High Performance Microcontrollers

ZNEO® Z16F Series

Product Specification

PS022012-1113

PRELIMINARY

Copyright ©2013 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

©2013 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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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!, ZNEO and Z16F 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 this document's revision history reflects a change from its previous edition. For more details, refer to the corresponding page(s) or appropriate links furnished in the table below.

Date	Revision Level	Section	Description	Page No.
Nov 2013	12	Signal Descriptions	Corrected active status of RD, WR and $\overline{CS}$ signals.	12
Jul 2013	11	Analog Functions	Updated the Analog Functions Block Diagram.	242
Aug 2011	10	Multi-Channel PWM Timer	Per CR#13095, corrected PWMEN description in PWM Control 0 Register (PWMCTL0) table; corrected description in PWM Deadband Register (PWMDB) table and added footnote; added same footnote to PWM Minimum Pulse Width Filter (PWMMPF), PWM Fault Mask Register (PWMFM), and PWM Fault Control Register (PWMFCTL) tables.	125, 127–129, 131
Jun 2011	09	Electrical Characteristics	Corrected V_{COFF} input offset value in Comparator Electrical Characteristics table	347
Aug 2010	08	N/A	Removed ISO information.	ii
		All	Updated logos.	All
		Table 191	Changed the Minimum, Typical and Maximum values for V_{REF} (Externally supplied Voltage Reference only).	346
Jan 2009	07	Timer 0–2 Control 0 Register	Table 62: added "Only Counter Mode should be used with this feature" to Bit 4 description.	109
		Analog Functions	ADC Overview, updated fast conversion time to 2.5µs.	243
		Electrical Characteristics	Updated Table 185.	337
		Internal Precision Oscillator	Removed reference to 32kHz.	336

Feb 2007	06	Independent and Complementary PWM Outputs	Corrected PWM Registers. Updated Edge-Aligned PWM Output figure.	117
		Electrical Characteristics	Replaced 105°C with 125°C in Tables 185 through 192. Added Figures 73 through 75.	337
		I2C Master/Slave Controller	Changes to Software Control of I2C Transactions section.	209
		Packaging	Updated Part Number Suffix Designations section.	359
		Enhanced Serial Peripheral Interface	Throughput section modified.	181
Jul 2006	05	External Interface, General-Purpose Input/Output, DMA Controller, Option Bits, On-Chip Debugger and Electrical Characteristics	Modifications done in the following chapters: External Interface, GPIO, DMA Controller, Option bits, on-chip debugger and Electrical characteristics.	37, 66, 267, 292, 298, 337
		Ordering Information	Ordering Information modified.	356
Jan 2006	04	All	Changed zneo to ZNEO in the entire document.	All
		All	Added TM symbol to ZNEO.	All
		Signal and Pin Descriptions, Interrupt Controller and Analog Functions	Modifications done to following chapters: Pin description, Interrupt controller and Analog functions.	7, 80, 242
		Ordering Information	Ordering Information modified.	356

Table of Contents

Revision History iii
List of Figures xvi
List of Tables xix
Document Objectives xxv
About This Manual xxv
Intended Audience xxv
Manual Conventions xxv
Safeguards xxvii
Introduction 1
Features 1
Block Diagram 2
ZNEO CPU Features 3
External Interface 3
Flash Controller 3
Random Access Memory 3
ZNEO Peripheral Overview 3
10-Bit Analog-to-Digital Converter with Programmable Gain Amplifier 4
Analog Comparator 4
Operational Amplifier 4
General-Purpose Input/Output 4
Universal Asynchronous Receiver/Transmitter 4
Infrared Encoder/Decoders 4
Inter-Integrated Circuit Master/Slave Controller 4
Enhanced Serial Peripheral Interface 4
DMA Controller 5
Pulse Width Modulator 5
Standard Timers 5
Interrupt Controller 5
Crystal Oscillator 5
Reset Controller 5
On-Chip Debugger 6
Signal and Pin Descriptions 7
Available Packages 7
Pin Configurations 7
Signal Descriptions 12
Pin Characteristics 16

Address Space 17
Memory Map 17
Internal Nonvolatile Memory 18
Internal RAM 19
Input/Output Memory 19
Input/Output Memory Precautions 19
CPU Control Registers 19
External Memory 20
Endianness 20
Bus Widths 21
Peripheral Address Map 23
External Interface 37
External Interface Signals 37
Chip Selects 38
Tools Compatibility Guidelines 39
External WAIT Pin Operation 40
Operation 41
Wait State Generator 41
ISA-Compatible Mode 42
External Interface Control Register Definitions 42
External Interface Control Register 42
Chip Select Control Registers 43
External Interface Timing 47
External Interface Write Timing, Normal Mode 47
External Interface Write Timing, ISA Mode 49
External Interface Read Timing, Normal Mode 50
External Interface Read Timing, ISA Mode 53
Reset and Stop Mode Recovery 56
Reset Types 56
System Reset 57
Power-On Reset 57
Voltage Brown-Out Reset 58
Watchdog Timer Reset 59
External Pin Reset 59
External Reset Indicator 60
User Reset 60
Fault Detect Logic Reset 60
Stop Mode Recovery 60
Stop Mode Recovery Using WDT Time-Out 61
Stop Mode Recovery Using a GPIO Port Pin Transition 61

Reset Status and Control Register 62
Low-Power Modes 64
Stop Mode 64
Halt Mode 65
Peripheral-Level Power Control 65
Power Control Option Bits 65
General-Purpose Input/Output 66
GPIO Port Availability by Device 66
Architecture 66
GPIO Alternate Functions 67
GPIO Interrupts 70
GPIO Control Register Definitions 71
Port A-K Input Data Registers 71
Port A-K Output Data Registers 72
Port A-K Data Direction Registers 73
Port A-K High Drive Enable Registers 74
Port A-K Alternate Function High and Low Registers 74
Port A-K Output Control Registers 75
Port A-K Pull-Up Enable Registers 76
Port A-K Stop Mode Recovery Source Enable Registers 77
Port A IRQ MUX1 Register 77
Port A IRQ MUX Register 78
Port A IRQ Edge Register 78
Port C IRQ MUX Register 79
Interrupt Controller 80
Interrupt Vector Listing 80
Architecture 82
Operation 82
Master Interrupt Enable 82
Interrupt Vectors and Priority 83
System Exceptions 83
Interrupt Assertion 84
System Exception Status Registers 84
Last IRQ Register 85
Interrupt Request 0 Register 86
Interrupt Request 1 Register 87
Interrupt Request 2 Register 88
IRQ0 Enable High and Low Bit Registers 90
IRQ1 Enable High and Low Bit Registers 91
IRQ2 Enable High and Low Bit Registers 92

Timers 95
Architecture 95
Operation 96
Timer Operating Modes 97
Reading Timer Count Values 106
Timer Control Register Definitions 106
Timer 0–2 High and Low Byte Registers 106
Timer X Reload High and Low Byte Registers 108
Timer 0–2 PWM High and Low Byte Registers 109
Timer 0–2 Control Registers 109
Multi-Channel PWM Timer 114
Architecture 114
Operation 115
PWM Option Bits 115
PWM Reload Event 116
PWM Prescaler 116
PWM Period and Count Resolution 116
PWM Duty Cycle Registers 118
Independent and Complementary PWM Outputs 118
Manual Off-State Control of PWM Output Channels 119
Deadband Insertion 119
Minimum PWM Pulse Width Filter 119
Synchronization of PWM and ADC 120
Synchronized Current-Sense Sample and Hold 120
PWM Timer and Fault Interrupts 120
Fault Detection and Protection 120
PWM Operation in CPU Halt Mode 121
PWM Operation in CPU Stop Mode 121
Observing the State of PWM Output Channels 121
PWM Control Register Definitions 121
PWM High and Low Byte Registers 122
PWM Reload High and Low Byte Registers 123
PWM 0–2 Duty Cycle High and Low Byte Registers 124
PWM Control 0 Register 125
PWM Control 1 Register 126
PWM Deadband Register 127
PWM Minimum Pulse Width Filter 128
PWM Fault Mask Register 128
PWM Fault Status Register 130
PWM Fault Control Register 131
PWM Input Sample Register 132

PWM Output Control Register 133
Current-Sense Sample and Hold Control Registers 134
LIN-UART 135
Architecture 135
Operation 136
Data Format for Standard UART Modes 136
Transmitting Data using the Polled Method 137
Transmitting Data Using Interrupt-Driven Method 138
Receiving Data Using the Polled Method 139
Receiving Data Using the Interrupt-Driven Method 140
Clear To Send Operation 141
External Driver Enable 141
LIN-UART Special Modes 142
Multiprocessor (9-Bit) Mode 142
LIN Protocol Mode 144
LIN-UART Interrupts 147
LIN-UART DMA Interface 150
LIN-UART Baud Rate Generator 150
Noise Filter 151
Architecture 151
Operation 152
LIN-UART Control Register Definitions 153
LIN-UART Transmit Data Register 153
LIN-UART Receive Data Register 154
LIN-UART Status 0 Register 155
LIN-UART Mode Select and Status Register 158
LIN-UART Control 0 Register 160
LIN-UART Control 1 Registers 162
LIN-UART Address Compare Register 166
LIN-UART Baud Rate High and Low Byte Registers 166
Infrared Encoder/Decoder 172
Architecture 172
Operation 172
Transmitting IrDA Data 173
Receiving IrDA Data 174
Infrared Encoder/Decoder Control Register Definitions 175
Enhanced Serial Peripheral Interface 176
Architecture 176
ESPI Signals 178
Master-In/Slave-Out 178

Master-Out/Slave-In 178
Serial Clock 178
Slave Select 179
ESPI Register Overview 179
Comparison with Basic SPI Block 179
Operation 180
Throughput 181
ESPI Clock Phase and Polarity Control 181
Modes of Operation 183
SPI Protocol Configuration 186
Error Detection 189
ESPI Interrupts 190
DMA Interface 191
ESPI Baud Rate Generator 192
ESPI Control Register Definitions 193
ESPI Data Register 193
ESPI Transmit Data Command Register 193
ESPI Control Register 194
ESPI Mode Register 197
ESPI Status Register 198
ESPI State Register 200
ESPI Baud Rate High and Low Byte Registers 201
I^2C Master/Slave Controller 203
Architecture 203
I^2C Master/Slave Controller Registers 205
Comparison with Master Mode only I^2C Controller 205
Operation 206
SDA and SCL Signals 206
I^2C Interrupts 206
Start and Stop Conditions 208
Software Control of I^2C Transactions 209
Master Transactions 209
Slave Transactions 217
DMA Control of I^2C Transactions 224
I^2C Control Register Definitions 227
I^2C Data Register 227
I^2C Interrupt Status Register 227
I^2C Control Register 229
I^2C Baud Rate High and Low Byte Registers 230
I^2C State Register 232
I^2C Mode Register 236

I2C Slave Address Register 237
Watchdog Timer 238
Operation 238
Watchdog Timer Refresh 239
Watchdog Timer Time-Out Response 239
Watchdog Timer Reload Unlock Sequence 240
Watchdog Timer Register Definitions 241
Watchdog Timer Reload High and Low Byte Registers 241
Analog Functions 242
ADC Overview 243
Architecture 243
Operation 243
ADC Timing 244
ADC Interrupts 245
ADC0 Timer 0 Capture 245
ADC Convert on Read 245
Reference Buffer, RBUF 246
Internal Voltage Reference Generator 246
ADC Control Register Definitions 246
ADC0 Control Register 0 246
ADC0 Data High Byte Register 247
ADC0 Data Low Bits Register 248
Sample Settling Time Register 248
Sample Time Register 249
ADC Clock Prescale Register 249
ADC0 Max Register 250
ADC Timer 0 Capture Register 251
Comparator and Operational Amplifier Overview 251
Comparator Operation 252
Operational Amplifier Operation 252
Interrupts 253
Comparator Control Register Definitions 253
Comparator and Operational Amplifier Control Register 253
Flash Memory 255
Information Area 256
Operation 257
Timing Using the Flash Frequency Register 257
Flash Read Protection 258
Flash Write/Erase Protection 258
Programming 259

Page Erase . . . 259
Mass Erase . . . 260
Flash Controller Bypass . . . 260
Flash Controller Behavior using the On-Chip Debugger . . . 260
Flash Control Register Definitions . . . 261
Flash Command Register . . . 261
Flash Status Register . . . 262
Flash Control Register . . . 263
Flash Sector Protect Register . . . 264
Flash Page Select Register . . . 265
Flash Frequency Register . . . 265
DMA Controller . . . 267
DMA Features . . . 267
DMA Block Diagram . . . 267
DMA Description . . . 269
DMA Register Description . . . 269
DMA Control Bit Definitions . . . 271
DMA Watermark . . . 273
DMA Peripheral Interface signals . . . 273
Buffer Closure . . . 274
DMA Modes . . . 274
Linked List Mode . . . 277
DMA Priority . . . 280
DMA Interrupts . . . 281
DMA Request Select Register . . . 282
DMA Control Registers . . . 285
DMA Control Register . . . 285
DMA X Transfer Length Register . . . 286
DMA Destination Address . . . 287
DMA Source Address Registers . . . 288
DMA List Address Register . . . 289
External DMA Signals . . . 290
DMA Timing . . . 290
Option Bits . . . 292
Operation . . . 292
Option Bit Address Space . . . 292
Program Memory Address 0001h . . . 294
Program Memory Address 0002h . . . 295
Program Memory Address 0003h . . . 295
Information Area . . . 296

IPO Trim Registers (Information Area Address 0021h and 0022h) 296
ADC Reference Voltage Trim (Information Area Address 0023h) 297
On-Chip Debugger 298
Architecture 298
Operation 299
On-Chip Debug Enable 299
Serial Interface 300
Serial Data Format 300
Baud Rate Generator 301
Auto-Baud Detector 302
Line Control 303
9-Bit Mode 303
Start Bit Flow Control 304
Initialization 304
Initialization during Reset 305
Debug Lock 305
Error Reset 305
DEBUG Halt Mode 306
Reading and Writing Memory 306
Reading Memory CRC 307
Breakpoints 307
Instruction Trace 308
On-Chip Debugger Commands 309
Cyclic Redundancy Check 313
Memory Cyclic Redundancy Check 313
UART Mode 313
Serial Errors 314
Interrupts 314
DBG Pin as a GPIO Pin 314
Control Register Definitions 315
Receive Data Register 315
Transmit Data Register 315
Baud Rate Reload Register 316
Line Control Register 316
Status Register 318
Control Register 319
OCD Control Register 321
OCD Status Register 323
Hardware Breakpoint Registers 324
Trace Control Register 325
Trace Address Register 326

On-Chip Oscillator 327
Operating Modes 327
Crystal Oscillator Operation 327
Oscillator Operation with an External RC Network 329
Oscillator Control 331
Operation 331
System Clock Selection 331
Clock Selection Following System Reset 332
Clock Failure Detection and Recovery 333
Oscillator Control Register Definitions 333
Oscillator Control Register 333
Oscillator Divide Register 335
Internal Precision Oscillator 336
Operation 336
Electrical Characteristics 337
Absolute Maximum Ratings 337
DC Characteristics 339
On-Chip Peripheral AC and DC Electrical Characteristics 344
AC Characteristics 349
General Purpose I/O Port Input Data Sample Timing 350
On-Chip Debugger Timing 350
SPI Master Mode Timing 351
SPI Slave Mode Timing 351
I^2C Timing 353
UART Timing 353
Packaging 356
Ordering Information 356
Part Number Suffix Designations 359
Precharacterization Product 359
Index 360
Customer Support 366

