 Reload Low Byte Register (TxRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F03h, F0Bh, F13h, F1Bh | | | | | | | |

### Hex Address: F0C

**Table 150. Timer 0–3 PWM High Byte Register (TxPWMH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWMH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F04h, F0Ch, F14h, F1Ch | | | | | | | |

### Hex Address: F0D

**Table 151. Timer 0–3 PWM Low Byte Register (TxPWML)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWML | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F05h, F0Dh, F15h, F1Dh | | | | | | | |

### Hex Address: F0E

**Table 152. Timer 0–3 Control 0 Register (TxCTL0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | CSC | Reserved | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F06h, F0Eh, F16h, F1Eh | | | | | | | |

### Hex Address: F0F

**Table 153. Timer 0–3 Control 1 Register (TxCTL1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | TPOL | PRES | | | TMODE | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F07h, F0Fh, F17h, F1Fh | | | | | | | |

### Hex Address: F10

**Table 154. Timer 0–3 High Byte Register (TxH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F00h, F08h, F10h, F18h | | | | | | | |

### Hex Address: F11

**Table 155. Timer 0–3 Low Byte Register (TxL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TL | | | | | | | |
| RESET | 0 | | | | | | | 1 |
| R/W | R/W | | | | | | | |
| Address | F01h, F09h, F11h, F19h | | | | | | | |

### Hex Address: F12

**Table 156. Timer 0–3 Reload High Byte Register (TxRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F02h, F0Ah, F12h, F1Ah | | | | | | | |

### Hex Address: F13

**Table 157. Timer 0–3 Reload Low Byte Register (TxRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F03h, F0Bh, F13h, F1Bh | | | | | | | |

### Hex Address: F14

**Table 158. Timer 0–3 PWM High Byte Register (TxPWMH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWMH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F04h, F0Ch, F14h, F1Ch | | | | | | | |

### Hex Address: F15

**Table 159. Timer 0–3 PWM Low Byte Register (TxPWML)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWML | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F05h, F0Dh, F15h, F1Dh | | | | | | | |

### Hex Address: F16

**Table 160. Timer 0–3 Control 0 Register (TxCTL0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | CSC | Reserved | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F06h, F0Eh, F16h, F1Eh | | | | | | | |

### Hex Address: F17

**Table 161. Timer 0–3 Control 1 Register (TxCTL1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | TPOL | PRES | | | TMODE | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F07h, F0Fh, F17h, F1Fh | | | | | | | |

### Hex Address: F18

**Table 162. Timer 0–3 High Byte Register (TxH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F00h, F08h, F10h, F18h | | | | | | | |

### Hex Address: F19

**Table 163. Timer 0–3 Low Byte Register (TxL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TL | | | | | | | |
| RESET | 0 | | | | | | | 1 |
| R/W | R/W | | | | | | | |
| Address | F01h, F09h, F11h, F19h | | | | | | | |

### Hex Address: F1A

**Table 164. Timer 0–3 Reload High Byte Register (TxRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F02h, F0Ah, F12h, F1Ah | | | | | | | |

### Hex Address: F1B

**Table 165. Timer 0–3 Reload Low Byte Register (TxRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F03h, F0Bh, F13h, F1Bh | | | | | | | |

### Hex Address: F1C

**Table 166. Timer 0–3 PWM High Byte Register (TxPWMH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWMH | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F04h, F0Ch, F14h, F1Ch | | | | | | | |

### Hex Address: F1D

**Table 167. Timer 0–3 PWM Low Byte Register (TxPWML)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PWML | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F05h, F0Dh, F15h, F1Dh | | | | | | | |

### Hex Address: F1E

**Table 168. Timer 0–3 Control 0 Register (TxCTL0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | CSC | Reserved | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F06h, F0Eh, F16h, F1Eh | | | | | | | |

### Hex Address: F1F

**Table 169. Timer 0–3 Control 1 Register (TxCTL1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | TPOL | PRES | | | TMODE | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F07h, F0Fh, F17h, F1Fh | | | | | | | |

### Hex Addresses: F20–F39

This address range is reserved.

## Universal Asynchronous Receiver/Transmitter (UART)

For more information about these UART Control registers, see the UART Control Register Definitions section on page 98.

### Hex Address: F40

**Table 170. UART Transmit Data Register (U*x*TXD)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TXD | | | | | | | |
| RESET | X | | | | | | | |
| R/W | W | | | | | | | |
| Address | F40h and F48h | | | | | | | |

**Table 171. UART Receive Data Register (U*x*RXD)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | RXD | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | F40h and F48h | | | | | | | |

### Hex Address: F41

**Table 172. UART Status 0 Register (U*x*STAT0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | RDA | PE | OE | FE | BRKD | TDRE | TXE | CTS |
| RESET | 0 | | | | | | 1 | X |
| R/W | R | | | | | | | |
| Address | F41h and F49h | | | | | | | |

### Hex Address: F42

**Table 173. UART Control 0 Register (U*x*CTL0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | REN | CTSE | PEN | PSEL | SBRK | STOP | LBEN |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F42h and F4Ah | | | | | | | |

### Hex Address: F43

**Table 174. UART Control 1 Register (U*x*CTL1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | MPMD[1] | MPEN | MPMD[0] | MPBT | DEPOL | BRGCTL | RDAIRQ | IREN |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F43h and F4Bh | | | | | | | |

### Hex Address: F44

**Table 175. UART Status 1 Register (U*x*STAT1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | | | NEWFRM | MPRX |
| RESET | 0 | | | | | | | |
| R/W | R | | | | R/W | | R | |
| Address | F44h and F4Ch | | | | | | | |

### Hex Address: F45

**Table 176. UART Address Compare Register (U*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | COMP_ADDR | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F45h and F4Dh | | | | | | | |

### Hex Address: F46

**Table 177. UART Baud Rate High Byte Register (U*x*BRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F46h and F4Eh | | | | | | | |

## Hex Address: F47

**Table 178. UART Baud Rate Low Byte Register (U*x*BRL)**

| Bit7 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F47h and F4Fh | | | | | | | |

## Hex Address: F48

**Table 179. UART Transmit Data Register (U*x*TXD)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TXD | | | | | | | |
| RESET | X | | | | | | | |
| R/W | W | | | | | | | |
| Address | F40h and F48h | | | | | | | |

**Table 180. UART Receive Data Register (U*x*RXD)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | RXD | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | F40h and F48h | | | | | | | |

## Hex Address: F49

**Table 181. UART Status 0 Register (U*x*STAT0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | RDA | PE | OE | FE | BRKD | TDRE | TXE | CTS |
| RESET | 0 | | | | | | 1 | X |
| R/W | R | | | | | | | |
| Address | F41h and F49h | | | | | | | |