List of Figures

Figure 1. Motor Control MCUs Z16F Series Block Diagram 2
Figure 2. Z16F2810 MCU, 64-Pin Low-Profile Quad Flat Package (LQFP) 8
Figure 3. Z16F2810 MCU, 68-Pin Plastic Leaded Chip Carrier (PLCC) 9
Figure 4. ZNEO Z16F Series, 80-Pin Quad Flat Package (QFP) 10
Figure 5. ZNEO Z16F Series, 100-Pin Low-Profile Quad Flat Package (LQFP) . . . 11
Figure 6. Physical Memory Map 18
Figure 7. Endianness of Words and Quads 21
Figure 8. Alignment of Word and Quad Operations on 16-bit Memories 22
Figure 9. Chip Select Boundary Addressing with 128 KB Internal Flash 38
Figure 10. External Interface Wait State Operation Example (Write Operation) 41
Figure 11. External Interface Timing for a Write Operation, Normal Mode 48
Figure 12. External Interface Timing for a Write Operation, ISA Mode 50
Figure 13. External Interface Timing for a Read Operation, Normal Mode 52
Figure 14. External Interface Timing for a Read Operation, 2 Wait States and 1 Post Read Wait State 53
Figure 15. External Interface Timing for a Read Operation, ISA Mode 55
Figure 16. Power-On Reset Operation 58
Figure 17. Voltage Brown-Out Reset Operation 59
Figure 18. GPIO Port Pin Block Diagram 67
Figure 19. Interrupt Controller Block Diagram 82
Figure 20. Timer Block Diagram 96
Figure 21. PWM Block Diagram 115
Figure 22. Edge-Aligned PWM Output 117
Figure 23. Center-Aligned PWM Output 117
Figure 24. LIN-UART Block Diagram 136
Figure 25. LIN-UART Asynchronous Data Format without Parity 137
Figure 26. LIN-UART Asynchronous Data Format with Parity 137
Figure 27. LIN-UART Driver Enable Signal Timing (shown with 1 Stop Bit and Parity) 142
Figure 28. LIN-UART Asynchronous Multiprocessor Mode Data Format 143
Figure 29. LIN-UART Receiver Interrupt Service Routine Flow 149
Figure 30. Noise Filter System Block Diagram 152
Figure 31. Noise Filter Operation 153

Figure 32. Infrared Data Communication System Block Diagram 172
Figure 33. Infrared Data Transmission 173
Figure 34. Infrared Data Reception 174
Figure 35. ESPI Block Diagram 177
Figure 36. ESPI Timing when PHASE = 0 182
Figure 37. ESPI Timing when PHASE = 1 183
Figure 38. SPI Mode (SSMD = 000) 185
Figure 39. I2S Mode (SSMD = 010) 186
Figure 40. ESPI Configured as an SPI Master in a Single Master, Single Slave System 187
Figure 41. ESPI Configured as an SPI Master in a Single Master, Multiple Slave System 187
Figure 42. ESPI Configured as an SPI Slave 189
Figure 43. I2C Controller Block Diagram 204
Figure 44. Data Transfer Format, Master Write Transaction with 7-Bit Addressing . 211
Figure 45. Data Transfer Format, Master Write Transaction with 10-Bit Addressing 212
Figure 46. Data Transfer Format, Master Read Transaction with 7-Bit Addressing . 214
Figure 47. Data Transfer Format, Master Read Transaction with 10-Bit Addressing 215
Figure 48. Data Transfer Format, Slave Receive Transaction with 7-Bit Addressing 218
Figure 49. Data Transfer Format, Slave Receive Transaction with 10-Bit Addressing 220
Figure 50. Data Transfer Format, Slave Transmit Transaction with 7-Bit Addressing 221
Figure 51. Data Transfer Format, Slave Transmit Transaction with 10-Bit Addressing 222
Figure 52. Analog Functions Block Diagram 242
Figure 53. ADC Timing Diagram 244
Figure 54. ADC Convert Timing 245
Figure 55. Flash Memory Arrangement 256
Figure 56. DMA Block Diagram 268
Figure 57. DMA Channel Registers 269
Figure 58. Direct DMA Diagram 275
Figure 59. Linked List Diagram 278
Figure 60. External DMA Transfer 290
Figure 61. External ISA DMA transfer 291
Figure 62. On-Chip Debugger Block Diagram 299
Figure 63. Interfacing a Serial Pin with an RS-232 Interface, #1 of 2 300
Figure 64. Interfacing a Serial Pin with an RS-232 Interface, #2 of 2 300

Figure 65. OCD Serial Data Format 301
Figure 66. Output Driver when Drive High and Open Drain Enabled 303
Figure 67. 9-Bit Mode 304
Figure 68. Start Bit Flow Control 304
Figure 69. Initialization During Reset 305
Figure 70. Recommended 20 MHz Crystal Oscillator Configuration 328
Figure 71. Connecting the On-Chip Oscillator to an External RC Network 329
Figure 72. Typical RC Oscillator Frequency as a Function of the External Capacitance with a 15 kΩ Resistor 330
Figure 73. Typical I_{DD} Versus System Clock Frequency 341
Figure 74. Typical Halt Mode IDD Versus System Clock Frequency 342
Figure 75. Stop Mode Current Versus V_{DD} 343
Figure 76. Port Input Sample Timing 350
Figure 77. SPI Master Mode Timing 351
Figure 78. SPI Slave Mode Timing 352
Figure 79. I^2C Timing 353
Figure 80. UART Timing with CTS 354
Figure 81. UART Timing without CTS 355

List of Tables

Table 1. ZNEO Z16F Series Package Options 7
Table 2. Signal Descriptions 12
Table 3. Pin Characteristics of the ZNEO CPU 16
Table 4. Reserved Memory Map Example 19
Table 5. ZNEO CPU Control Registers 20
Table 6. Register File Address Map 23
Table 7. External Interface Signals Description 37
Table 8. Example Usage of Chip Selects 39
Table 9. External Interface Control Register (EXTCT) 42
Table 10. External Chip Select Control Registers High (EXTCSxH) 43
Table 11. External Chip Select Control Registers Low for $\overline{CS0}$ (EXTCS0L) 44
Table 12. External Chip Select Control Registers Low for $\overline{CS1}$ (EXTCS1L) 45
Table 13. External Chip Select Control Registers Low for $\overline{CS2}$ to $\overline{CS5}$ (EXTCSxL) . 46
Table 14. External Interface Timing for a Write Operation, Normal Mode 47
Table 15. External Interface Timing for a Write Operation, ISA Mode 49
Table 16. External Interface Timing for a Read Operation, Normal Mode 51
Table 17. External Interface Timing for a Read Operation, ISA Mode 54
Table 18. Reset and Stop Mode Recovery Characteristics and Latency 56
Table 19. System Reset Sources and Resulting Reset Action 57
Table 20. Stop Mode Recovery Sources and Resulting Action 61
Table 21. Reset Status and Control Register (RSTSCR) 62
Table 22. Reset Status Register Values Following Reset 63
Table 23. GPIO Port Availability by Device 66
Table 24. Port Alternate Function Mapping 68
Table 25. Port A-K Input Data Registers (PxIN) 71
Table 26. Port A-K Output Data Registers (PxOUT) 72
Table 27. Port A-K Data Direction Registers (PxDD) 73
Table 28. Port A-K High Drive Enable Registers (PxHDE) 74
Table 29. Port A-K Alternate Function Low Registers (PxAFL) 75
Table 30. Alternate Function Enabling 75
Table 31. Port A-K Alternate Function High Registers (PxAFH) 75
Table 32. Port A-K Output Control Registers (PxOC) 76
Table 33. Port A-K Pull-Up Enable Registers (PxPUE) 76

Table 34. Port A-K Stop Mode Recovery Source Enable Registers (PxSMRE) 77
Table 35. Port A IRQ MUX1 Register (PAIMUX1) 77
Table 36. Port A IRQ MUX Register (PAIMUX) 78
Table 37. Port A IRQ Edge Register (PAIEDGE) 78
Table 38. Port C IRQ MUX Register (PCIMUX) 79
Table 39. Interrupt Vectors in Order of Priority 81
Table 40. Interrupt Vector Placement 82
Table 41. System Exception Register High (SYSEXCPH) 84
Table 42. System Exception Register Low (SYSEXCPL) 85
Table 43. Last IRQ Register (LASTIRQ) 86
Table 44. Interrupt Request 0 Register (IRQ0) and Interrupt Request 0 Set Register (IRQ0SET) 86
Table 45. Interrupt Request1 Register (IRQ1) and Interrupt Request1 Set Register (IRQ1SET) 88
Table 46. Interrupt Request 2 Register (IRQ2) and Interrupt Request 2 Set Register (IRQ2SET) 89
Table 47. IRQ0 Enable and Priority Encoding 90
Table 48. IRQ0 Enable High Bit Register (IRQ0ENH) 90
Table 49. IRQ0 Enable Low Bit Register (IRQ0ENL) 91
Table 50. IRQ1 Enable and Priority Encoding 91
Table 51. IRQ1 Enable High Bit Register (IRQ1ENH) 92
Table 52. IRQ1 Enable Low Bit Register (IRQ1ENL) 92
Table 53. IRQ2 Enable and Priority Encoding 92
Table 54. IRQ2 Enable Low Bit Register (IRQ2ENL) 93
Table 55. IRQ2 Enable High Bit Register (IRQ2ENH) 93
Table 56. Timer 0–2 High Byte Register (TxH) 107
Table 57. Timer 0–2 Low Byte Register (TXL) 107
Table 58. Timer 0–2 Reload High Byte Register (TxRH) 108
Table 59. Timer 0–2 Reload Low Byte Register (TxRL) 108
Table 60. Timer 0–2 PWM High Byte Register (TxPWMH) 109
Table 61. Timer 0–2 PWM Low Byte Register (TxPWML) 109
Table 62. Timer 0–2 Control 0 Register (TxCTL0) 110
Table 63. Timer 0–2 Control 1 Register (TxCTL1) 111
Table 64. PWM High Byte Register (PWMH) 122
Table 65. PWM Low Byte Register (PWML) 122
Table 66. PWM Reload High Byte Register (PWMRH) 123

Table 67. PWM Reload Low Byte Register (PWMRL) 123
Table 68. PWM 0–2 H/L Duty Cycle High Byte Register (PWMHxDH, PWMLxDH) 124
Table 69. PWM Control 0 Register (PWMCTL0) 125
Table 70. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMHxDL, PWMLxDL) 125
Table 71. PWM Control 1 Register (PWMCTL1) 126
Table 72. PWM Deadband Register (PWMDB) 127
Table 73. PWM Minimum Pulse Width Filter (PWMMPF) 128
Table 74. PWM Fault Mask Register (PWMFM) 129
Table 75. PWM Fault Status Register (PWMFSTAT) 130
Table 76. PWM Fault Control Register (PWMFCTL) 131
Table 77. PWM Input Sample Register (PWMIN) 132
Table 78. PWM Output Control Register (PWMOUT) 133
Table 79. Current-Sense Sample and Hold Control Register (CSSHR0 and CSSHR1) 134
Table 80. LIN-UART Transmit Data Register (UxTXD) 154
Table 81. LIN-UART Receive Data Register (UxRXD) 154
Table 82. LIN-UART Status 0 Register, Standard UART Mode (UxSTAT0) 155
Table 83. LIN-UART Status 0 Register, LIN Mode (UxSTAT0) 156
Table 84. LIN-UART Mode Select and Status Register (UxMDSTAT) 158
Table 85. Multiprocessor Mode Status Field (MSEL = 000b) 159
Table 86. Digital Noise Filter Mode Status Field (MSEL = 001b) 159
Table 87. LIN Mode Status Field (MSEL = 010b) 159
Table 88. Hardware Revision Mode Status Field (MSEL = 111b) 159
Table 89. LIN-UART Control 0 Register (UxCTL0) 160
Table 90. MultiProcessor Control Register (UxCTL1 with MSEL = 000b) 162
Table 91. Noise Filter Control Register (UxCTL1 with MSEL = 001b) 164
Table 92. LIN Control Register (UxCTL1 with MSEL = 010b) 165
Table 93. LIN-UART Address Compare Register (UxADDR) 166
Table 94. LIN-UART Baud Rate High Byte Register (UxBRH) 166
Table 95. LIN-UART Baud Rate Low Byte Register (UxBRL) 167
Table 96. LIN-UART Baud Rates 169
Table 97. ESPI Registers 179
Table 98. ESPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation . . 181
Table 99. ESPI Tx DMA Descriptor Command Field 191

Table 100. ESPI Tx DMA Descriptor Status Field 191
Table 101. ESPI Rx DMA Descriptor Status Field 192
Table 102. ESPI Data Register (ESPIDATA) 193
Table 103. ESPI Transmit Data Command Register (ESPITDCR) 194
Table 104. ESPI Control Register (ESPICTL) 194
Table 105. ESPI Mode Register (ESPIMODE) 197
Table 106. ESPI Status Register (ESPISTAT) 198
Table 107. ESPI State Register (ESPISTATE) 200
Table 108. ESPISTATE Values and Description 200
Table 109. ESPI Baud Rate High Byte Register (ESPIBRH) 202
Table 110. ESPI Baud Rate Low Byte Register (ESPIBRL) 202
Table 111. I^2C Master/Slave Controller Registers 205
Table 112. I^2C Data Register (I2CDATA) 227
Table 113. I^2C Interrupt Status Register (I2CISTAT) 228
Table 114. I^2C Control Register (I2CCTL) 229
Table 115. I^2C Baud Rate High Byte Register (I2CBRH) 231
Table 116. I^2C Baud Rate Low Byte Register (I2CBRL) 231
Table 117. I^2C State Register (I2CSTATE), Description when DIAG = 0 232
Table 118. I^2C State Register (I2CSTATE), Description when DIAG = 1 233
Table 119. I2CSTATE_H 233
Table 120. I2CSTATE_L 234
Table 121. I^2C Mode Register (I2CMODE) 236
Table 122. I^2C Slave Address Register (I2CSLVAD) 237
Table 123. Watchdog Timer Approximate Time-Out Delays 239
Table 124. Watchdog Timer Reload High Byte Register (WDTH) 241
Table 125. Watchdog Timer Reload Low Byte Register (WDTL) 241
Table 126. ADC0 Control Register 0 (ADC0CTL) 246
Table 127. ADC0 Data High Byte Register (ADC0D_H) 247
Table 128. ADC0 Data Low Bits Register (ADC0D_L) 248
Table 129. Sample and Settling Time (ADCSST) 248
Table 130. Sample Time (ADCST) 249
Table 131. ADC Clock Prescale Register (ADCCP) 249
Table 132. ADC0 MAX Register (ADC0MAX) 250
Table 133. ADC Timer 0 Capture Register, High Byte (ADCTCAP_H) 251
Table 134. ADC Timer 0 Capture Register, Low Byte (ADCTCAP_L) 251
Table 135. Comparator and Op Amp Control Register (CMPOPC) 253

Table 136. Flash Memory Configurations 255
Table 137. Flash Memory Sector Addresses 255
Table 138. ZNEO Z16F Series Information Area Map 257
Table 139. Flash Command Register (FCMD) 261
Table 140. Flash Status Register (FSTAT) 262
Table 141. Flash Control Register (FCTL) 263
Table 142. Flash Sector Protect Register (FSECT) 264
Table 143. Flash Page Select Register (FPAGE) 265
Table 144. Flash Frequency Register (FFREQ) 266
Table 145. Linked List Descriptor 271
Table 146. DMA Priority 281
Table 147. DMA Bandwidth Selection 281
Table 148. DMA Select Register (DAMxREQSEL) 282
Table 149. DMA Control Register A (DMAxCTL) 285
Table 150. DMA X Transfer Length High Register (DMAxTXLNH) 286
Table 151. DMA X Transfer Length Low Register (DMAxTXLNL) 287
Table 152. DMA X Destination Address Register Upper (DMAxDARU) 287
Table 153. DMA X Destination Address Register High (DMAxDARH) 287
Table 154. DMA X Destination Address Register Low (DMAxDARL) 287
Table 155. DMA X Source Address Register Upper DMAxSARU 288
Table 156. DMA X Source Address Register High (DMAxSARH) 288
Table 157. DMA X Source Address Register Low (DMAxSARL) 288
Table 158. DMA X List Address Register Upper DMAxLARU 289
Table 159. DMA X List Address Register High (DMAxLARH) 289
Table 160. DMA X List Address Register Low (DMAxLARL) 289
Table 161. Option Bits At Program Memory Address 0000h 293
Table 162. Options Bits at Program Memory Address 0001h 294
Table 163. Options Bits at Program Memory Address 0002h 295
Table 164. Options Bits at Program Memory Address 0003h 295
Table 165. IPO Trim 1 (IPOTRIM1) 296
Table 166. IPO Trim 2 (IPOTRIM2) 296
Table 167. ADC Reference Voltage Trim (ADCTRIM) 297
Table 168. OCD Baud Rate Limits 302
Table 169. On-Chip Debugger Commands 312
Table 170. Receive Data Register (DBGRXD) 315
Table 171. Transmit Data Register (DBGTXD) 315

Table 172. Baud Rate Reload Register (DBGBR) 316
Table 173. Line Control Register (DBGLCR) 316
Table 174. Status Register (DBGSTAT) 318
Table 175. Control Register (DBGCTL) 319
Table 176. OCD Control Register (OCDCTL) 321
Table 177. OCD Status Register (OCDSTAT) 323
Table 178. Hardware Breakpoint Register (HWBPn) 324
Table 179. Trace Control Register (TRACECTL) 325
Table 180. Trace Address (TRACEADDR) 326
Table 181. Recommended Crystal Oscillator Specifications (20 MHz Operation) . . . 328
Table 182. Oscillator Configuration and Selection 332
Table 183. Oscillator Control Register (OSCCTL) 334
Table 184. Oscillator Divide Register (OSCDIV) 335
Table 185. Absolute Maximum Ratings 337
Table 186. DC Characteristics 339
Table 187. POR and VBO Electrical Characteristics and Timing 344
Table 188. Reset and Stop Mode Recovery Pin Timing 344
Table 189. Flash Memory Electrical Characteristics and Timing 345
Table 190. Watchdog Timer Electrical Characteristics and Timing 345
Table 191. ADC Electrical Characteristics and Timing 346
Table 192. Comparator Electrical Characteristics 347
Table 193. Operational Amplifier Electrical Characteristics 347
Table 194. AC Characteristics 349
Table 195. GPIO Port Input Timing 350
Table 196. On-Chip Debugger Timing 350
Table 197. SPI Master Mode Timing 351
Table 198. SPI Slave Mode Timing 352
Table 199. I^2C Timing 353
Table 200. UART Timing with CTS 354
Table 201. UART Timing without CTS 355
Table 202. ZNEO Part Selection Guide 356
Table 203. ZNEO Z16F Series Part Numbering 357

Document Objectives

This product specification provides detailed operating information about the Z16F2810, Z16F2811, Z16F3211, and Z16F6411 devices within Zilog's ZNEO Family of products. In this document, these four devices are collectively referred as the ZNEO or the ZNEO Z16F Series, unless specifically stated otherwise.