### Hex Address: F4A

**Table 182. UART Control 0 Register (U*x*CTL0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | REN | CTSE | PEN | PSEL | SBRK | STOP | LBEN |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F42h and F4Ah | | | | | | | |

### Hex Address: F4B

**Table 183. UART Control 1 Register (U*x*CTL1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | MPMD[1] | MPEN | MPMD[0] | MPBT | DEPOL | BRGCTL | RDAIRQ | IREN |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F43h and F4Bh | | | | | | | |

### Hex Address: F4C

**Table 184. UART Status 1 Register (U*x*STAT1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | | | NEWFRM | MPRX |
| RESET | 0 | | | | | | | |
| R/W | R | | | | R/W | | R | |
| Address | F44h and F4Ch | | | | | | | |

### Hex Address: F4D

**Table 185. UART Address Compare Register (U*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | COMP_ADDR | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F45h and F4Dh | | | | | | | |

### Hex Address: F4E

**Table 186. UART Baud Rate High Byte Register (UxBRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F46h and F4Eh | | | | | | | |

### Hex Address: F4F

**Table 187. UART Baud Rate Low Byte Register (UxBRL)**

| Bit7 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F47h and F4Fh | | | | | | | |

# Inter-Integrated Circuit ($I^2C$)

For more information about these $I^2C$ Control registers, see the I2C Control Register Definitions section on page 141.

### Hex Address: F50

**Table 188. $I^2C$ Data Register (I2CDATA)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DATA | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F50h | | | | | | | |

### Hex Address: F51

**Table 189. I²C Status Register (I2CSTAT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TDRE | RDRF | ACK | 10B | RD | TAS | DSS | NCKI |
| RESET | 1 | 0 | | | | | | |
| R/W | R | | | | | | | |
| Address | F51h | | | | | | | |

### Hex Address: F52

**Table 190. I²C Control Register (I2CCTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | IEN | START | STOP | BIRQ | TXI | NAK | FLUSH | FILTEN |
| RESET | 0 | | | | | | | |
| R/W | R/W | R/W1 | R/W1 | R/W | R/W | R/W1 | W1 | R/W |
| Address | F52h | | | | | | | |

### Hex Address: F53

**Table 191. I²C Baud Rate High Byte Register (I2CBRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRH | | | | | | | |
| RESET | FFh | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F53h | | | | | | | |

### Hex Address: F54

**Table 192. I²C Baud Rate Low Byte Register (I2CBRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRL | | | | | | | |
| RESET | FFh | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F54h | | | | | | | |

### Hex Address: F55

**Table 193. I²C Diagnostic State Register (I2CDST)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SCLIN | SDAIN | STPCNT | TXRXSTATE | | | | |
| RESET | X | | 0 | | | | | |
| R/W | R | | | | | | | |
| Address | F55h | | | | | | | |

### Hex Address: F56

**Table 194. I²C Diagnostic Control Register (I2CDIAG)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | | | | DIAG |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | R/W |
| Address | F56h | | | | | | | |

### Hex Addresses: F57–F5F

This address range is reserved.

## Serial Peripheral Interface

For more information about these SPI Control registers, see the SPI Control Register Definitions section on page 121.

### Hex Address: F60

**Table 195. SPI Data Register (SPIDATA)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DATA | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F60h | | | | | | | |

## Hex Address: F61

**Table 196. SPI Control Register (SPICTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | IRQE | STR | BIRQ | PHASE | CLKPOL | WOR | MMEN | SPIEN |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F61h | | | | | | | |

## Hex Address: F62

**Table 197. SPI Status Register (SPISTAT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | IRQ | OVR | COL | ABT | Reserved | | TXST | SLAS |
| RESET | 0 | | | | | | | 1 |
| R/W | R/W* | | | | R | | | |
| Address | F62h | | | | | | | |
| Note: R/W* = Read access. Write a 1 to clear the bit to 0. | | | | | | | | |

## Hex Address: F63

**Table 198. SPI Mode Register (SPIMODE)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | DIAG | NUMBITS[2:0] | | | SSIO | SSV |
| RESET | 0 | | | | | | | |
| R/W | R | | R/W | | | | | |
| Address | F63h | | | | | | | |

### Hex Address: F64

**Table 199. SPI Diagnostic State Register (SPIDST)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SCKEN | TCKEN | SPISTATE | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | |
| Address | F64h | | | | | | | |

### Hex Address: F65

This address is reserved.

### Hex Address: F66

**Table 200. SPI Baud Rate High Byte Register (SPIBRH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F66h | | | | | | | |

### Hex Address: F67

**Table 201. SPI Baud Rate Low Byte Register (SPIBRL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | BRL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | F67h | | | | | | | |

### Hex Addresses: F68–F6F

This address range is reserved.

# Analog-to-Digital Converter (ADC)

For more information about these ADC Control registers, see the ADC Control Register Definitions section on page 165.

### Hex Addresses: F70–F71

This address range is reserved.

### Hex Address: F72

**Table 202. ADC Data High Byte Register (ADCD_H)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | ADCD_H | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | F72h | | | | | | | |

### Hex Address: F73

**Table 203. ADC Data Low Bits Register (ADCD_L)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | ADCD_L | | Reserved | | | | | |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | F73h | | | | | | | |

### Hex Addresses: F74–FAF

This address range is reserved.

# Direct Memory Access (DMA)

For more information about these DMA Control registers, see the DMA Control Register Definitions section on page 152.

### Hex Address: FB0

**Table 204. DMA*x* Control Register (DMA*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DEN | DLE | DDIR | IRQEN | WSEL | RSS | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB0h, FB8h | | | | | | | |

### Hex Address: FB1

**Table 205. DMA*x* I/O Address Register (DMA*x*IO)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_IO | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB1h, FB9h | | | | | | | |

### Hex Address: FB2

**Table 206. DMA*x* Address High Nibble Register (DMA*x*H)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_END_H | | | | DMA_START_H | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB2h, FBAh | | | | | | | |

### Hex Address: FB3

**Table 207. DMA*x* Start/Current Address Low Byte Register (DMA*x*START)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_START | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB3h, FBBh | | | | | | | |

### Hex Address: FB4

**Table 208. DMA*x* End Address Low Byte Register (DMA*x*END)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_END | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB4h, FBCh | | | | | | | |

### Hex Addresses: FB5–FB7

This address range is reserved.