About This Manual

Zilog recommends that you read and understand the content contained in this product specification before setting up and using your ZNEO Z16F Series products. However, because we recognize that there are different styles of learning, this specification is designed to be used either as a procedural manual or a reference guide to important data.

Intended Audience

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

Manual Conventions

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

Courier New Typeface

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

Example. FLAGS[1] is SMRF.

Hexadecimal Values

Hexadecimal values are designated by lowercase h suffix and appear in the `Courier New` typeface.

Example. R1 is set to `F8h`.

Brackets

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

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

Braces

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

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

Parentheses

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

Example. (R1) is the memory location referenced by the address contained in the working register R1.

Parentheses/Bracket Combinations

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

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

Use of the Words Set, Reset, and Clear

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

Notation for Bits and Similar Registers

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

Example. ADDR[15:0] refers to bit 15 through bit 0 of the address.

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

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

Use of Initial Uppercase Letters

Initial uppercase letters designate settings and modes in general text.

Example 1. The receiver forces the SCL line to Low.

Example 2. Stop Mode.

Use of All Uppercase Letters

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

Example 1. The bus is considered BUSY after the Start condition.

Example 2. A Start command triggers the processing of the initialization sequence.

Bit Numbering

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

Safeguards

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

Caution: Indicates that a procedure or file may become corrupted if you do not follow instructions.

Introduction

Zilog's ZNEO® Z16F family of products are optimized for demanding applications. The ZNEO line of Zilog® microcontroller products is based on the ZNEO CPU.

Features

ZNEO family of products include the following features:

- 20 MHz ZNEO CPU
- 128 KB internal Flash memory with 16-bit access and In-Circuit Programming (ICP)
- 4 KB internal RAM with 16-bit access
- External interface allows seamless connection to external data memory and peripheral with:
  - Six chip selects with programmable wait states
  - 24-bit address bus supports 16 MB
  - Selectable 8-bit or 16-bit data bus widths
  - Programmable chip select signal polarity
  - ISA-compatible mode
- 12-channel, 10-bit Analog-to-Digital Converter (ADC)
- Operational Amplifier
- Analog Comparator
- 4-channel Direct Memory Access (DMA) controller supports internal or external DMA requests
- Two full-duplex 9-bit Universal Asynchronous Receiver/Transmitter (UARTs) with support for Local Interconnect Network (LIN) and Infrared Data Association (IrDA)
- Internal Precision Oscillator (IPO)
- Inter-Integrated Circuit (I^2C) master/slave controller
- Enhanced Serial Peripheral Interface (ESPI)
- 12-bit Pulse Width Modulation (PWM) module with three complementary pairs or six independent PWM outputs with deadband generation and fault trip input
- Three standard 16-bit timers with Capture, Compare and PWM capability
- Watchdog Timer (WDT) with internal RC oscillator

- 76 General-Purpose Input/Output (GPIO) pins
- 24 interrupts with programmable priority
- On-Chip Debugger (OCD)
- Voltage Brown-Out (VBO) protection
- Power-On Reset (POR)
- 2.7V to 3.6V operating voltage with 5 V-tolerant inputs
- 0°C to +70°C standard temperature and –40°C to +105°C extended temperature operating ranges

Block Diagram

Figure 1 displays the architecture of the ZNEO® Z16F Series MCU.

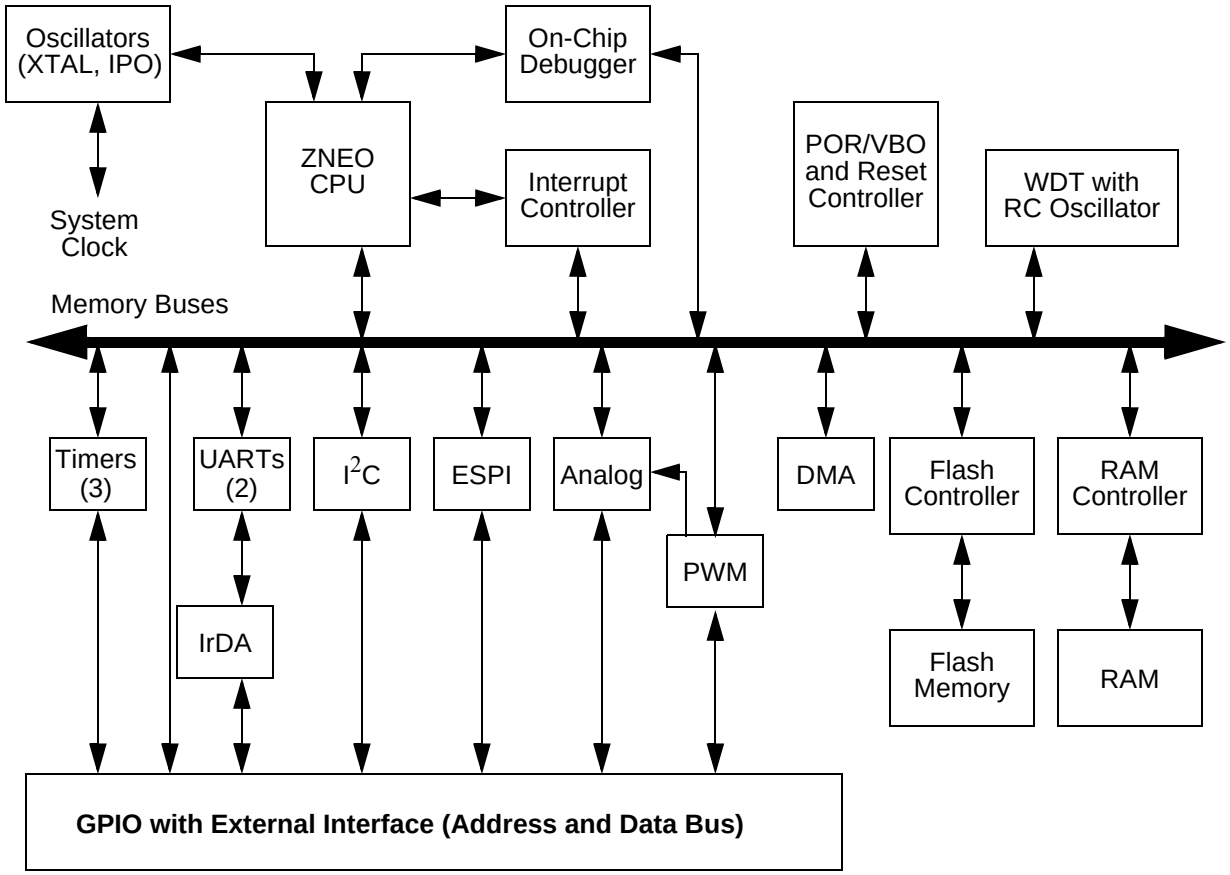


Figure 1. Motor Control MCUs Z16F Series Block Diagram

ZNEO CPU Features

Zilog's ZNEO® CPU meets the continuing demand for faster and more code-efficient microcontrollers. The ZNEO CPU features are as follows:

- 16MB of program memory address space for object code and data with 8-bit or 16-bit data paths
- 8-bit, 16-bit and 32-bit ALU operations
- 24-bit stack with overflow protection
- Direct register-to-register architecture allows each memory address to function as an accumulator to improve execution time and decrease the required program memory
- New instructions improve execution efficiency for code developed using higher-level programming languages including 'C'
- Pipelined instructions: Fetch, Decode and Execute

For more information about the ZNEO CPU, refer to the ZNEO CPU Core User Manual (UM0188), available for download at www.zilog.com.

External Interface

The external interface allows seamless connection to external memory and peripherals. A 24-bit address bus and a selectable 8-bit or 16-bit data bus allows parallel access up to 16MB. The programmable nature of the external interface supports connection to various bus styles. More GPIO pins are utilized by controlling address and control signals bitwise.

Flash Controller

The Motor Control MCUs products contain 128 KB of internal Flash memory. The Flash controller programs and erases the Flash memory. ZNEO CPU accesses 16-bits at a time of internal Flash memory to improve the processor throughput. A sector protection scheme allows flexible protection of user code.

Random Access Memory

An internal RAM of 4 KB provides storage space for data, variables and stack operations. Like Flash memory, ZNEO CPU accesses 16-bits at a time of internal RAM to improve the processor performance.

ZNEO Peripheral Overview

The peripheral features of the ZNEO CPU are described in this section.

10-Bit Analog-to-Digital Converter with Programmable Gain Amplifier

The ADC converts an analog input signal to a 10-bit binary number. The ADC accepts inputs from 12 different analog input sources.

Analog Comparator

It features an on-chip analog comparator with external input pins.

Operational Amplifier

It features a two-input, one-output operational amplifier.

General-Purpose Input/Output

The Motor Control MCUs features 76 GPIO pins. Each pin is individually programmable.

Universal Asynchronous Receiver/Transmitter

It contains two fully-featured UARTs with LIN protocol support. The UART communication is full-duplex and capable of handling asynchronous data transfers. The UARTs support 8-bit and 9-bit data modes, selectable parity and an efficient bus transceiver driver enable signal for controlling a multi-transceiver bus, such as RS-485.

Infrared Encoder/Decoders

The ZNEO Z16F Series products contain two fully-functional, high-performance UART to Infrared Encoder/Decoders (Endecs). Each infrared endec is integrated with an on-chip UART to allow easy communication between the ZNEO Z16F Series device and IrDA physical layer specification Version 1.3-compliant infrared transceivers. Infrared communication provides secure, reliable, low-cost and point-to-point communication between PCs, PDAs, cell phones, printers and other infrared enabled devices.

Inter-Integrated Circuit Master/Slave Controller

The I^2C controller makes Z16F2811 compatible with the I^2C protocol. It consists of two bidirectional bus lines, a serial data (SDA) line and a serial clock (SCL) line. The I^2C operates as a Master and/or Slave and supports multi-master bus arbitration.

Enhanced Serial Peripheral Interface

The ESPI allows the data exchange between ZNEO Z16F Series and other peripheral devices such as electrically erasable programmable read-only memory (EEPROMs),

ADCs and integrated service digital network (ISDN) devices. The SPI is a full-duplex, synchronous, character-oriented channel which supports a four-wire interface.

DMA Controller

The Motor Control MCUs features a 4-channel DMA for efficient transfer of data between peripherals and/or memories. The DMA Controller supports data transfers to and from both internal and external devices.

Pulse Width Modulator

The Motor Control MCUs features a flexible PWM module with three complementary pairs or six independent PWM outputs supporting deadband operation and fault protection trip input. These features provide multiphase control capability for a variety of motor types and ensure safe operation of the motor by providing immediate shutdown of the PWM pins during Fault condition.

Standard Timers

Three 16-bit reloadable timers are used for timing/counting events and PWM signal generation. These timers provide a 16-bit programmable reload counter and operate in One-Shot, Continuous, Gated, Capture, Compare, Capture and Compare and PWM modes. The PWM function provides two complementary output signals with programmable dead-time insertion.

Interrupt Controller

The Motor Control MCUs products support three levels of programmable interrupt priority. The interrupt sources include internal peripherals, GPIO pins and system fault detection.

Crystal Oscillator

The on-chip crystal oscillator features programmable gain to support crystals and ceramic resonators from 32 kHz to 20 MHz. The oscillator is also used with external RC networks or clock drivers.

Reset Controller

The Motor Control MCUs is reset using the $\overline{\text{RESET}}$ pin, POR, WDT, Stop Mode Recovery or VBO warning signal. The bidirectional $\overline{\text{RESET}}$ pin also provides a system RESET output indicator.

On-Chip Debugger

The ZNEO Z16F Series features an integrated OCD. The OCD provides a rich-set of debugging capabilities, such as reading and writing memory, programming the Flash, setting breakpoints and executing code. A single-pin interface provides communication to the OCD.

Signal and Pin Descriptions

The ZNEO® Z16F Series products are available in various package styles and pin configurations. This chapter describes the signals and available pin configurations for each package style. For more information about the physical package specifications, see the Packaging chapter on page 356.

Available Packages

Table 1 lists the package styles available for each device within the ZNEO Z16F Series product line.

Table 1. ZNEO Z16F Series Package Options

Part Number	64-pin LQFP	68-pin PLCC	80-Pin QFP	100-pin LQFP
Z16F2811			X	X
Z16F2810*	X	X	X	
Z16F6411			X	X
Z16F3211			X	X

Note: *The Z16F2810 MCU does not feature an external bus interface.

Pin Configurations

Figures 2 through 5 display the configurations of all of the packages available in the ZNEO Z16F Series. For description of each signal, see Table 2 on page 12.

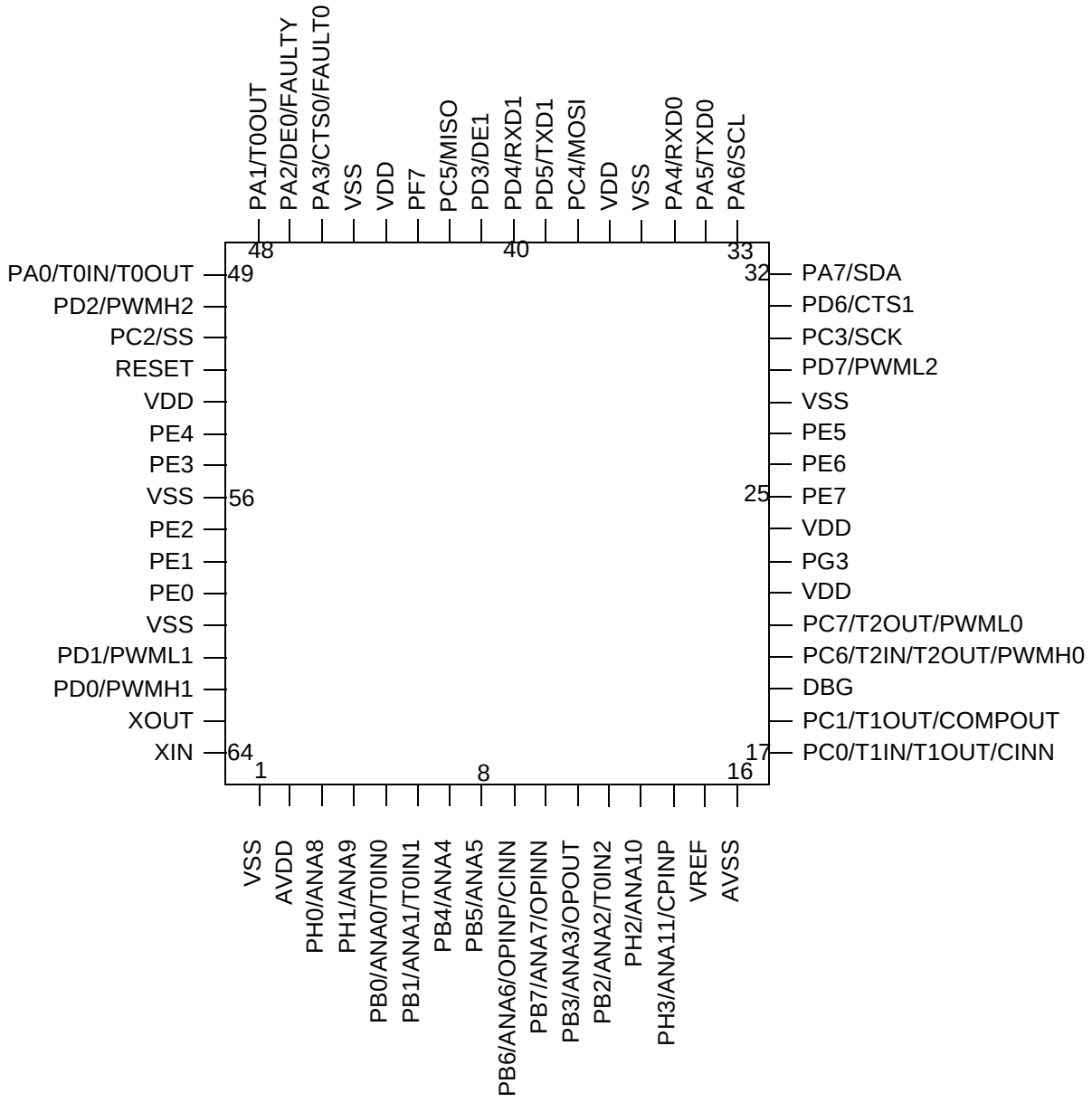


Figure 2. Z16F2810 MCU, 64-Pin Low-Profile Quad Flat Package (LQFP)

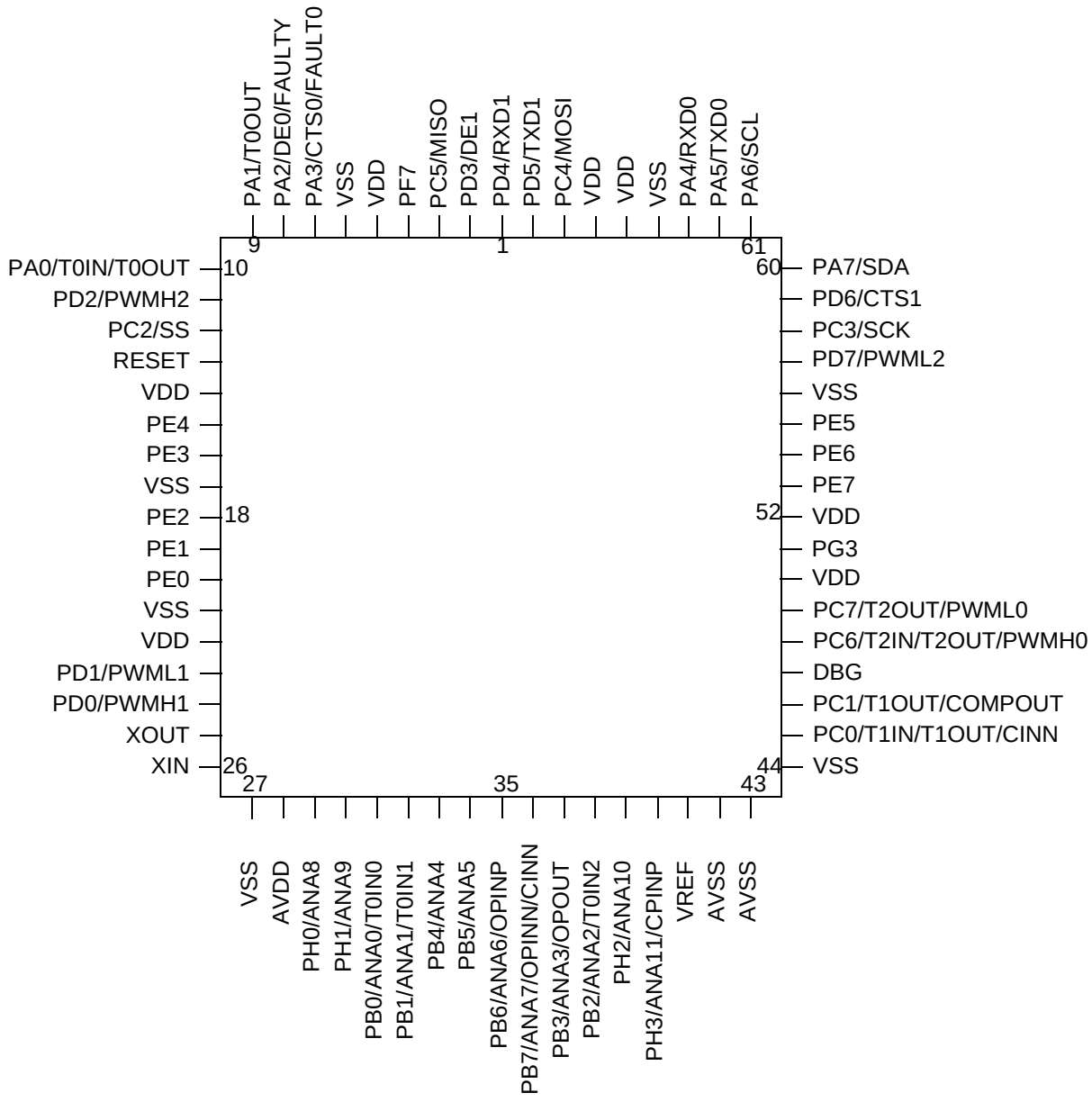


Figure 3. Z16F2810 MCU, 68-Pin Plastic Leaded Chip Carrier (PLCC)

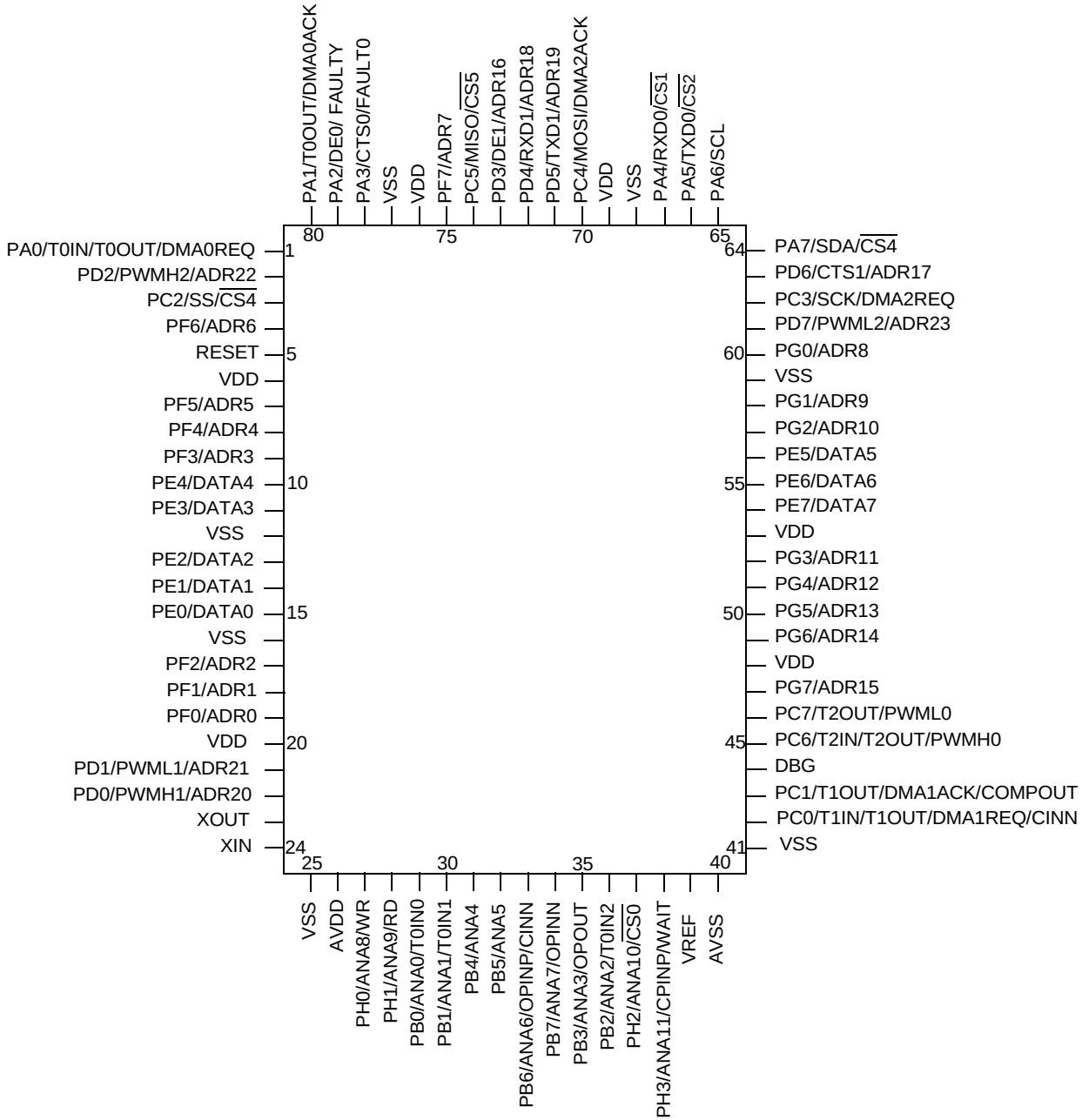


Figure 4. ZNEO Z16F Series, 80-Pin Quad Flat Package (QFP)

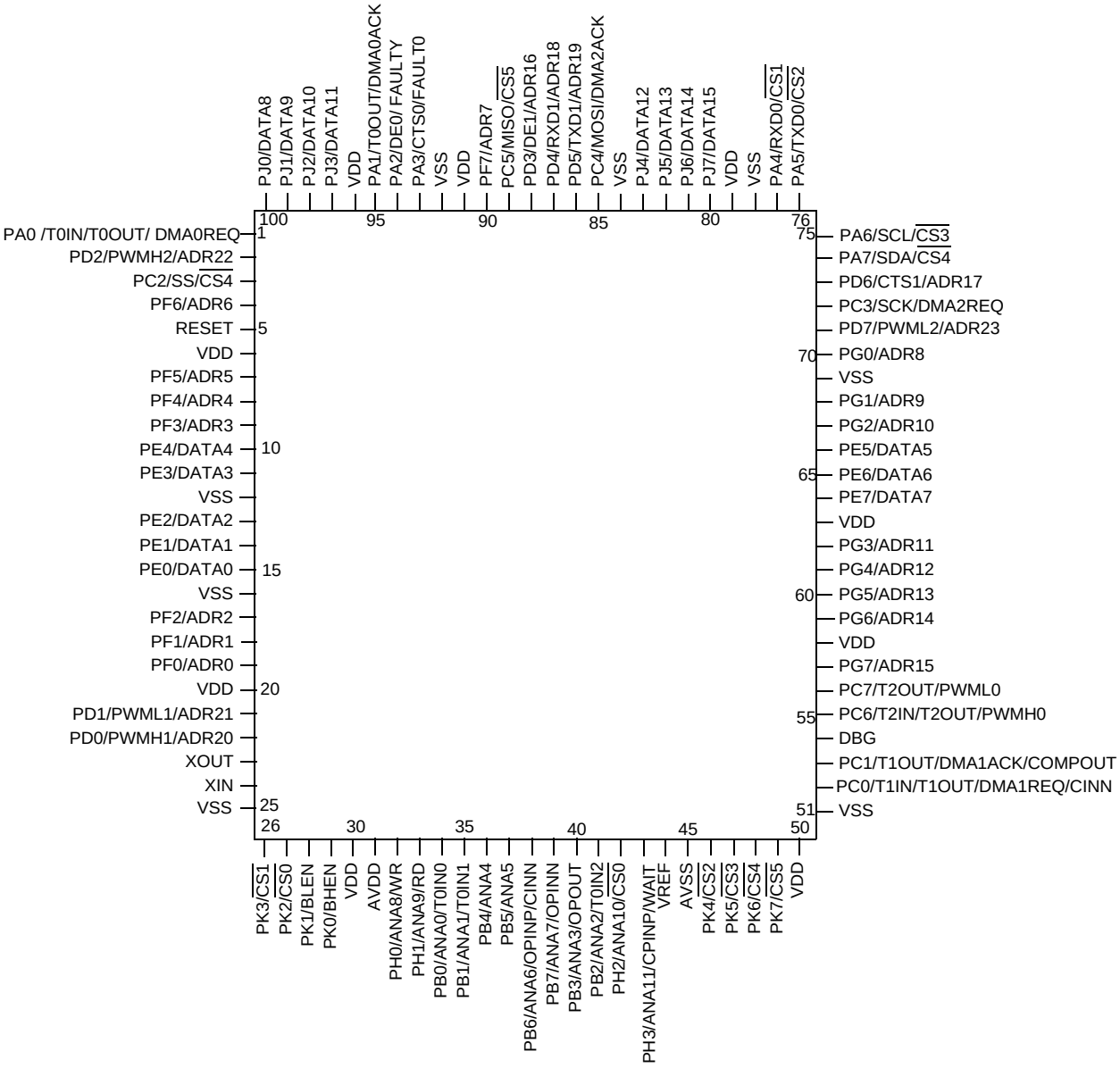


Figure 5. ZNEO Z16F Series, 100-Pin Low-Profile Quad Flat Package (LQFP)

Signal Descriptions

Table 2 describes the ZNEO signals. To determine the signals available for the specific package styles, see the Pin Configurations section on page 7. Most of the signals described in Table 2 are multiplexed with GPIO pins. These signals are available as alternate functions on the GPIO pins. For more details about the GPIO alternate functions, see the General-Purpose Input/Output chapter on page 66.

Table 2. Signal Descriptions

Signal Mnemonic	I/O	Description
General-Purpose Input/Output Ports A–K		
PA[7:0]	I/O	Port A[7:0]: These pins are used for GPIO
PB[7:0]	I/O	Port B[7:0]: These pins are used for GPIO
PC[7:0]	I/O	Port C[7:0]: These pins are used for GPIO
PD[7:0]	I/O	Port D[7:0]: These pins are used for GPIO
PE[7:0]	I/O	Port E[7:0]: These pins are used for GPIO
PF[7:0]	I/O	Port F[7:0]: These pins are used for GPIO
PG[7:0]	I/O	Port G[7:0]: These pins are used for GPIO
PH[3:0]	I/O	Port H[3:0]: These pins are used for GPIO
PJ[7:0]	I/O	Port J[7:0]: These pins are used for GPIO
PK[7:0]	I/O	Port K[7:0]: These pins are used for GPIO
External Interface		
ADR[23:0]	O	Address bus: When the associated GPIO pins are configured for alternate function and the external interface is enabled, these pins function as output pin only. The address bus signals are driven to 0, when execution is out of internal program memory. The address bus alternate functions are individually enabled and disabled.
DATA[15:0]	I/O	Data bus: When the associated GPIO pins are configured for alternate function and the external interface is enabled, these pins functions as input/output. The data bus alternate functions are individually enabled and disabled. When write operation is not performed through the external interface, these signals are tri-stated. The data bus is enabled as either 8-bits (DATA[7:0] only) or 16-bits (DATA[15:0]).
$\overline{RD}$	O	Read output: This pin is the Read output signal from the external interface. Assertion of the $\overline{RD}$ signal indicates that the ZNEO CPU is performing a read operation from the external memory or peripheral.

Table 2. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
$\overline{WR}$	O	Write output: This pin is the $\overline{Write}$ output signal from the external interface. Assertion of the $\overline{WR}$ signal indicates that the ZNEO CPU is performing a write operation to the external memory or peripheral.
$\overline{CS0}$/$\overline{CS1}$/$\overline{CS2}$ $\overline{CS3}$/$\overline{CS4}$/$\overline{CS5}$	O	Chip select outputs: These pins $\overline{are}$ the chip select output signals from the external interface. The $\overline{CS}$ output pins have programmable polarity through the external interface control register.
$\overline{BHEN}$/$\overline{BLEN}$	O	Byte high enable and byte low enable indicators.
$\overline{WAIT}$	I	Wait input: Asserting this input signal will pause the CPU to provide slower external peripherals more time to complete bus transactions through the external interface.
Direct Memory Access Controller		
$\overline{DMA0REQ}$ DMA1REQ DMA2REQ	I	DMA request inputs: Each of the DMA channels have an external request input which allows external peripherals to request access to the address and data buses for data transfer.
$\overline{DMA0ACK}$ DMA1ACK DMA2ACK	O	DMA request outputs: Each of the DMA channels have an acknowledge indicator output to notify external peripherals that their request for access to address and data buses has been approved.
Inter-Integrated Circuit Controller		
SCL	I/O	Serial clock: an input or an output clock for the I^2C. When the GPIO pin is configured for alternate function to enable the SCL function, this pin is open-drain.
SDA	I/O	Serial data: This open-drain pin transfers data between the I^2C and a slave. When the GPIO pin is configured for alternate function to enable the SDA function, this pin is open-drain.
Enhanced Serial Peripheral Interface Controller		
$\overline{SS}$	I/O	Slave select: This signal is an output or an input. If ZNEO is the SPI master, this pin is configured as the slave select output. If ZNEO is the SPI slave, this pin is an input slave select.
SCK	I/O	SPI serial clock: The SPI master supplies this pin. If the ZNEO Z16F Series device is the SPI master, this pin is an output. If the ZNEO Z16F Series device is the SPI slave, this pin is an input.
MOSI	I/O	Master-Out/Slave-In: This signal is the data output from the SPI master device and the data input to the SPI slave device.
MISO	I/O	Master-In/Slave-Out: This pin is the data input to the SPI master device and the data output from the SPI slave device.