### Hex Address: FB8

**Table 209. DMA*x* Control Register (DMA*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DEN | DLE | DDIR | IRQEN | WSEL | RSS | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB0h, FB8h | | | | | | | |

### Hex Address: FB9

**Table 210. DMA*x* I/O Address Register (DMA*x*IO)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_IO | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB1h, FB9h | | | | | | | |

### Hex Address: FBA

**Table 211. DMA*x* Address High Nibble Register (DMA*x*H)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_END_H | | | | DMA_START_H | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB2h, FBAh | | | | | | | |

### Hex Address: FBB

**Table 212. DMA*x* Start/Current Address Low Byte Register (DMA*x*START)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_START | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB3h, FBBh | | | | | | | |

### Hex Address: FBC

**Table 213. DMA*x* End Address Low Byte Register (DMA*x*END)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMA_END | | | | | | | |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FB4h, FBCh | | | | | | | |

### Hex Address: FBD

**Table 214. DMA_ADC Address Register (DMAA_ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DMAA_ADDR | | | | | | | Reserved |
| RESET | X | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FBDh | | | | | | | |

### Hex Address: FBE

**Table 215. DMA_ADC Control Register (DMAACTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DAEN | IRQEN | Reserved | | ADC_IN | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FBEh | | | | | | | |

### Hex Address: FBF

**Table 216. DMA_ADC Status Register (DMAA_STAT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | CADC[3:0] | | | | Reserved | IRQA | IRQ1 | IRQ0 |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | |
| Address | FBFh | | | | | | | |

## Interrupt Request (IRQ)

For more information about these IRQ Control registers, see the Interrupt Control Register Definitions section on page 51.

### Hex Address: FC0

**Table 217. Interrupt Request 0 Register (IRQ0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T2I | T1I | T0I | U0RXI | U0TXI | I2CI | SPII | ADCI |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC0h | | | | | | | |

### Hex Address: FC1

**Table 218. IRQ0 Enable High Bit Register (IRQ0ENH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T2ENH | T1ENH | T0ENH | U0RENH | U0TENH | I2CENH | SPIENH | ADCENH |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC1h | | | | | | | |

### Hex Address: FC2

**Table 219. IRQ0 Enable Low Bit Register (IRQ0ENL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T2ENL | T1ENL | T0ENL | U0RENL | U0TENL | I2CENL | SPIENL | ADCENL |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC2h | | | | | | | |

### Hex Address: FC3

**Table 220. Interrupt Request 1 Register (IRQ1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PAD7I | PAD6I | PAD5I | PAD4I | PAD3I | PAD2I | PAD1I | PAD0I |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC3h | | | | | | | |

### Hex Address: FC4

**Table 221. IRQ1 Enable High Bit Register (IRQ1ENH)**

| Bit | 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 |
| Address | FC4h | | | | | | | |

### Hex Address: FC5

**Table 222. IRQ1 Enable Low Bit Register (IRQ1ENL)**

| Bit | 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 |
| Address | FC5h | | | | | | | |

### Hex Address: FC6

**Table 223. Interrupt Request 2 Register (IRQ2)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T3I | U1RXI | U1TXI | DMAI | PC3I | PC2I | PC1I | PC0I |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC6h | | | | | | | |

### Hex Address: FC7

**Table 224. IRQ2 Enable High Bit Register (IRQ2ENH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T3ENH | U1RENH | U1TENH | DMAENH | C3ENH | C2ENH | C1ENH | C0ENH |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC7h | | | | | | | |

### Hex Address: FC8

**Table 225. IRQ2 Enable Low Bit Register (IRQ2ENL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | T3ENL | U1RENL | U1TENL | DMAENL | C3ENL | C2ENL | C1ENL | C0ENL |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FC8h | | | | | | | |

### Hex Addresses: FC9–FCC

This address range is reserved.

### Hex Address: FCD

**Table 226. Interrupt Edge Select Register (IRQES)**

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

### Hex Address: FCE

**Table 227. Interrupt Port Select Register (IRQPS)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PAD7S | PAD6S | PAD5S | PAD4S | PAD3S | PAD2S | PAD1S | PAD0S |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FCEh | | | | | | | |

### Hex Address: FCF

**Table 228. Interrupt Control Register (IRQCTL)**

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

# General-Purpose Input/Output (GPIO)

For more information about these GPIO Control registers, see the GPIO Control Register Definitions section on page 39.

## Hex Address: FD0

**Table 229. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

## Hex Address: FD1

**Table 230. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

## Hex Address: FD2

**Table 231. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

### Hex Address: FD3

**Table 232. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

### Hex Address: FD4

**Table 233. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

### Hex Address: FD5

**Table 234. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

### Hex Address: FD6

**Table 235. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

### Hex Address: FD7

**Table 236. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

### Hex Address: FD8

**Table 237. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

### Hex Address: FD9

**Table 238. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

### Hex Address: FDA

**Table 239. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

### Hex Address: FDB

**Table 240. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

### Hex Address: FDC

**Table 241. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

### Hex Address: FDD

**Table 242. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

### Hex Address: FDE

**Table 243. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

### Hex Address: FDF

**Table 244. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

### Hex Address: FE0

**Table 245. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

### Hex Address: FE1

**Table 246. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

### Hex Address: FE2

**Table 247. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

### Hex Address: FE3

**Table 248. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

### Hex Address: FE4

**Table 249. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

### Hex Address: FE5

**Table 250. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

### Hex Address: FE6

**Table 251. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

### Hex Address: FE7

**Table 252. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

### Hex Address: FE8

**Table 253. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

### Hex Address: FE9

**Table 254. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

### Hex Address: FEA

**Table 255. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

### Hex Address: FEB

**Table 256. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

### Hex Address: FEC

**Table 257. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0h, FD4h, FD8h, FDCh, FE0h, FE4h, FE8h, FECh | | | | | | | |

### Hex Address: FED

**Table 258. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00h | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1h, FD5h, FD9h, FDDh, FE1h, FE5h, FE9h, FEDh | | | | | | | |

### Hex Address: FEE

**Table 259. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2h, FD6h, FDAh, FDEh, FE2h, FE6h, FEAh, FEEh | | | | | | | |

### Hex Address: FEF

**Table 260. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3h, FD7h, FDBh, FDFh, FE3h, FE7h, FEBh, FEFh | | | | | | | |

# Watchdog Timer

For more information about these Watchdog Timer Control registers, see the Watchdog Timer Control Register Definitions section on page 83.

### Hex Address: FF0

**Table 261. Watchdog Timer Control Register (WDTCTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POR | STOP | WDT | EXT | Reserved | | | SM |
| RESET | See Table 48 on page 84. | | | 0 | | | | |
| R/W | R | | | | | | | |
| Address | FF0h | | | | | | | |

### Hex Address: FF1

**Table 262. Watchdog Timer Reload Upper Byte Register (WDTU)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | WDTU | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF1h | | | | | | | |

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

### Hex Address: FF2

**Table 263. Watchdog Timer Reload High Byte Register (WDTH)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | WDTH | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF2h | | | | | | | |
| Note: *R/W = Read returns the current WDT count value; write sets the appropriate reload value. | | | | | | | | |

### Hex Address: FF3

**Table 264. Watchdog Timer Reload Low Byte Register (WDTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | WDTL | | | | | | | |
| RESET | 1 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF3h | | | | | | | |
| Note: *R/W = Read returns the current WDT count value; write sets the appropriate reload value. | | | | | | | | |

### Hex Addresses: FF4–FF7

This address range is reserved.

# Flash

For more information about these Flash Control registers, see the Flash Control Register Definitions section on page 175.

## Hex Address: FF8

**Table 265. Flash Control Register (FCTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | FCMD | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | W | | | | | | | |
| Address | FF8h | | | | | | | |

**Table 266. Flash Status Register (FSTAT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | FSTAT | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | |
| Address | FF8h | | | | | | | |

## Hex Address: FF9

**Table 267. Page Select Register (FPS)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | INFO_EN | PAGE | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FF9h | | | | | | | |

Table 268. Flash Sector Protect Register (FPROT)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SECT7 | SECT6 | SECT5 | SECT4 | SECT3 | SECT2 | SECT1 | SECT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF9h | | | | | | | |

Note: *R/W = This register is accessible for read operations; it can be written to 1 only via user code.

## Hex Address: FFA

Table 269. Flash Frequency High Byte Register (FFREQH)

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

## Hex Address: FFB

Table 270. Flash Frequency Low Byte Register (FFREQL)

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

## Hex Addresses: FFC–FFF

Refer to the eZ8 CPU Core User Manual (UM0128)

# Packaging

Zilog's F64xx Series of MCUs includes the Z8F1621, Z8F2421, Z8F3221, Z8F4821 and Z8F6421 devices, which are available in the following packages:

- 40-pin Pin Dual Inline Package (PDIP)
- 44-pin Low Profile Quad Flat Package (LQFP)
- 44-pin Plastic Lead Chip Carrier (PLCC)

Zilog's F64xx Series of MCUs also includes the Z8F1622, Z8F2422, Z8F3222, Z8F4822 and Z8F6422 devices, which are available in the following packages:

- 64-pin Low-Profile Quad Flat Package (LQFP)
- 68-pin Plastic Lead Chip Carrier (PLCC)

Lastly, Zilog's F64xx Series of MCUs includes the Z8F4823 and Z8F6423 devices, which are available in the following package:

- 80-pin Quad Flat Package (QFP)

Current diagrams for each of these packages are published in Zilog's Packaging Product Specification (PS0072), which is available free for download from the Zilog website.

# Ordering Information

Order your F64xx Series products from Zilog using the part numbers shown in Table 271. For more information about ordering, please consult your local Zilog sales office. The Sales Location page on the Zilog website lists all regional offices.

**Table 271. Z8 Encore! XP F64xx Series Ordering Matrix**

| Part Number | Flash | RAM | I/O Lines | Interrupts | 16-Bit Timers w/PWM | 10-Bit A/D Channels | $I^2C$ | SPI | UARTs with IrDA | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| **Z8F642x with 64KB Flash, 10-Bit Analog-to-Digital Converter** | | | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | | | |
| Z8F6421PM020SG | 64KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F6421AN020SG | 64KB | 4KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F6421VN020SG | 64KB | 4KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F6422AR020SG | 64KB | 4KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F6422VS020SG | 64KB | 4KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| Z8F6423FT020SG | 64KB | 4KB | 60 | 24 | 4 | 12 | 1 | 1 | 2 | 80-pin QFP package |
| **Extended Temperature: –40°C to +105°C** | | | | | | | | | | |
| Z8F6421PM020EG | 64KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F6421AN020EG | 64KB | 4KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F6421VN020EG | 64KB | 4KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F6422AR020EG | 64KB | 4KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F6422VS020EG | 64KB | 4KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| Z8F6423FT020EG | 64KB | 4KB | 60 | 24 | 4 | 12 | 1 | 1 | 2 | 80-pin QFP package |
| **Automotive/Industrial Temperature: –40°C to +125°C** | | | | | | | | | | |
| Z8F6421PM020AG | 64KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F6421AN020AG | 64KB | 4KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F6421VN020AG | 64KB | 4KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F6422AR020AG | 64KB | 4KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F6422VS020AG | 64KB | 4KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| Z8F6423FT020AG | 64KB | 4KB | 60 | 24 | 4 | 12 | 1 | 1 | 2 | 80-pin QFP package |

Note: Select the 10mm x 10mm package option when choosing the 64-pin LQFP package.

**Table 271. Z8 Encore! XP F64xx Series Ordering Matrix**

| Part Number | Flash | RAM | I/O Lines | Interrupts | 16-Bit Timers w/PWM | 10-Bit A/D Channels | $I^2C$ | SPI | UARTs with IrDA | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| **Z8F482x with 48 KB Flash, 10-Bit Analog-to-Digital Converter** | | | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | | | |
| Z8F4821PM020SG | 48 KB | 4 KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F4821AN020SG | 48 KB | 4 KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F4821VN020SG | 48 KB | 4 KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F4822AR020SG | 48 KB | 4 KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F4822VS020SG | 48 KB | 4 KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| Z8F4823FT020SG | 48 KB | 4 KB | 60 | 24 | 4 | 12 | 1 | 1 | 2 | 80-pin QFP package |
| **Extended Temperature: –40°C to +105°C** | | | | | | | | | | |
| Z8F4821PM020EG | 48 KB | 4 KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F4821AN020EG | 48 KB | 4 KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F4821VN020EG | 48 KB | 4 KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F4822AR020EG | 48 KB | 4 KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F4822VS020EG | 48 KB | 4 KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| Z8F4823FT020EG | 48 KB | 4 KB | 60 | 24 | 4 | 12 | 1 | 1 | 2 | 80-pin QFP package |
| **Automotive/Industrial Temperature: –40°C to +125°C** | | | | | | | | | | |
| Z8F4821PM020AG | 48 KB | 4 KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F4821AN020AG | 48 KB | 4 KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F4821VN020AG | 48 KB | 4 KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F4822AR020AG | 48 KB | 4 KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F4822VS020AG | 48 KB | 4 KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| Z8F4823FT020AG | 48 KB | 4 KB | 60 | 24 | 4 | 12 | 1 | 1 | 2 | 80-pin QFP package |

Note: Select the 10 mm x 10 mm package option when choosing the 64-pin LQFP package.