Table 2. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
UART Controllers		
TXD0/TXD1	O	Transmit data: These signals transmit outputs from the UARTs.
RXD0/RXD1	I	Receive data: These signals receives inputs for the UARTs and IrDAs.
$\overline{\text{CTS0}}$/$\overline{\text{CTS1}}$	I	Clear to Send: These signals are control inputs for the UARTs.
DE0/DE1	O	Driver Enable (DE): 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 is used to ensure an external RS-485 driver is enabled when data is transmitted by the UART.
General-Purpose Timers		
T0OUT/$\overline{\text{T0OUT}}$ T1OUT/$\overline{\text{T1OUT}}$ T2OUT/$\overline{\text{T2OUT}}$	O	General-purpose timer outputs: These signals are output pins from the timers.
T0IN/T0IN1/T0IN2 /T1IN/T2IN	I	General-purpose timer inputs: These signals are used as the capture, gating and counter inputs.
Pulse-Width Modulator for Motor Control		
PWMH0/PWMH1/ PWMH2	O	PWM High output.
PWML0/PWML1/ PWML2	O	PWM Low output.
FAULT0/FAULTY	I	PWM Fault condition input: FAULT0 and FAULTY are active Low.
Analog		
ANA[11:0]	I	Analog input: These signals are inputs to the ADC.
VREF	I	ADC reference voltage input or internal reference output: The VREF pin must be capacitively coupled to analog ground, if the internal voltage reference is selected as the ADC reference voltage. A 10 mF capacitor is recommended.
CINP	I	Comparator positive input
CINN	I	Comparator negative input
COMPOUT	O	Comparator output
OPINP	I	Operational amplifier positive input
OPINN	I	Operational amplifier negative input
OPOUT	O	Operational amplifier output

Table 2. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
Oscillators		
X_{IN}	I	External crystal input: the input pin to the crystal oscillator. A crystal is connected between it and the X_{OUT} pin to form the oscillator. In addition, this pin is used with external RC networks or external clock drivers to provide the system clock to the system.
X_{OUT}	O	External crystal output: This pin is the output of crystal oscillator. A crystal is connected between it and the X_{IN} pin to form the oscillator. This pin must be left unconnected when not using a crystal.
On-Chip Debugger		
DBG	I/O	Debug: This pin is the control and data input and output to and from the OCD. **Caution:** For operation of the OCD, all power pins (V_{DD} and AVDD) must be supplied with power and all ground pins (V_{SS} and AVSS) must be grounded. This pin is open-drain and must have an external pull-up resistor to ensure proper operation.
Reset		
$\overline{\text{RESET}}$	I/O	**RESET**: Bidirectional $\overline{\text{RESET}}$ signals generates a Reset when asserted (driven Low) and drives a Low output when the ZNEO is in Reset.
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 3 provides information about the characteristics of each pin available on the ZNEO products. The data in Table 3 are sorted alphabetically by pin symbol mnemonic.

Table 3. Pin Characteristics of the ZNEO CPU

Symbol Mnemonic	Direction	Reset Direction	Active Low/High	Tri–State Output	Internal Pull-up or Pull-down	Schmitt Trigger Input	Open Drain Output
AV_{DD}	N/A	N/A	N/A	N/A	No	No	N/A
AV_{SS}	N/A	N/A	N/A	N/A	No	No	N/A
DBG	I/O	I	N/A	Yes	Pull-up	Yes	Yes
PA[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PB[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PC[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PD[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PE[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PF[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PG[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PH[3:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PJ[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PK[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
RESET	I/O	I	Low	N/A	Pull-up	Yes	Yes
V_{REF}	I/O	I	N/A	Yes	N/A	No	No
V_{DD}	N/A	N/A	N/A	N/A	No	No	N/A
V_{SS}	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	N/A	No	No	No

Note: X represents integers 0, 1,... to indicate multiple pins with symbol mnemonics which differ only by an integer.

Address Space

The ZNEO CPU features a unique architecture with a single, unified 24-bit address space. It supports up to four memory areas:

- Internal nonvolatile memory (Flash, EEPROM, EPROM or ROM).
- Internal RAM.
- Internal I/O memory (internal peripherals).
- External memory (and/or memory-mapped peripherals).

The 24-bit address space supports up to 16MB (16,777,216 bytes) of memory. The ZNEO CPU accesses any two of the above memory areas in parallel. In addition, the ZNEO CPU supports three different data widths:

- Byte (8-bit)
- Word (16-bit)
- Quad (32-bit)

The ZNEO CPU accesses memories of different bus width:

- 8-bit wide memories
- 16-bit wide memories

Memory Map

The memory map of the ZNEO CPU displayed in Figure 6 shows the locations of internal nonvolatile memory, internal RAM and internal I/O memory. External memory, however, is placed at addresses which are not occupied by internal memory.

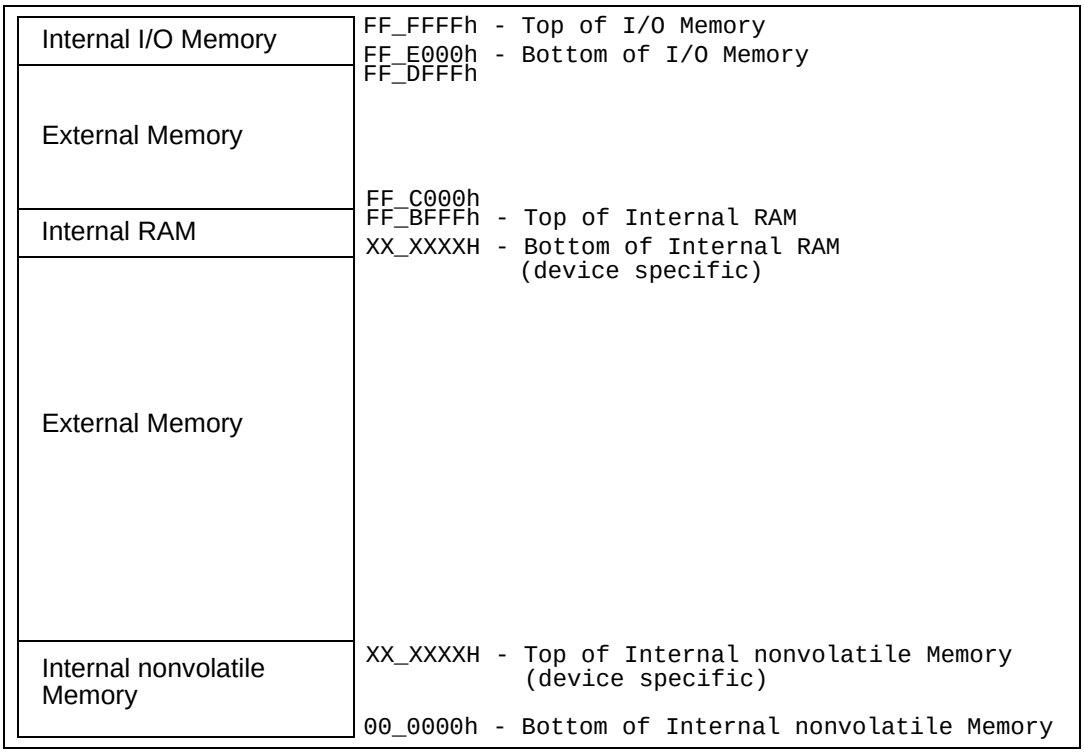


Figure 6. Physical Memory Map

To determine the amount of internal RAM and internal nonvolatile memory available for the specific device, see the Ordering Information section on page 356.

Internal Nonvolatile Memory

Internal nonvolatile memory contains executable program code, constants and data. For each product within the ZNEO CPU family, a memory block beginning at address 00_0000h is reserved for user option bits and system vectors (for example, RESET, Trap, Interrupts and System Exceptions, etc.). Table 4 provides an example of reserved memory map for a ZNEO CPU product with 24 interrupt vectors.

Table 4. Reserved Memory Map Example

Memory Address (Hex)	Description
00_0000–00_0003	Option bits
00_0004–00_0007	RESET vector
00_0008–00_000B	System exception vector
00_000C–00_000F	Privileged trap vector
00_0010–00_006F	Interrupt vectors

Internal RAM

Internal RAM is mainly employed for data and stacks. However, internal RAM also contains program code for execution. Most ZNEO CPU devices contain some internal RAM. The top (highest address) of internal RAM is always located at address FF_BFFFh. The bottom (lowest address) of internal RAM is a function of the amount of internal RAM available. To determine the amount of internal RAM available, see the Ordering Information section on page 356.

Input/Output Memory

The ZNEO CPU supports 8 KB (8,192 bytes) of I/O memory space located at addresses FF_E000h through FF_FFFFh. The I/O memory addresses are reserved for control of the ZNEO CPU, the on-chip peripherals and the I/O ports. Refer to the device-specific Product Specification for descriptions of the peripheral and I/O control registers. Attempts to read or execute from unavailable I/O memory addresses returns FFh. Attempts to write to unavailable I/O memory addresses produce no effect.

Input/Output Memory Precautions

Some control registers within the I/O memory provide read-only or write-only access. When accessing these read-only or write-only registers, ensure that the instructions do not attempt to read from a write-only register or conversely write to a read-only register.

CPU Control Registers

Some registers are reserved within 8KB of I/O memory for ZNEO CPU control; these ZNEO CPU Control registers are listed in Table 5. For information about the operation of the ZNEO CPU Control registers, refer to the ZNEO CPU Core User Manual (UM0188), available for download at www.zilog.com.

Table 5. ZNEO CPU Control Registers

Address (Hex)	Register Description	Register Mnemonic
FF_E004-FF_E007	Program counter overflow	PCOV
FF_E00C-FF_E00F	Stack pointer overflow	SPOV
FF_E010	Flags	FLAGS
FF_E012	CPU control	CPUCTL

External Memory

Many ZNEO CPU products support external data and address buses for connecting to additional external memories and/or memory-mapped peripherals. The external addresses are used for storing program code, data, constants and stack, etc. Attempts to read from or write to unavailable external addresses is undefined.

Endianness

The ZNEO CPU accesses data in big endian order, that is, the address ofa multi-byte word or quad points to the most significant byte. Figure 7 displays the Endianness of the ZNEO CPU.

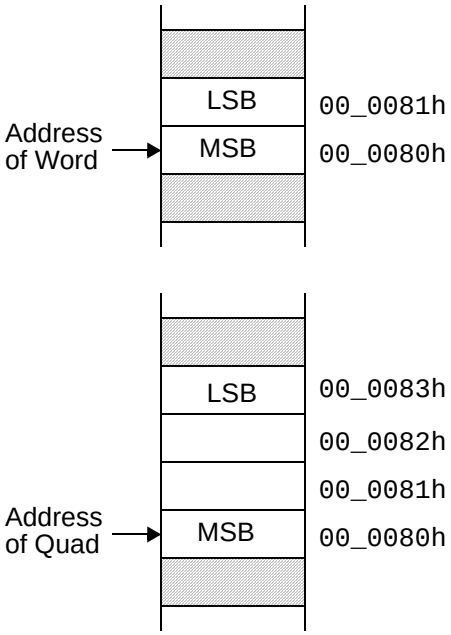


Figure 7. Endianness of Words and Quads

Bus Widths

The ZNEO CPU accesses 8-bit or 16-bit memories. The data buses of the internal nonvolatile memory and internal RAM are 16-bit wide. The internal peripherals are a mix of 8-bit and 16-bit peripherals. The external memory bus is configured as an 8-bit or 16-bit memory bus.

If a Word or Quad operation occurs on a 16-bit wide memory, the number of memory accesses depends on the alignment of the address. If the address is aligned on an even boundary, a Word operation takes one memory access and a Quad operation takes two memory accesses. If the address is on an odd boundary (unaligned), a Word operation takes two memory accesses and a Quad operation takes three memory accesses. Figure 8 displays the alignment Word and Quad operations on 16-bit memories.

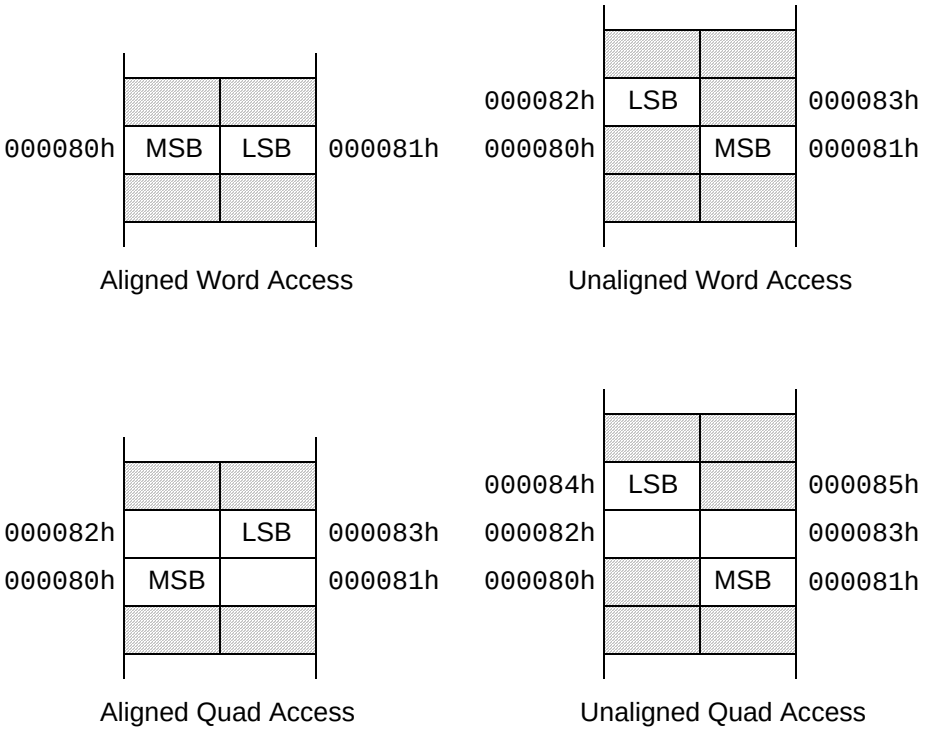


Figure 8. Alignment of Word and Quad Operations on 16-bit Memories

Peripheral Address Map

Table 6 provides the address map for the peripheral space of the ZNEO® Z16F Series of products. Not all devices and package styles in the ZNEO Z16F Series support all peripherals or all GPIO ports. Registers for unimplemented peripherals are considered as reserved.