**Table 271. Z8 Encore! XP F64xx Series Ordering Matrix**

| Part Number | Flash | RAM | I/O Lines | Interrupts | 16-Bit Timers w/PWM | 10-Bit A/D Channels | $I^2C$ | SPI | UARTs with IrDA | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| **Z8F322x with 32KB Flash, 10-Bit Analog-to-Digital Converter** | | | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | | | |
| Z8F3221PM020SG | 32KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F3221AN020SG | 32KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F3221VN020SG | 32KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F3222AR020SG | 32KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F3222VS020SG | 32KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| **Extended Temperature: –40°C to 105°C** | | | | | | | | | | |
| Z8F3221PM020EG | 32KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F3221AN020EG | 32KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F3221VN020EG | 32KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F3222AR020EG | 32KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F3222VS020EG | 32KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| **Automotive/Industrial Temperature: –40°C to 125°C** | | | | | | | | | | |
| Z8F3221PM020AG | 32KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F3221AN020AG | 32KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F3221VN020AG | 32KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F3222AR020AG | 32KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F3222VS020AG | 32KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |

Note: Select the 10mm x 10mm package option when choosing the 64-pin LQFP package.

**Table 271. Z8 Encore! XP F64xx Series Ordering Matrix**

| Part Number | Flash | RAM | I/O Lines | Interrupts | 16-Bit Timers w/PWM | 10-Bit A/D Channels | $I^2C$ | SPI | UARTs with IrDA | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| **Z8F242x with 24KB Flash, 10-Bit Analog-to-Digital Converter** | | | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | | | |
| Z8F2421PM020SG | 24KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F2421AN020SG | 24KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F2421VN020SG | 24KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F2422AR020SG | 24KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F2422VS020SG | 24KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| **Extended Temperature: –40°C to 105°C** | | | | | | | | | | |
| Z8F2421PM020EG | 24KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F2421AN020EG | 24KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F2421VN020EG | 24KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F2422AR020EG | 24KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F2422VS020EG | 24KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| **Automotive/Industrial Temperature: –40°C to 125°C** | | | | | | | | | | |
| Z8F2421PM020AG | 24KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F2421AN020AG | 24KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F2421VN020AG | 24KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F2422AR020AG | 24KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F2422VS020AG | 24KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |

Note: Select the 10mm x 10mm package option when choosing the 64-pin LQFP package.

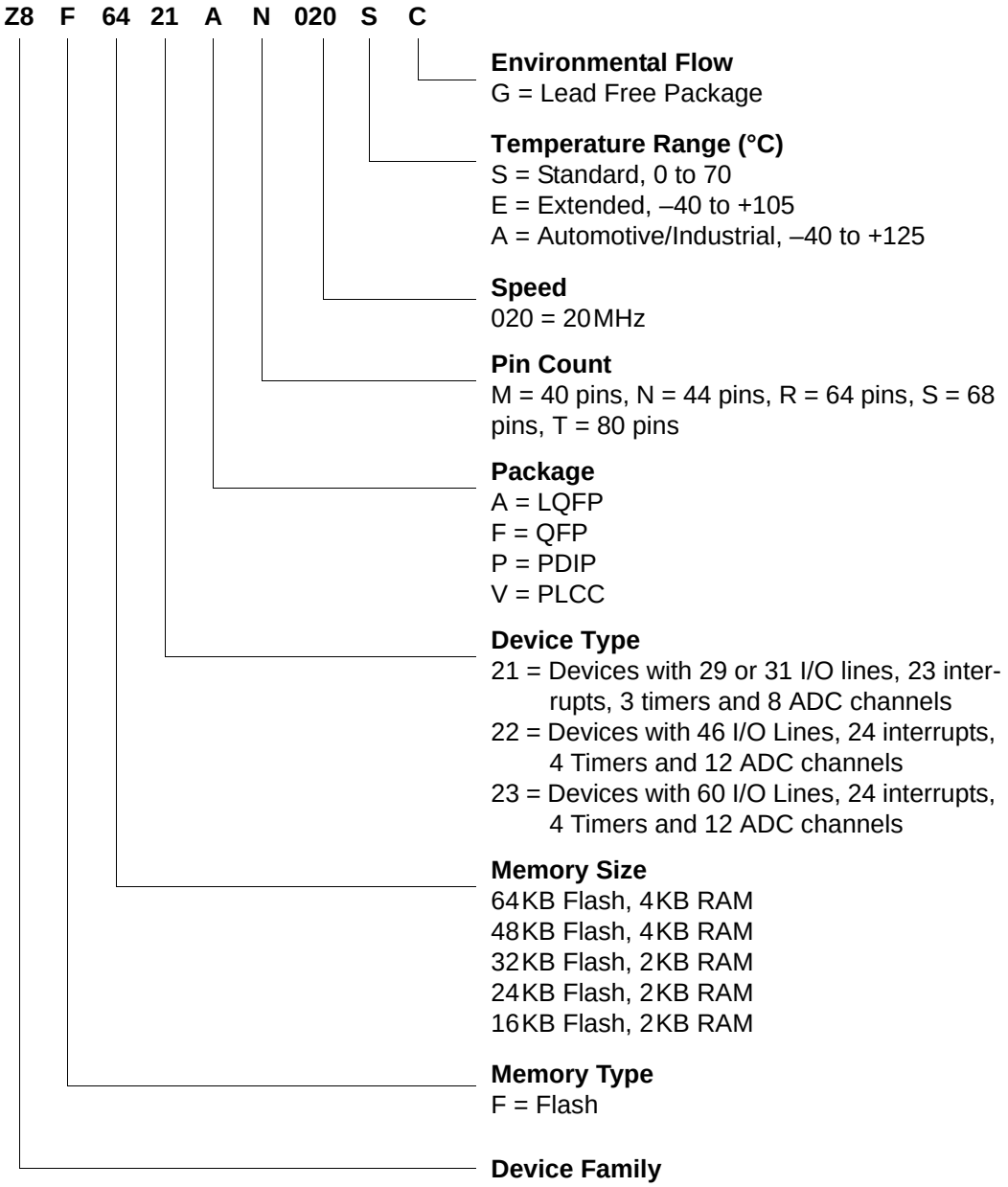
**Table 271. Z8 Encore! XP F64xx Series Ordering Matrix**

| Part Number | Flash | RAM | I/O Lines | Interrupts | 16-Bit Timers w/PWM | 10-Bit A/D Channels | $I^2C$ | SPI | UARTs with IrDA | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| **Z8F162x with 16KB Flash, 10-Bit Analog-to-Digital Converter** | | | | | | | | | | |
| **Standard Temperature: 0°C to 70°C** | | | | | | | | | | |
| Z8F1621PM020SG | 16KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F1621AN020SG | 16KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F1621VN020SG | 16KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F1622AR020SG | 16KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F1622VS020SG | 16KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| **Extended Temperature: –40°C to +105°C** | | | | | | | | | | |
| Z8F1621PM020EG | 16KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F1621AN020EG | 16KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F1621VN020EG | 16KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F1622AR020EG | 16KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F1622VS020EG | 16KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| **Automotive/Industrial Temperature: –40°C to +125°C** | | | | | | | | | | |
| Z8F1621PM020AG | 16KB | 2KB | 29 | 23 | 3 | 8 | 1 | 1 | 2 | 40-pin PDIP package |
| Z8F1621AN020AG | 16KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin LQFP package |
| Z8F1621VN020AG | 16KB | 2KB | 31 | 23 | 3 | 8 | 1 | 1 | 2 | 44-pin PLCC package |
| Z8F1622AR020AG | 16KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 64-pin LQFP package* |
| Z8F1622VS020AG | 16KB | 2KB | 46 | 24 | 4 | 12 | 1 | 1 | 2 | 68-pin PLCC package |
| Z8F64200100KITG | | | | | | | | | | Development Kit |
| ZUSBSC00100ZACG | | | | | | | | | | USB Smart Cable Accessory Kit |
| ZUSBOPTSC01ZACG | | | | | | | | | | Opto-Isolated USB Smart Cable Accessory Kit |
| ZENETSC0100ZACG | | | | | | | | | | Ethernet Smart Cable Accessory Kit |

Note: Select the 10mm x 10mm package option when choosing the 64-pin LQFP package.

# Part Number Suffix Designations

Zilog part numbers consist of a number of component. In the following example, part number Z8F6421AN020SG is an 8-bit Flash MCU with 4KB of program memory in a 44-pin LQFP package, operating with a maximum 20MHz external clock frequency over a 0ºC to +70ºC temperature range and built using environmentally friendly (lead-free) solder.



# Index


## Numerics

10-bit ADC 5

## A

absolute maximum ratings 201
AC characteristics 217
ADC 231
  architecture 162
  automatic power-down 164
  block diagram 163
  continuous conversion 165
  control register 166
  control register definitions 166
  data high byte register 168
  data low bits register 169
  DMA control 166
  electrical characteristics and timing 215
  operation 164
  single-shot conversion 164
ADCCTL register 166
ADCDH register 168
ADCDL register 169
ADCX 231
ADD 231
add - extended addressing 231
add with carry 231
add with carry - extended addressing 231
additional symbols 229
address space 19
ADDX 231
analog signals 16
analog-to-digital converter (ADC) 162
AND 234
ANDX 234
arithmetic instructions 231
assembly language programming 226
assembly language syntax 227

## B

baud rate generator, UART 99
BCLR 232
binary number suffix 229
BIT 232
bit 228
  clear 232
  manipulation instructions 232
  set 232
  set or clear 232
  swap 232
  test and jump 234
  test and jump if non-zero 234
  test and jump if zero 234
bit jump and test if non-zero 234
bit swap 235
block diagram 4
block transfer instructions 232
BRK 234
BSET 232
BSWAP 232, 235
BTJ 234
BTJNZ 234
BTJZ 234

## C

CALL procedure 234
Capture Mode 79
Capture/Compare modes 79
cc 228
CCF 233
characteristics, electrical 201
clear 233
clock phase (SPI) 117
CLR 233
COM 234
compare 79
compare - extended addressing 231
Compare Mode 79

compare with carry 231
compare with carry - extended addressing 231
complement 234
complement carry flag 232, 233
condition code 228
continuous conversion (ADC) 165
Continuous Mode 79
control register definition, UART 99
control register, I2C 145
counter modes 79
CP 231
CPC 231
CPCX 231
CPU and peripheral overview 4
CPU control instructions 233
CPX 231
Customer Feedback Form 305
customer feedback form 294
Customer Information 305

## D

DA 228, 231
data register, I2C 142
DC characteristics 203
debugger, on-chip 184
DEC 231
decimal adjust 231
decrement 231
decrement and jump non-zero 234
decrement word 231
DECW 231
destination operand 229
device, port availability 37
DI 233
direct address 228
direct memory access controller 151
disable interrupts 233
DJNZ 234
DMA
  address high nibble register 156
  configuring DMA0-1 data transfer 151
  configuring for DMA_ADC data transfer 153
  control of ADC 166
  control register 154
  control register definitions 153
  controller 6
  DMA_ADC address register 158
  DMA_ADC control register 159
  DMA_ADC operation 152
  end address low byte register 157
  I/O address register 155
  operation 151
  start/current address low byte register 157
  status register 160
DMAA_STAT register 160
DMAACTL register 159
DMAxCTL register 154, 268, 269
DMAxEND register 157, 269, 270
DMAxH register 156, 268, 270
DMAxI/O address (DMAxIO) 155, 268, 269
DMAxIO register 155, 268, 269
DMAxSTART register 157, 268, 270
dst 229

## E

EI 233
electrical characteristics 201
  ADC 215
  flash memory and timing 214
  GPIO input data sample timing 218
  watch-dog timer 214
enable interrupt 233
ER 228
extended addressing register 228
external pin reset 33
external RC oscillator 213
eZ8 CPU features 4
eZ8 CPU instruction classes 231
eZ8 CPU instruction notation 228
eZ8 CPU instruction set 226
eZ8 CPU instruction summary 235

## F

FCTL register 177, 285
features, Z8 Encore! 1

first opcode map 247
FLAGS 229
flags register 229
flash
  controller 5
  option bit address space 181
  option bit configuration - reset 181
  program memory address 0001h 183
flash memory
  arrangement 171
  byte programming 174
  code protection 173
  configurations 170
  control register definitions 176
  controller bypass 175
  electrical characteristics and timing 214
  flash control register 177, 285
  flash status register 178
  frequency high and low byte registers 180
  mass erase 175
  operation 172
  operation timing 172
  page erase 175
  page select register 178
FPS register 178
FSTAT register 178

## G

Gated Mode 79
general-purpose I/O 37
GPIO 5, 37
  alternate functions 38
  architecture 38
  control register definitions 40
  input data sample timing 218
  interrupts 40
  port A-H address registers 41
  port A-H alternate function sub-registers 43
  port A-H control registers 42
  port A-H data direction sub-registers 42
  port A-H high drive enable sub-registers 45
  port A-H input data registers 47
  port A-H output control sub-registers 44
  port A-H output data registers 47
  port A-H Stop Mode Recovery sub-registers 46
  port availability by device 37
  port input timing 218
  port output timing 219