Table 6. Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
ZNEO CPU Base Address = FF_E000				
FF_E004-FF_E007	Program Counter Overflow	PCOV	00FFFFFF	Refer to the ZNeo CPU User Manual
FF_E00C-FF_E00F	Stack Pointer Overflow	SPOV	00000000	
FF_E010	Flags	FLAGS	XX	
FF_E012	CPU Control	CPUCTL	00	
ZNEO Trace Address = FF_E014				
FF_E013	Trace Control	TRACECTL	00	325
FF_E014-FF_E017	Trace Address	TRACEADDR	XXXXXXXX	326
Interrupt Controller Base Address = FF_E020				
FF_E020	System Exception Status High	SYSEXCPH	0000	84
FF_E021	System Exception Status Low	SYSEXCPL	0000	84
FF_E022	Reserved	—	XX	—
FF_E023	Last IRQ Register	LASTIRQ	02	85
FF_E024-FF_E02F	Reserved	—	—	—
FF_E030	Interrupt Request 0	IRQ0	00	86
FF_E031	Interrupt Request 0 Set	IRQ0SET	XX	86
FF_E032	IRQ0 Enable High Bit	IRQ0ENH	00	90
FF_E033	IRQ0 Enable Low Bit	IRQ0ENL	00	90
FF_E034	Interrupt Request 1	IRQ1	00	87
FF_E035	Interrupt Request 1Set	IRQ1SET	XX	87
FF_E036	IRQ1 Enable High Bit	IRQ1ENH	00	91
FF_E037	IRQ1 Enable Low Bit	IRQ1ENL	00	91
FF_E038	Interrupt Request 2	IRQ2	00	88
FF_E039	Interrupt Request 2 Set	IRQ2SET	XX	89
FF_E03A	IRQ2 Enable High Bit	IRQ2ENH	00	92

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E03B	IRQ2 Enable Low Bit	IRQ2ENL	00	92
FF_E03C-FF_E03F	Reserved	—	XX	—
Watchdog Timer Base Address = FF_E040				
FF_E040-FF_E041	Reserved	—	—	—
FF_E042	Watchdog Timer Reload High Byte	WDTH	04	241
FF_E043	Watchdog Timer Reload Low Byte	WDTL	00	241
FF_E044-FF_E04F	Reserved	—	—	—
Reset Base Address = FF_E050				
FF_E050	Reset Status and Control Register	RSTSCR	XX	62
FF_E051-FF_E06F	Reserved	—	XX	—
Flash Controller Base Address = FF_E060				
FF_E060	Flash Command Register	FCMD	XX	261
FF_E060	Flash Status Register	FSTAT	00	262
FF_E061	Flash Control Register	FCTL	00	263
FF_E062	Flash Sector Protect Register	FSECT	00	264
FF_E063	Reserved	—	XX	—
FF_E064-FF_E065	Flash Page Select Register	FPAGE	0000	265
FF_E066-FF_E067	Flash Frequency Register	FFREQ	0000	266
External Interface Base Address = FF_E070				
FF_E070	External Interface Control	EXTCT		42
FF_E071	Reserved	—	—	—
FF_E072	Chip Select 0 Control High	EXTCS0H		43
FF_E073	Chip Select 0 Control Low	EXTCS0L		44
FF_E074	Chip Select 1 Control High	EXTCS1H		43
FF_E075	Chip Select 1 Control Low	EXTCS1L		45
FF_E076	Chip Select 2 Control High	EXTCS2H		43
FF_E077	Chip Select 2 Control Low	EXTCS2L		46
FF_E078	Chip Select 3 Control High	EXTCS3H		43
FF_E079	Chip Select 3 Control Low	EXTCS3L		46
FF_E07A	Chip Select 4 Control High	EXTCS4H		43

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E07B	Chip Select 4 Control Low	EXTCS4L		46
FF_E07C	Chip Select 5 Control High	EXTCS5H		43
FF_E07D	Chip Select 5 Control Low	EXTCS5L		46
FF_E07E-FF_E07F	Reserved	—	—	—
On Chip Debugger = FF_E080				
FF_E080	Debug Receive Data	DBGRXD	XX	315
FF_E081	Debug Transmit Data	DBGTXD	XX	315
FF_E082-FF_E083	Debug Baud Rate	DBGBR	XXXX	316
FF_E084	Debug Line Control	DBGLCR	XX	316
FF_E085	Debug Status	DBGSTAT	XX	318
FF_E086	Debug Control	DBGCTL	XX	319
Hardware Breakpoints = FF_E090				
FF_E090-FF_E093	Hardware Breakpoint 0	HWBP0	00000000	324
FF_E094-FF_E097	Hardware Breakpoint 1	HWBP1	00000000	324
FF_E098-FF_E09B	Hardware Breakpoint 2	HWBP2	00000000	324
FF_E09C-FF_E09F	Hardware Breakpoint 3	HWBP3	00000000	324
Oscillator Control Base Address = FF_E0A0				
FF_E0A0	Oscillator Control	OSCCTL	A0	334
FF_E0A1	Oscillator Divide	OSCDIV	00	335
GPIO Base Address = FF_E100				
GPIO Port A Base Address = FF_E100				
FF_E100	Port A Input Data	PAIN	XX	71
FF_E101	Port A Output Data	PAOUT	00	72
FF_E102	Port A Data Direction	PADD	00	73
FF_E103	Port A High Drive Enable	PAHDE	00	74
FF_E104	Port A Alternate Function High	PAAFH	00	75
FF_E105	Port A Alternate Function Low	PAAFL	00	75
FF_E106	Port A Output Control	PAOC	00	76
FF_E107	Port A Pull-Up Enable	PAPUE	00	76
FF_E108	Port A Stop Mode Recovery Enable	PASMRE	00	77
FF_E109-FF_E10B	Port A Reserved	—	—	—

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E10C	Port A IRQ MUX1	PAIMUX1	00	77
FF_E10D	Port A Reserved	—	—	—
FF_E10E	Port A IRQ MUX	PAIMUX	00	78
FF_E10F	Port A IRQ Edge	PAIEDGE	00	78
GPIO Port B Base Address = FF_E110				
FF_E110	Port B Input Data	PBIN	XX	71
FF_E111	Port B Output Data	PBOUT	00	72
FF_E112	Port B Data Direction	PBDD	00	73
FF_E113	Port B High Drive Enable	PBHDE	00	74
FF_E114	Reserved	—	—	—
FF_E115	Port B Alternate Function Low	PBAFL	00	75
FF_E116	Port B Output Control	PBOC	00	76
FF_E117	Port B Pull-Up Enable	PBPUE	00	76
FF_E118	Port B Stop Mode Recovery Enable	PBSMRE	00	77
FF_E119-FF_E11F	Port B Reserved	—	—	—
GPIO Port C Base Address = FF_E120				
FF_E120	Port C Input Data	PCIN	XX	71
FF_E121	Port C Output Data	PCOUT	00	72
FF_E122	Port C Data Direction	PCDD	00	73
FF_E123	Port C High Drive Enable	PCHDE	00	74
FF_E124	Port C Alternate Function High	PCAFH	00	75
FF_E125	Port C Alternate Function Low	PCAFL	00	76
FF_E126	Port C Output Control	PCOC	00	76
FF_E127	Port C Pull-Up Enable	PCPUE	00	76
FF_E128	Port C Stop Mode Recovery Enable	PCSMRE	00	77
FF_E129-FF_E12D	Port C Reserved	—	—	—
FF_E12E	Port C IRQ MUX	PCIMUX	00	79
FF_E12F	Port C Reserved	—	—	—
GPIO Port D Base Address = FF_E130				
FF_E130	Port D Input Data	PDIN	XX	71
FF_E131	Port D Output Data	PDOUT	00	72

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E132	Port D Data Direction	PDDD	00	73
FF_E133	Port D High Drive Enable	PDHDE	00	74
FF_E134	Port D Alternate Function High	PDAFH	00	75
FF_E135	Port D Alternate Function Low	PDAFL	00	76
FF_E136	Port D Output Control	PDOC	00	76
FF_E137	Port D Pull-Up Enable	PDPUE	00	76
FF_E138	Port D Stop Mode Recovery Enable	PDSMRE	00	77
FF_E139-FF_E13F	Port D Reserved	—	—	—
GPIO Port E Base Address = FF_E140				
FF_E140	Port E Input Data	PEIN	XX	71
FF_E141	Port E Output Data	PEOUT	00	72
FF_E142	Port E Data Direction	PEDD	00	73
FF_E143	Port E High Drive Enable	PEHDE	00	74
FF_E144	Reserved	—	—	—
FF_E145	Reserved	—	—	—
FF_E146	Port E Output Control	PEOC	00	76
FF_E147	Port E Pull-Up Enable	PEPUE	00	76
FF_E148	Port E Stop Mode Recovery Enable	PESMRE	00	77
FF_E149-FF_E14F	Port E Reserved	—	—	—
GPIO Port F Base Address = FF_E150				
FF_E150	Port F Input Data	PFIN	XX	71
FF_E151	Port F Output Data	PFOUT	00	72
FF_E152	Port F Data Direction	PFDD	00	73
FF_E153	Port F High Drive Enable	PFHDE	00	74
FF_E154	Reserved	—	—	—
FF_E155	Port F Alternate Function Low	PFAFL	00	76
FF_E156	Port F Output Control	PFOC	00	76
FF_E157	Port F Pull-Up Enable	PFPUE	00	76
FF_E158	Port F Stop Mode Recovery Enable	PFSMRE	00	77
FF_E159-FF_E15F	Port F Reserved	—	—	—

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
GPIO Port G Base Address = FF_E160				
FF_E160	Port G Input Data	PGIN	XX	71
FF_E161	Port G Output Data	PGOUT	00	72
FF_E162	Port G Data Direction	PGDD	00	73
FF_E163	Port G High Drive Enable	PGHDE	00	74
FF_E164	Reserved	—	—	—
FF_E165	Port G Alternate Function Low	PGAFL	00	76
FF_E166	Port G Output Control	PGOC	00	76
FF_E167	Port G Pull-Up Enable	PGPUE	00	76
FF_E168	Port G Stop Mode Recovery Enable	PGSMRE	00	77
FF_E169-FF_E16F	Port G Reserved	—	—	—
GPIO Port H Base Address = FF_E170				
FF_E170	Port H Input Data	PHIN	XX	71
FF_E171	Port H Output Data	PHOUT	00	72
FF_E172	Port H Data Direction	PHDD	00	73
FF_E173	Port H High Drive Enable	PHHDE	00	74
FF_E174	Port H Alternate Function High	PHAFH	00	75
FF_E175	Port H Alternate Function Low	PHAFL	00	76
FF_E176	Port H Output Control	PHOC	00	76
FF_E177	Port H Pull-Up Enable	PHPUE	00	76
FF_E178	Port H Stop Mode Recovery Enable	PHSMRE	00	77
FF_E179-FF_E17F	Port H Reserved	—	—	—
GPIO Port J Base Address = FF_E180				
FF_E180	Port J Input Data	PJIN	XX	71
FF_E181	Port J Output Data	PJOUT	00	72
FF_E182	Port J Data Direction	PJDD	00	73
FF_E183	Port J High Drive Enable	PJHDE	00	74
FF_E184	Reserved	—	—	—
FF_E185	Reserved	—	—	—
FF_E186	Port J Output Control	PJOC	00	76
FF_E187	Port J Pull-Up Enable	PJPUE	00	76

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E188	Port J Stop Mode Recovery Enable	PJSMRE	00	77
FF_E189-FF_E18F	Port J Reserved	—	—	—
GPIO Port K Base Address = FF_E190				
FF_E190	Port K Input Data	PKIN	XX	71
FF_E191	Port K Output Data	PKOUT	00	72
FF_E192	Port K Data Direction	PKDD	00	73
FF_E193	Port K High Drive Enable	PKHDE	00	74
FF_E194	Reserved	—	—	—
FF_E195	Port K Alternate Function Low	PKAFL	00	76
FF_E196	Port K Output Control	PKOC	00	76
FF_E197	Port K Pull-Up Enable	PKPUE	00	76
FF_E198	Port K Stop Mode Recovery Enable	PKSMRE	00	77
FF_E199-FF_E19F	Port K Reserved	—	—	—
Serial Channels Base Address = FF_E200				
LIN-UART 0 Base Address = FF_E200				
FF_E200	LIN-UART0 Transmit Data	U0TXD	XX	154
	LIN-UART0 Receive Data	U0RXD	XX	154
FF_E201	LIN-UART0 Status 0	U0STAT0	0000011Xb	155
FF_E202	LIN-UART0 Control 0	U0CTL0	00	160
FF_E203	LIN-UART0 Control 1	U0CTL1	00	164
FF_E204	LIN-UART0 Mode Select and Status	U0MDSTAT	00	162
FF_E205	LIN-UART0 Address Compare Register	U0ADDR	00	166
FF_E206	LIN-UART0 Baud Rate High Byte	U0BRH	FF	166
FF_E207	LIN-UART0 Baud Rate Low Byte	U0BRL	FF	167
FF_E208-FF_E20F	Reserved	—	XX	—
LIN-UART 1 Base Address = FF_E210				

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E210	LIN-UART1 Transmit Data	U1TXD	XX	154
	LIN-UART1 Receive Data	U1RXD	XX	154
FF_E211	LIN-UART1 Status 0	U1STAT0	0000011Xb	155
FF_E212	LIN-UART1 Control 0	U1CTL0	00	160
FF_E213	LIN-UART1 Control 1	U1CTL1	00	164
FF_E214	LIN-UART1 Mode Select and Status	U1MDSTAT	00	162
FF_E215	LIN-UART1 Address Compare Register	U1ADDR	00	166
FF_E216	LIN-UART1 Baud Rate High Byte	U1BRH	FF	166
FF_E217	LIN-UART1 Baud Rate Low Byte	U1BRL	FF	167
FF_E218-FF_E23F	Reserved	—	XX	—
I^2C Base Address = FF_E240				
FF_E240	I^2C Data	I2CDATA	00	227
FF_E241	I^2C Interrupt Status	I2CISTAT	80	228
FF_E242	I^2C Control	I2CCTL	00	229
FF_E243	I^2C Baud Rate High Byte	I2CBRH	FF	231
FF_E244	I^2C Baud Rate Low Byte	I2CBRL	FF	231
FF_E245	I^2C State	I2CSTATE	C0	232
FF_E246	I^2C Mode	I2CMODE	00	236
FF_E247	I^2C Slave Address	I2CSLVAD	00	237
FF_E248-FF_E25F	Reserved	—	XX	—
Enhanced Serial Peripheral Interface Base Address = FF_E260				
FF_E260	ESPI Data	ESPIDATA	XX	193
FF_E261	Reserved	—	XX	
FF_E262	ESPI Control	ESPICTL	00	194
FF_E263	ESPI Mode	ESPIMODE	00	197
FF_E264	ESPI Status	ESPISTAT	01	198
FF_E265	ESPI State	ESPISTATE	00	200
FF_E266	ESPI Baud Rate High Byte	ESPIBRH	FF	202
FF_E267	ESPI Baud Rate Low Byte	ESPIBRL	FF	202

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
Timers: Base Address = FFF_E300				
Timer 0 (General-Purpose Timer) Base Address = FF_E300				
FF_E300	Timer 0 High Byte	T0H	00	107
FF_E301	Timer 0 Low Byte	T0L	01	107
FF_E302	Timer 0 Reload High Byte	T0RH	FF	108
FF_E303	Timer 0 Reload Low Byte	T0RL	FF	108
FF_E304	Timer 0 PWM High Byte	T0PWMH	00	109
FF_E305	Timer 0 PWM Low Byte	T0PWML	00	109
FF_E306	Timer 0 Control 0	T0CTL0	00	110
FF_E307	Timer 0 Control 1	T0CTL1	00	111
Timer 1 (General-Purpose Timer) Base Address = FF_E310				
FF_E310	Timer 1 High Byte	T1H	00	107
FF_E311	Timer 1 Low Byte	T1L	01	107
FF_E312	Timer 1 Reload High Byte	T1RH	FF	108
FF_E313	Timer 1 Reload Low Byte	T1RL	FF	108
FF_E314	Timer 1 PWM High Byte	T1PWMH	00	109
FF_E315	Timer 1 PWM Low Byte	T1PWML	00	109
FF_E316	Timer 1 Control 0	T1CTL0	00	110
FF_E317	Timer 1 Control 1	T1CTL1	00	111
Timer 2 (General-Purpose Timer) Base Address = FF_E320				
FF_E320	Timer 2 High Byte	T2H	00	107
FF_E321	Timer 2 Low Byte	T2L	01	107
FF_E322	Timer 2 Reload High Byte	T2RH	FF	108
FF_E323	Timer 2 Reload Low Byte	T2RL	FF	108
FF_E324	Timer 2 PWM High Byte	T2PWMH	00	109
FF_E325	Timer 2 PWM Low Byte	T2PWML	00	109
FF_E326	Timer 2 Control 0	T2CTL0	00	110
FF_E327	Timer 2 Control 1	T2CTL1	00	111
Pulse Width Modulator (PWM) Base Address = FF_E380				
FF_E380	PWM Control 0	PWMCTL0	00	125
FF_E381	PWM Control 1	PWMCTL1	00	126
FF_E382	PWM Deadband	PWMDB	00	127

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E383	PWM Minimum Pulse Width Filter	PWMMPF	00	128
FF_E384	PWM Fault Mask	PWMFM	00	129
FF_E385	PWM Fault Status	PWMFSTAT	00	130
FF_E386	PWM Input Sample Register	PWMIN	00	132
FF_E387	PWM Output Control	PWMOUT	00	133
FF_E388	PWM Fault Control	PWMFCTL	00	131
FF_E389	Reserved	—	—	—
FF_E38A	Current-Sense Sample and Hold Control 0	CSSHR0	00	134
FF_E38B	Current-Sense Sample and Hold Control 1	CSSHR1	00	134
FF_E38C-FF_E38B	Reserved	—	—	—
FF_E38C	PWM High Byte	PWMH	XX	122
FF_E38D	PWM Low Byte	PWML	XX	122
FF_E38E	PWM Reload High Byte	PWMRH	FF	123
FF_E38F	PWM Reload Low Byte	PWMRL	FF	123
FF_E390	PWM 0 High Side Duty Cycle High Byte	PWMH0DH	00	124
FF_E391	PWM 0 High Side Duty Cycle Low Byte	PWMH0DL	00	125
FF_E392	PWM 0 Low Side Duty Cycle High Byte	PWML0DH	00	124
FF_E393	PWM 0 Low Side Duty Cycle Low Byte	PWML0DL	00	125
FF_E394	PWM 1 High Side Duty Cycle High Byte	PWMH1DH	00	124
FF_E395	PWM 1 High Side Duty Cycle Low Byte	PWMH1DL	00	125
FF_E396	PWM 1 Low Side Duty Cycle High Byte	PWML1DH	00	124
FF_E397	PWM 1 Low Side Duty Cycle Low Byte	PWML1DL	00	125
FF_E398	PWM 2 High Side Duty Cycle High Byte	PWMH2DH	00	124

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E399	PWM 2 High Side Duty Cycle Low Byte	PWMH2DL	00	125
FF_E39A	PWM 2 Low Side Duty Cycle High Byte	PWML2DH	00	124
FF_E39B	PWM 2 Low Side Duty Cycle Low Byte	PWML2DL	00	125
FF_E39C-FF_E3BF	Reserved for PWM	—	—	—
DMA Block Base Address = FF_E400				
DMA Request Selection Control				
FF_E400	DMA0 Request Select	DMA0REQSEL	00	282
FF_E401	DMA1 Request Select	DMA1REQSEL	00	282
FF_E402	DMA2 Request Select	DMA2REQSEL	00	282
FF_E403	DMA3 Request Select	DMA3REQSEL	00	282
FF_E404-F	Reserved	—	—	—
DMA Channel 0 Base Address = FF_E410				
FF_E410	DMA0 Control 0	DMA0CTL0	00	285
FF_E411	DMA0 Control 1	DMA0CTL1	00	285
FF_E412	DMA0 Transfer Length High	DMA0TXLNH	00	286
FF_E413	DMA0 Transfer Length Low	DMA0TXLNL	00	287
FF_E414	Reserved	—	—	—
FF_E415	DMA0 Destination Address Upper	DMA0DARU	00	287
FF_E416	DMA0 Destination Address High	DMA0DARH	00	287
FF_E417	DMA0 Destination Address Low	DMA0DARL	00	287
FF_E418	Reserved	—	—	—
FF_E419	DMA0 Source Address Upper	DMA0SARU	00	288
FF_E41A	DMA0 Source Address High	DMA0SARH	00	288
FF_E41B	DMA0 Source Address Low	DMA0SARL	00	288
FF_E41C	Reserved	—	—	—
FF_E41D	DMA0 List Address Upper	DMA0LARU	00	289
FF_E41E	DMA0 List Address High	DMA0LARH	00	289
FF_E41F	DMA0 List Address Low	DMA0LARL	00	289

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
DMA Channel 1 Base Address = FF_E420				
FF_E420	DMA1 Control 0	DMA1CTL0	00	285
FF_E421	DMA1 Control 1	DMA1CTL1	00	285
FF_E422	DMA1 Transfer Length High	DMA1TXLNH	00	286
FF_E423	DMA1 Transfer Length Low	DMA1TXLNL	00	287
FF_E424	Reserved	—	—	—
FF_E425	DMA1 Destination Address Upper	DMA1DARU	00	287
FF_E426	DMA1 Destination Address High	DMA1DARH	00	287
FF_E427	DMA1 Destination Address Low	DMA1DARL	00	287
FF_E428	Reserved	—	—	—
FF_E429	DMA1 Source Address Upper	DMA1SARU	00	288
FF_E42A	DMA1 Source Address High	DMA1SARH	00	288
FF_E42B	DMA1 Source Address Low	DMA1SARL	00	288
FF_E42C	Reserved	—	—	—
FF_E42D	DMA1 List Address Upper	DMA1LARU	00	289
FF_E42E	DMA1 List Address High	DMA1LARH	00	289
FF_E42F	DMA1 List Address Low	DMA1LARL	00	289
DMA Channel 2 Base Address = FF_E430				
FF_E430	DMA2 Control 0	DMA2CTL0	00	285
FF_E431	DMA2 Control 1	DMA2CTL1	00	285
FF_E432	DMA2 Transfer Length High	DMA2TXLNH	00	286
FF_E433	DMA2 Transfer Length Low	DMA2TXLNL	00	287
FF_E434	Reserved	—	—	—
FF_E435	DMA2 Destination Address Upper	DMA2DARU	00	287
FF_E436	DMA2 Destination Address High	DMA2DARH	00	287
FF_E437	DMA2 Destination Address Low	DMA2DARL	00	287
FF_E438	Reserved	—	—	—
FF_E439	DMA2 Source Address Upper	DMA2SARU	00	288

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E43A	DMA2 Source Address High	DMA2SARH	00	288
FF_E43B	DMA2 Source Address Low	DMA2SARL	00	288
FF_E43C	Reserved			
FF_E43D	DMA2 List Address Upper	DMA2LARU	00	289
FF_E43E	DMA2 List Address High	DMA2LARH	00	289
FF_E43F	DMA2 List Address Low	DMA2LARL	00	289
DMA Channel 3 Base Address = FF_E440				
FF_E440	DMA3 Control 0	DMA3CTL0	00	285
FF_E441	DMA3 Control 1	DMA3CTL1	00	285
FF_E442	DMA3 Transfer Length High	DMA3TXLNH	00	286
FF_E443	DMA3 Transfer Length Low	DMA3TXLNL	00	287
FF_E444	Reserved	—	—	—
FF_E445	DMA3 Destination Address Upper	DMA3DARU	00	287
FF_E446	DMA3 Destination Address High	DMA3DARH	00	287
FF_E447	DMA3 Destination Address Low	DMA3DARL	00	287
FF_E448	Reserved	—	—	—
FF_E449	DMA3 Source Address Upper	DMA3SARU	00	288
FF_E44A	DMA3 Source Address High	DMA3SARH	00	288
FF_E44B	DMA3 Source Address Low	DMA3SARL	00	288
FF_E44C	Reserved	—	—	—
FF_E44D	DMA3 List Address Upper	DMA3LARU	00	289
FF_E44E	DMA3 List Address High	DMA3LARH	00	289
FF_E44F	DMA3 List Address Low	DMA3LARL	00	289
Analog Block Base Address = FF_E500				
ADC Base Address = FF_E500				
FF_E500	ADC0 Control Register	ADC0CTL	00	246
FF_E501	Reserved	—	—	—
FF_E502	ADC0 Data High Byte Register	ADC0D_H	XX	247
FF_E503	ADC0 Data Low Bits Register	ADC0D_L	XX	248

XX = Undefined.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E504	ADC Sample and Settling Time Register	ADCSST	0F	248
FF_E505	ADC Sample Hold Time	ADCST	3F	249
FF_E506	ADC Clock Prescale Register	ADCCP	00	249
FF_E507	ADC0 MAX Register	ADC0MAX	00	250
FF_E508-FF_E50F	Reserved	—	—	—
FF_E510	Comparator and Op-Amp Control	CMPOPC	00	253
FF_E511	Reserved	—	—	—
FF_E512	ADC Sample Timer Capture High	ADCTCAPH	XX	251
FF_E513	ADC Sample Timer Capture Low	ADCTCAPL	XX	251
Option Trim Registers Base Address = FF_FF00				
FF_FF00-FF_FF24	Reserved for internal Zilog® use	—	—	—
FF_FF25	IPO Trim 1	IPOTRIM1	XX	296
FF_FF26	IPO Trim 2	IPOTRIM2	XX	296
FF_FF27	ADC Reference Voltage Trim	ADCTRIM	XX	297
XX = Undefined.				

External Interface

The external interface allows seamless connection to external memory and/or peripherals. The configurable nature of the external interface supports connection with many different bus styles and signal formats. Bit-wise control of the address, data and control signals means no wasted GPIO pins. Other features of the external interface includes:

- Hardware bus controller with programmable signal polarity for chip selects
- Programmable wait state generator
- Selectable address and data bus widths
- Six external chip selects
- ISA-compatible mode
- External program execution and Stack operations
- External $\overline{\text{WAIT}}$ pin for slow peripherals

External Interface Signals

Table 7 lists the external interface signals. The external interface consists of a 24-bit address bus, an 8-bit/16-bit bidirectional data bus and control signals (Read, Write, Chip Selects and Wait). It is not necessary to use all pins for proper operation of the external interface. The external interface signals are enabled pin-by-pin using the GPIO alternate functions. For more information about GPIO alternate functions, see the General-Purpose Input/Output section on page 66.

Table 7. External Interface Signals Description

External Interface Signal	Direction
DATA[7:0]	Input/Output
DATA[15:8]	Input/Output
ADDR[7:0]	Output
ADDR[15:8]	Output
ADDR[23:16]	Output
WR	Output
RD	Output
$\overline{\text{CS0}}$	Output
WAIT	Input

Table 7. External Interface Signals Description (Continued)

External Interface Signal	Direction
$\overline{CS1}$	Output
$\overline{CS2}$	Output
$\overline{CS3}$	Output
$\overline{CS4}$	Output
$\overline{CS5}$	Output

Chip Selects

The chip selects support connection of multiple memories and peripherals to the external interface. Figure 9 displays the memory map of the chip selects. The chip select boundaries are at fixed addresses. On-chip memory always have priority over external memory. Chip select 0 has the lowest priority and chip select 5 has the highest priority.

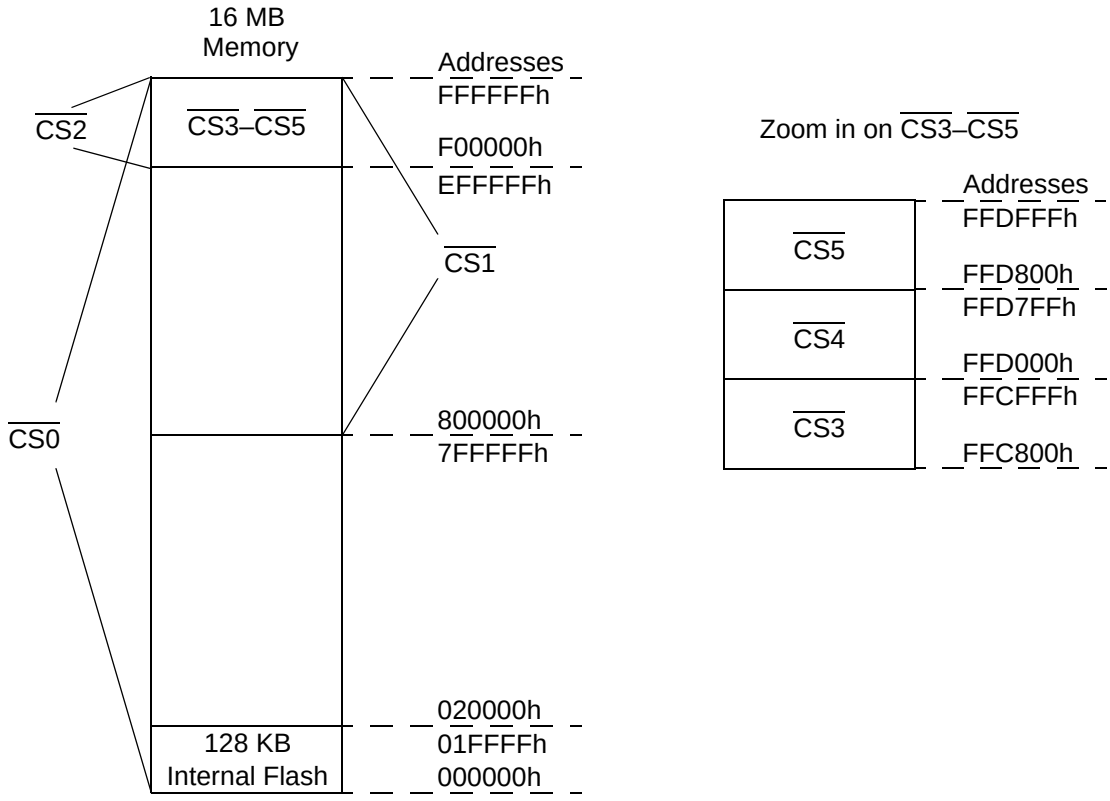


Figure 9. Chip Select Boundary Addressing with 128 KB Internal Flash

Table 8. Example Usage of Chip Selects

Chip Select	Memory/Peripheral	Lower Address	Upper Address	Typical Uses
	64 KB Internal Flash	000000h	00FFFFh	Program code, look-up tables and interrupt vectors.
$\overline{CS0}$	8 MB External ROM or Flash	010000h	7FFFFFh	Program code and look-up tables. Lowest 64 KB of the 8 MB is inaccessible as the on-chip Flash has higher priority for addresses 000000h through 00FFFFh.
$\overline{CS1}$	128 KB External RAM	080000h	09FFFFh	Stack and data is placed anywhere in $\overline{CS1}$ address space except where higher priority $\overline{CS2}$ or $\overline{CS3}$ overlaps.
		0AFFFFh	0EFFFFh	Unused addresses as no external memory placed in these locations.
$\overline{CS2}$	External Ethernet MAC	F00000h	F3FFFFh	Ethernet MAC is an example of an external communication peripheral that is connected to the external interface.
$\overline{CS3}$	External CAN controller.	FFC800h	FFCFFFh	CAN controller is another example of an external communication peripheral that is connected to the external interface.
		F80000h	FFFFFFh	Unused addresses as no external memory or peripherals placed in these locations.

Tools Compatibility Guidelines

The external interface offers the designer the flexibility to place external devices in almost any range of the 24-bit address space. The primary hardware consideration is that more chip selects are available in high memory and devices needing more than 15 wait states must use one of chip selects $\overline{CS}$[2–5]. After a design is completed, it is necessary to develop software for it. The task of designing application software is much easier if the hardware designer considers the following guidelines:

- The microcontroller's internal Flash memory must be enabled during software development. This simplifies downloading of program code and allows the ZDSII default program configuration to be used. If the internal Flash is disabled, the address space beginning at 00_0000h must address external Flash or other memory containing the necessary option bits, vectors and application startup code.
- Any external nonvolatile memory must be located above the internal Flash in the address space, but below any volatile (random access) memory. There is a gap or hole in

the address space between internal and external nonvolatile memory and between nonvolatile and volatile memory.

- Any external volatile (random access) memory must be located at or above 80_0000h in the 24-bit address space (in the $\overline{CS1}$ range). This location is a requirement of the ZDSII GUI. Volatile memory on $\overline{CS0}$ is located in a lower address range if it is configured by adding an edited linker RANGE command to the Additional Linker Commands field of the ZDSII project settings.
- External volatile memory falling below FF_8000h must be addressed as a contiguous block. The ZDSII C-Compiler large model does not support holes in 32-bit addressed volatile memory. There is a hole between this memory and volatile memory at or above FF_8000h, however.
- External volatile memory at or above FF_8000h must be addressed as a block contiguous with the microcontroller's internal RAM. The ZDSII C-Compiler small model does not support holes in 16-bit addressable volatile memory.
- External volatile memory must not be located above internal RAM, which ends at FF_BFFFh. This location is a requirement of the ZDSII GUI. Volatile memory is located at FF_C000h and extend up to FF_DFFFh if the space is not used for I/O, but the range must be configured by adding an edited linker RANGE command to the Additional Linker Commands field of the project settings. The debugger memory window always displays this range as part of the I/O Data space, however.
- The ZDSII GUI assumes external I/O is located in the range FF_C000h to FF_DFFFh. Any external I/O that is located elsewhere is accessed using absolute addressing. The debugger memory window displays all addresses below FF_C000h as part of the Memory space.

For details about how ZDSII development tools use memory, refer to the Zilog Developer Studio II – ZNEO User Manual (UM0171).

External $\overline{WAIT}$ Pin Operation

Setup of the external $\overline{WAIT}$ pin is selected by the GPIO alternate function. When using the external $\overline{WAIT}$ pin, at least one internal wait state must be added to allow sufficient address valid to wait input setup time.

Operation

Wait State Generator

Programmable wait states are inserted to provide external devices with additional clock cycles to complete their read and write operations. The number of wait states are controlled by the CSxWAIT[3:0] field and the PRxWAIT[1:0] field as shown in the the Chip Select Control Registers section on page 43. The wait states idle the ZNEO CPU for the specified number of system clock cycles; a maximum of 31 wait states are inserted. An example of wait state operation is illustrated in Figure 10. In this example, the external interface has been configured to provide two wait states. See the detailed timing diagrams in the External Interface Timing section on page 47.