## H

H 229
HALT 233
Halt Mode 36, 233
hexadecimal number prefix/suffix 229

## I

$I^2C$ 5
  10-bit address read transaction 140
  10-bit address transaction 137
  10-bit addressed slave data transfer format 137
  10-bit receive data format 140
  7-bit address transaction 134
  7-bit address, reading a transaction 139
  7-bit addressed slave data transfer format 134, 135, 136
  7-bit receive data transfer format 139
  baud high and low byte registers 146, 148, 150
  C status register 143, 263
  control register definitions 142
  controller 129
  controller signals 15
  interrupts 131
  operation 130
  SDA and SCL signals 131
  stop and start conditions 133
I2CBRH register 147, 148, 150, 263, 264
I2CBRL register 147, 263
I2CCTL register 145, 263
I2CDATA register 143, 262
I2CSTAT register 143, 263
IM 228
immediate data 228
immediate operand prefix 229
INC 231
increment 231

increment word 231
INCW 231
indexed 229
indirect address prefix 229
indirect register 228
indirect register pair 228
indirect working register 228
indirect working register pair 228
infrared encoder/decoder (IrDA) 110
instruction set, ez8 CPU 226
instructions
ADC 231
ADCX 231
ADD 231
ADDX 231
AND 234
ANDX 234
arithmetic 231
BCLR 232
BIT 232
bit manipulation 232
block transfer 232
BRK 234
BSET 232
BSWAP 232, 235
BTJ 234
BTJNZ 234
BTJZ 234
CALL 234
CCF 232, 233
CLR 233
COM 234
CP 231
CPC 231
CPCX 231
CPU control 233
CPX 231
DA 231
DEC 231
DECW 231
DI 233
DJNZ 234
EI 233
HALT 233
INC 231
INCW 231
IRET 234
JP 234
LD 233
LDC 233
LDCI 232, 233
LDE 233
LDEI 232
LDX 233
LEA 233
load 233
logical 234
MULT 232
NOP 233
OR 234
ORX 234
POP 233
POPX 233
program control 234
PUSH 233
PUSHX 233
RCF 232, 233
RET 234
RL 235
RLC 235
rotate and shift 235
RR 235
RRC 235
SBC 232
SCF 232, 233
SRA 235
SRL 235
SRP 233
STOP 233
SUB 232
SUBX 232
SWAP 235
TCM 232
TCMX 232
TM 232
TMX 232
TRAP 234
watch-dog timer refresh 233

XOR 234
XORX 234
instructions, eZ8 classes of 231
interrupt control register 62
interrupt controller 6, 48
architecture 48
interrupt assertion types 51
interrupt vectors and priority 51
operation 50
register definitions 52
software interrupt assertion 52
interrupt edge select register 61
interrupt port select register 61
interrupt request 0 register 52
interrupt request 1 register 54
interrupt request 2 register 55
interrupt return 234
interrupt vector listing 48
interrupts
not acknowledge 131
receive 131
SPI 121
transmit 131
UART 97
introduction 1
IR 228
Ir 228
IrDA
architecture 110
block diagram 110
control register definitions 113
operation 110
receiving data 112
transmitting data 111
IRET 234
IRQ0 enable high and low bit registers 56
IRQ1 enable high and low bit registers 57
IRQ2 enable high and low bit registers 59
IRR 228
Irr 228

## J

JP 234
jump, conditional, relative, and relative conditional 234

## L

LD 233
LDC 233
LDCI 232, 233
LDE 233
LDEI 232, 233
LDX 233
LEA 233
load 233
load constant 232
load constant to/from program memory 233
load constant with auto-increment addresses 233
load effective address 233
load external data 233
load external data to/from data memory and auto-increment addresses 232
load external to/from data memory and auto-increment addresses 233
load instructions 233
load using extended addressing 233
logical AND 234
logical AND/extended addressing 234
logical exclusive OR 234
logical exclusive OR/extended addressing 234
logical instructions 234
logical OR 234
logical OR/extended addressing 234
low power modes 35

## M

master interrupt enable 50
master-in, slave-out and-in 116
memory
program 20
MISO 116
mode
capture 79
capture/compare 79
continuous 79

counter 79
gated 79
one-shot 79
PWM 79
modes 79
MULT 232
multiply 232
multiprocessor Mode, UART 94

## N

NOP (no operation) 233
not acknowledge interrupt 131
notation
b 228
cc 228
DA 228
ER 228
IM 228
IR 228
Ir 228
IRR 228
Irr 228
p 228
R 228
r 228
RA 229
RR 229
rr 229
vector 229
X 229
notational shorthand 228

## O

OCD
architecture 184
auto-baud detector/generator 187
baud rate limits 187
block diagram 184
breakpoints 188
commands 189
control register 194
data format 187
DBG pin to RS-232 Interface 185
Debug Mode 186
debugger break 234
interface 185
serial errors 188
status register 195
timing 220
OCD commands
execute instruction (12h) 193
read data memory (0Dh) 193
read OCD control register (05h) 191
read OCD revision (00h) 190
read OCD status register (02h) 191
read program counter (07h) 191
read program memory (0Bh) 192
read program memory CRC (0Eh) 193
read register (09h) 192
step instruction (10h) 193
stuff instruction (11h) 193
write data memory (0Ch) 192
write OCD control register (04h) 191
write program counter (06h) 191
write program memory (0Ah) 192
write register (08h) 191
on-chip debugger 6
on-chip debugger (OCD) 184
on-chip debugger signals 17
on-chip oscillator 197
One-Shot Mode 79
opcode map
abbreviations 245
cell description 245
first 247
second after 1Fh 248
OR 234
ordering information 288
ORX 234
oscillator signals 17

## P

p 228
Packaging 287
part number description 293

part selection guide 2
PC 229
peripheral AC and DC electrical characteristics 212
PHASE=0 timing (SPI) 118
PHASE=1 timing (SPI) 119
pin characteristics 18
polarity 228
POP 233
pop using extended addressing 233
POPX 233
port availability, device 37
port input timing (GPIO) 218
port output timing, GPIO 219
power supply signals 17
power-down, automatic (ADC) 164
power-on and voltage brown-out 212
power-on reset (POR) 31
program control instructions 234
program counter 229
program memory 20
PUSH 233
push using extended addressing 233
PUSHX 233
PWM Mode 79
PxADDR register 41, 275, 276, 277, 278, 279, 280, 281, 282
PxCTL register 42, 275, 276, 277, 278, 279, 280, 281, 282

## R

R 228
r 228
RA
    register address 229
RCF 232, 233
receive
    10-bit data format ($I^2C$) 140
    7-bit data transfer format ($I^2C$) 139
    IrDA data 112
receive interrupt 131
receiving UART data-interrupt-driven method 93
receiving UART data-polled method 92
register 126, 155, 228, 265, 268, 269
ADC control (ADCCTL) 166
ADC data high byte (ADCDH) 168
ADC data low bits (ADCDL) 169
baud low and high byte ($I^2C$) 146, 148, 150
baud rate high and low byte (SPI) 127
control (SPI) 123
control, $I^2C$ 145
data, SPI 122
DMA status (DMAA_STAT) 160
DMA_ADC address 158
DMA_ADC control DMAACTL) 159
DMAx address high nibble (DMAxH) 156, 268, 270
DMAx control (DMAxCTL) 154, 268, 269
DMAx end/address low byte (DMAxEND) 157, 269, 270
DMAx start/current address low byte register (DMAxSTART) 157, 268, 270
flash control (FCTL) 177, 285
flash high and low byte (FFREQH and FREEQL) 180
flash page select (FPS) 178
flash status (FSTAT) 178
GPIO port A-H address (PxADDR) 41, 275, 276, 277, 278, 279, 280, 281, 282
GPIO port A-H alternate function sub-registers 43
GPIO port A-H control address (PxCTL) 42, 275, 276, 277, 278, 279, 280, 281, 282
GPIO port A-H data direction sub-registers 42
$I^2C$ baud rate high (I2CBRH) 147, 148, 150, 263, 264
$I^2C$ control (I2CCTL) 145, 263
$I^2C$ data (I2CDATA) 143, 262
$I^2C$ status 143, 263
$I^2C$ status (I2CSTAT) 143, 263
$I^2C$ baud rate low (I2CBRL) 147, 263
Mode, SPI 126
OCD control 194
OCD status 195
SPI baud rate high byte (SPIBRH) 128, 266
SPI baud rate low byte (SPIBRL) 128, 266
SPI control (SPICTL) 123, 265
SPI data (SPIDATA) 123, 264

SPI status (SPISTAT) 124, 265
status, $I^2C$ 143
status, SPI 124
UARTx baud rate high byte (UxBRH) 107, 259, 262
UARTx baud rate low byte (UxBRL) 107, 260, 262
UARTx Control 0 (UxCTL0) 103, 106, 258, 259, 261
UARTx control 1 (UxCTL1) 104, 259, 261
UARTx receive data (UxRXD) 100, 258, 260
UARTx status 0 (UxSTAT0) 101, 258, 260
UARTx status 1 (UxSTAT1) 102, 259, 261
UARTx transmit data (UxTXD) 100, 258, 260
watchdog timer control (WDTCTL) 85, 283
watchdog timer reload high byte (WDTH) 87, 284
watchdog timer reload low byte (WDTL) 87, 284
watchdog timer reload upper byte (WDTU) 86, 283
register file 19
register file address map 23
register pair 229
register pointer 229
reset
and Stop Mode characteristics 29
carry flag 232
controller 6
sources 30
RET 234
return 234
RL 235
RLC 235
rotate and shift instructions 235
rotate left 235
rotate left through carry 235
rotate right 235
rotate right through carry 235
RP 229
RR 229, 235
rr 229
RRC 235

## S

SBC 232
SCF 232, 233
SDA and SCL (IrDA) signals 131
second opcode map after 1Fh 248
serial clock 117
serial peripheral interface (SPI) 114
set carry flag 232, 233
set register pointer 233
shift right arithmetic 235
shift right logical 235
signal descriptions 15
single-shot conversion (ADC) 164
SIO 6
slave data transfer formats ($I^2C$) 137
slave select 117
software trap 234
source operand 229
SP 229
SPI
architecture 114
baud rate generator 121
baud rate high and low byte register 127
clock phase 117
configured as slave 115
control register 123
control register definitions 122
data register 122
error detection 120
interrupts 121
mode fault error 120
mode register 126
multi-master operation 119
operation 116
overrun error 120
signals 116
single master, multiple slave system 115
single master, single slave system 114
status register 124
timing, PHASE = 0 118
timing, PHASE=1 119
SPI controller signals 15
SPI Mode (SPIMODE) 126, 265
SPIBRH register 128, 266

SPIBRL register 128, 266
SPICTL register 123, 265
SPIDATA register 123, 264
SPIMODE register 126, 265
SPISTAT register 124, 265
SRA 235
src 229
SRL 235
SRP 233
stack pointer 229
status register, $I^2C$ 143
STOP 233
Stop Mode 35, 233
Stop Mode Recovery
    sources 33
    using a GPIO port pin transition 34
    using watchdog timer time-out 34
SUB 232
subtract 232
subtract - extended addressing 232
subtract with carry 232
subtract with carry - extended addressing 232
SUBX 232
SWAP 235
swap nibbles 235
symbols, additional 229
system and core resets 30

## T

TCM 232
TCMX 232
test complement under mask 232
test complement under mask - extended addressing 232
test under mask 232
test under mask - extended addressing 232
timer signals 16
timers 6, 63
    architecture 63
    block diagram 64
    Capture Mode 69, 79
    Capture/Compare modes 71, 79
    Compare Mode 70, 79
    Continuous Mode 65, 79
    Counter Mode 66
    Counter modes 79
    Gated Mode 71, 79
    One-Shot Mode 64, 79
    operating mode 64
    PWM Mode 67, 79
    reading the timer count values 72
    reload high and low byte registers 75
    timer control register definitions 73
    timer output signal operation 73
timers 0-3
    control 0 registers 77
    control 1 registers 78
    high and low byte registers 73, 76
TM 232
TMX 232
transmit
    IrDA data 111
transmit interrupt 131
transmitting UART data-interrupt-driven method 91
transmitting UART data-polled method 90
TRAP 234

## U

UART 5, 88
    architecture 88
    asynchronous data format without/with parity 90
    baud rate generator 99
    baud rates table 108
    control register definitions 99
    controller signals 16
    interrupts 97
    Multiprocessor Mode 94
    receiving data using interrupt-driven method 93
    receiving data using the polled method 92
    transmitting data using the interrupt-driven method 91
    transmitting data using the polled method 90
    x baud rate high and low registers 106
    x control 0 and control 1 registers 103

x status 0 and status 1 registers 101, 102
Universal Asynchronous Receiver/Transmitter 88
UxBRH register 107, 259, 262
UxBRL register 107, 260, 262
UxCTL0 register 103, 106, 258, 259, 261
UxCTL1 register 104, 259, 261
UxRXD register 100, 258, 260
UxSTAT0 register 101, 258, 260
UxSTAT1 register 102, 259, 261
UxTXD register 100, 258, 260

## V

vector 229
voltage brownout reset (VBR) 32

## W

watch-dog timer
approximate time-out delay 82
CNTL 32
control register 84
refresh 82
watchdog timer
electrical characteristics and timing 214
interrupt in normal operation 82
interrupt in Stop Mode 82
refresh 233
reload unlock sequence 83
reload upper, high and low registers 86
reset 33
reset in normal operation 83
reset in Stop Mode 83
time-out response 82
WDTCTL register 85, 283
WDTH register 87, 284
WDTL register 87, 284
working register 228
working register pair 229
WTDU register 86, 283

## X

X 229
XOR 234
XORX 234

## Z

Z8 Encore!
block diagram 4
features 1
introduction 1
part selection guide 2

# Customer Support

To share comments, get your technical questions answered or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.