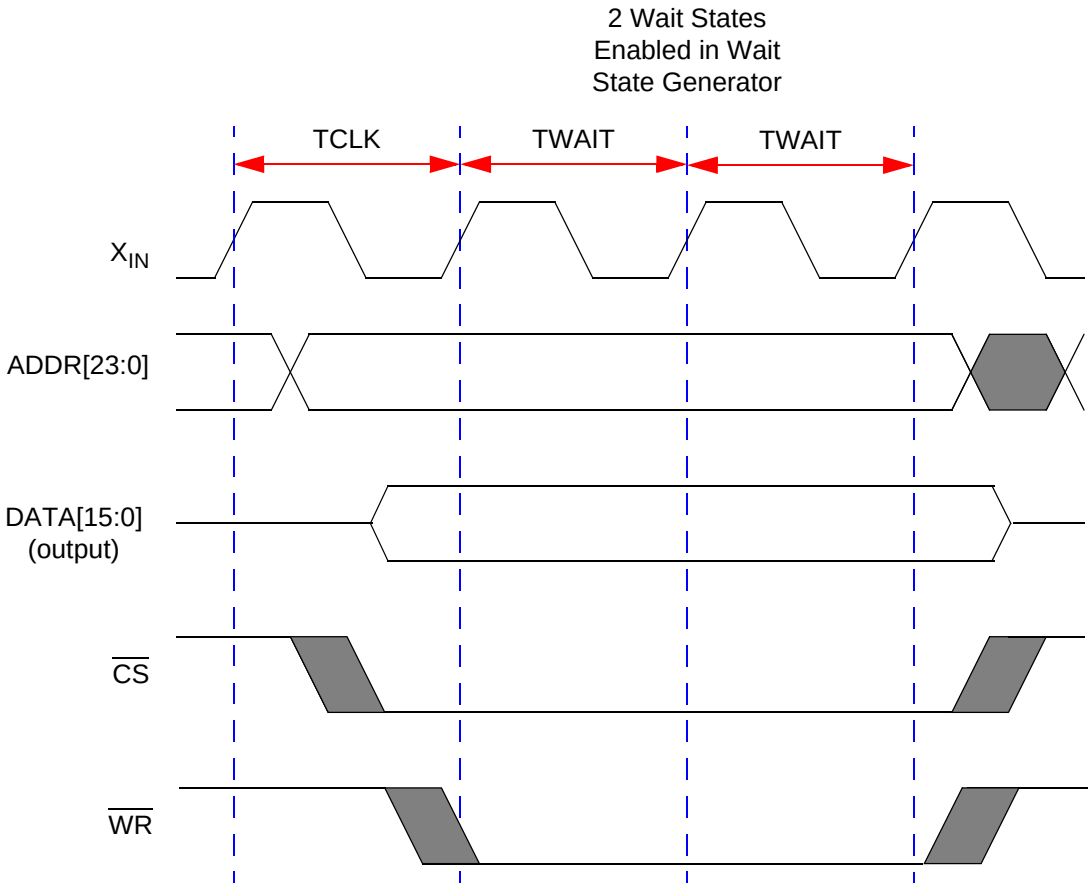


Figure 10. External Interface Wait State Operation Example (Write Operation)

ISA-Compatible Mode

Configuring the external interface for ISA Mode adjusts the Read timing to follow the ISA Mode commonly employed in PC and related applications. In ISA Mode, assertion of the Read signal ($\overline{RD}$) is delayed one-half system clock. Also, an extra wait state is added during read operations.

External Interface Control Register Definitions

The following section describes the various control registers.

External Interface Control Register

The External Interface Control Register, shown in Table 9, enables the interface and sets the internal memory size.

Table 9. External Interface Control Register (EXTCT)

Bits	7	6	5	4	3	2	1	0
Field	BUSSEL		MEMSIZE		RESERVED			
RESET	00		00		0000			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E070h							

Bit	Description
[7:6] BUSSEL	**Bus Select External Interface Enable** 00 = No External Bus. 01 = 8-bit External Bus Interface is enabled (Port E). 1X = 16-bit External Bus Interface is enabled (Port J and Port E).
[5:4] MEMSIZE	**Select Internal Memory Size** 00 = 128 KB of internal memory. 01 = 64 KB of internal memory. 10 = 32 KB of internal memory. 11 = No Internal Memory.
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

Chip Select Control Registers

The Chip Select Control registers control the chip select outputs. Each chip select has a High byte and Low byte.

Table 10. External Chip Select Control Registers High (EXTCSxH)

Bits	7	6	5	4	3	2	1	0
Field	CSxEN	POLSEL	CSxISA	W/B	RESERVED			
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_(E072, E074, E076, E078, E07A, E07C)H							

Bit	Description
[7] CSEN	**Chip Select Enable** 0 = CSx is disabled. 1 = CSx is enabled.
[6] POLSEx	**Polarity Select** 0 = CSx is active Low. 1 = CSx is active High.
[5] CSxISA	**Chip Select ISA Mode Enable** 0 = ISA Mode disabled. 1 = ISA Mode enabled.
[4] W/B	**Word or Byte Mode Select per Chip Select for 16-Bit or 8-Bit Peripherals** 0 = External interface uses Data[15:0] for this chip select. 1 = External interface uses Data[7:0] for this chip select.
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

Table 11 lists the External Chip Select Control Registers Low for $\overline{CS0}$ (EXTCS0L). This register sets the number of wait states for chip select 0. Waits are only added if the chip select is enabled. Chip Select 0 is enabled automatically in ROMLESS Mode.

Table 11. External Chip Select Control Registers Low for $\overline{CS0}$ (EXTCS0L)

Bits	7	6	5	4	3	2	1	0
Field	RESERVED		PR0WAIT		CS0WAIT			
RESET	0	0	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_(E073)H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:4] PR0WAIT[2:0]	**Post Read Wait Selection** 00 = 0 wait state. 01 = 1 wait state. 10 = 2 wait states. 11 = 3 wait states.
[3:0] CS0WAIT	**Chip Select 0 Wait Selection** 0000 = 0 wait state. 0001 = 1 wait state. 0010 = 2 wait states. 0011 = 3 wait states. 0100 = 4 wait states. 0101 = 5 wait states. 0110 = 6 wait states. 0111 = 7 wait states. 1000 = 8 wait states. 1001 = 9 wait states. 1010 = 10 wait state.s 1011 = 11 wait states. 1100 = 12 wait states. 1101 = 13 wait states. 1110 = 14 wait states. 1111 = 15 wait states.

Table 12 shows the External Chip Select Control Registers Low for $\overline{CS1}$(EXTSC1L). This register sets the number of wait states for Chip Select 1. Waits are only added if the chip select is enabled.

Table 12. External Chip Select Control Registers Low for $\overline{CS1}$ (EXTCS1L)

Bits	7	6	5	4	3	2	1	0
Field	RESERVED		PR1WAIT		CS1WAIT			
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_(E075)H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:4] PR1WAIT[2:0]	**Post Read Wait Selection** 00 = 0 wait state. 01 = 1 wait state. 10 = 2 wait states. 11 = 3 wait states.
[3:0] CS1WAIT	**Chip Select 1 Wait Selection** 0000 = 0 wait state. 0001 = 1 wait state. 0010 = 2 wait states. 0011 = 3 wait states. 0100 = 4 wait states. 0101 = 5 wait states. 0110 = 6 wait states. 0111 = 7 wait states. 1000 = 8 wait states. 1001 = 9 wait states. 1010 = 10 wait states. 1011 = 11 wait states. 1100 = 12 wait states. 1101 = 13 wait states. 1110 = 14 wait states. 1111 = 15 wait states.

Table 13 lists the External Chip Select Control Registers Low for $\overline{CS2}$ to $\overline{CS5}$ (EXTCSxL). This register sets the number of wait states for chip selects 2 through 5. Waits are only added if the chip select is enabled.

Table 13. External Chip Select Control Registers Low for $\overline{CS2}$ to $\overline{CS5}$ (EXTCSxL)

Bits	7	6	5	4	3	2	1	0
Field	RESERVED		PRxWAIT		CSxWAIT			
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_(E077, E079, E07B, E07D)H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:4] PRxWAIT[2:0]	**Post Read Wait Selection** 00 = 0 wait state. 01 = 1 wait state. 10 = 2 wait states. 11 = 3 wait states.
[3:0] CSxWAIT	**Chip Select x Wait Selection** 0000 = 0 wait state. 0001 = 2 wait state. 0010 = 4 wait states. 0011 = 6 wait states. 0100 = 8 wait states. 0101 = 10 wait states. 0110 = 12 wait states. 0111 = 14 wait states. 1000 = 16 wait states. 1001 = 18 wait states. 1010 = 20 wait states. 1011 = 22 wait states. 1100 = 24 wait states. 1101 = 26 wait states. 1110 = 28 wait states. 1111 = 30 wait states.

External Interface Timing

The following sections describe the ZNEO Z16F Series MCU's external interface timing.

External Interface Write Timing, Normal Mode

Figure 11 and Table 14 provide timing information for the external interface performing a write operation. In Figure 11, it is assumed that the wait state generator is configured to provide 1 wait state during write operations. The external $\overline{\text{WAIT}}$ input pin is generating an additional Wait period. It is assumed in Figure 11 that the chip select ($\overline{\text{CS}}$) signal has been configured for active Low operation. Though the internal system clock is not provided as an external signal, it provides a useful reference for control signal events.

Note: At the completion of a Write cycle, the deassertion of the $\overline{\text{WR}}$ signal is fed back from the pin and used on chip to control the deassertion of the data, $\overline{\text{CS}}$, address and byte enable signals to assure proper timing of the data hold.

Table 14. External Interface Timing for a Write Operation, Normal Mode

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	SYS CLK Rise to Address Valid Delay		10
T_2	$\overline{\text{WR}}$ Rise to Address Output Hold Time	3	
T_3	SYS CLK Rise to Data Valid Delay		10
T_4	$\overline{\text{WR}}$ Rise to Data Output Hold Time	3	
T_5	SYS CLK Rise to $\overline{\text{CS}}$ Assertion Delay		10
T_6	$\overline{\text{WR}}$ Rise to $\overline{\text{CS}}$ Deassertion Hold Time	3	
T_7	SYS CLK Rise to $\overline{\text{WR}}$ Assertion Delay		$1/2T_{CLK}$ +10
T_8	SYS CLK Rise to $\overline{\text{WR}}$ Deassertion Hold Time	3	
T_9	$\overline{\text{WAIT}}$ Input Pin Assertion to X_{IN} Rise Setup Time	1	
T_{10}	$\overline{\text{WAIT}}$ Input Pin Deassertion to X_{IN} Rise Setup Time	1	
T_{11}	SYS CLK Rise to $\overline{\text{DMAACK}}$ Assertion Delay		10
T_{12}	SYS CLK Rise to $\overline{\text{DMAACK}}$ Deassertion Hold Time	3	
T_{13}	SYS CLK Rise to $\overline{\text{BHEN}}$ or $\overline{\text{BLEN}}$ Assertion Delay		10
T_{14}	$\overline{\text{WR}}$ Rise to $\overline{\text{BHEN}}$ or $\overline{\text{BLEN}}$ Deassertion Hold Time	3	

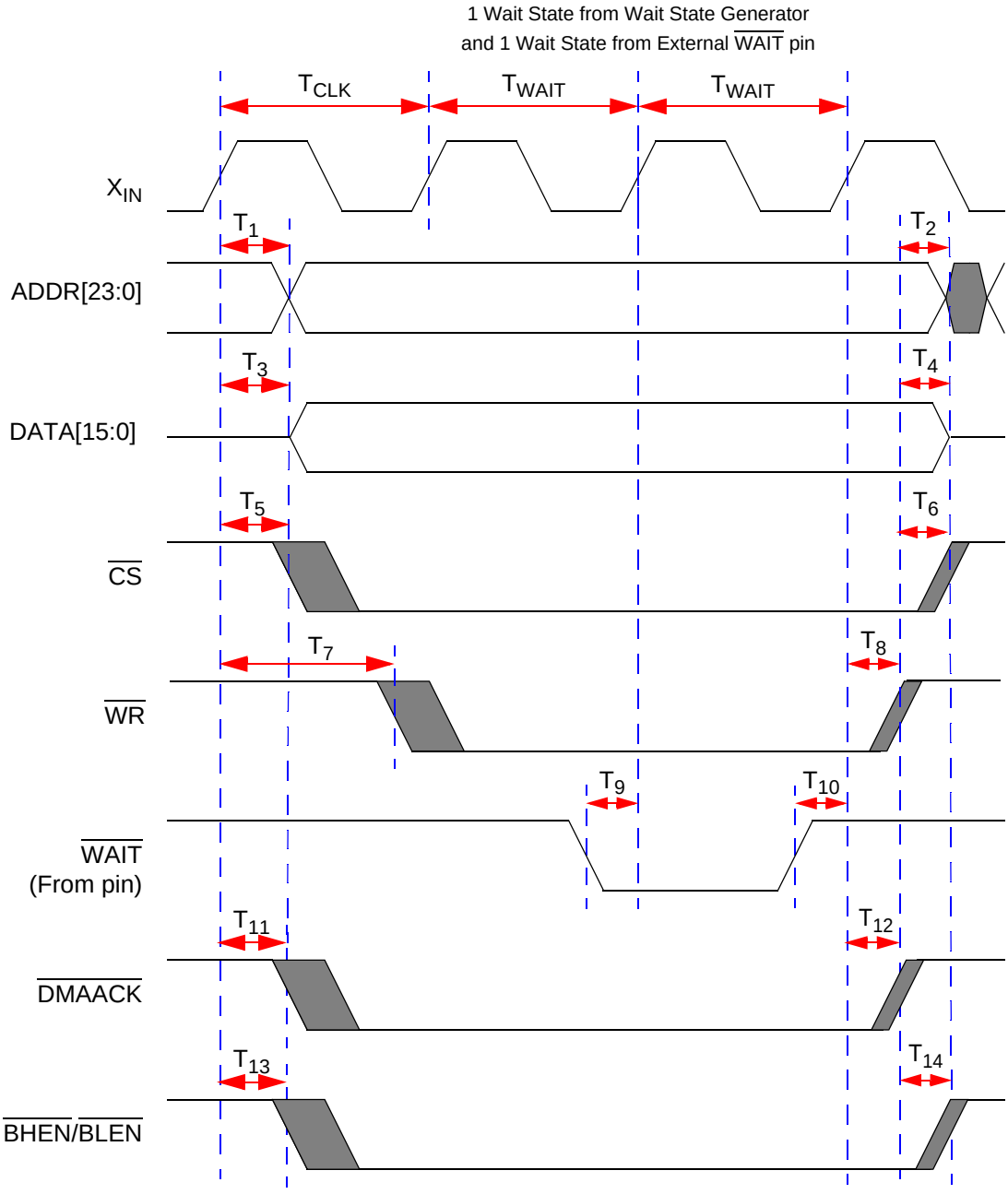


Figure 11. External Interface Timing for a Write Operation, Normal Mode

External Interface Write Timing, ISA Mode

Figure 12 and Table 15 provide timing information for the external interface performing a write operation. In Figure 12, it is assumed that the wait state generator has been configured to provide 1 wait state during write operations. The external $\overline{WAIT}$ input pin is generating an additional Wait period. As with Normal Mode, the $\overline{WR}$ signal is fed back from the pin and used on chip to time the removal of the data signals to ensure proper timing of the data hold.

Table 15. External Interface Timing for a Write Operation, ISA Mode

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	X_{IN} Rise to Address Valid Delay		10
T_2	X_{IN} Rise to Address Output Hold Time	3	
T_3	X_{IN} Rise to Data Valid Delay		10
T_4	$\overline{WR}$ Rise to Data Output Hold Time	3	
T_5	X_{IN} Rise to $\overline{CS}$ Assertion Delay		10
T_6	X_{IN} Rise to $\overline{CS}$ Deassertion Hold Time	3	
T_7	X_{IN} Fall to $\overline{WR}$ Assertion Delay		10
T_8	X_{IN} Fall to $\overline{WR}$ Deassertion Hold Time	3	
T_9	$\overline{WAIT}$ Input Pin Assertion to X_{IN} Rise Setup Time	1	
T_{10}	$\overline{WAIT}$ Input Pin Deassertion to X_{IN} Rise Setup Time	1	
T_{11}	X_{IN} Rise to $\overline{DMAACK}$ Assertion Delay		10
T_{12}	X_{IN} Rise to $\overline{DMAACK}$ Deassertion Hold Time	3	
T_{13}	X_{IN} Rise to $\overline{BHEN}$ or $\overline{BLEN}$ Assertion Delay		10
T_{14}	X_{IN} Rise to $\overline{BHEN}$ or $\overline{BLEN}$ Deassertion Hold Time	3	

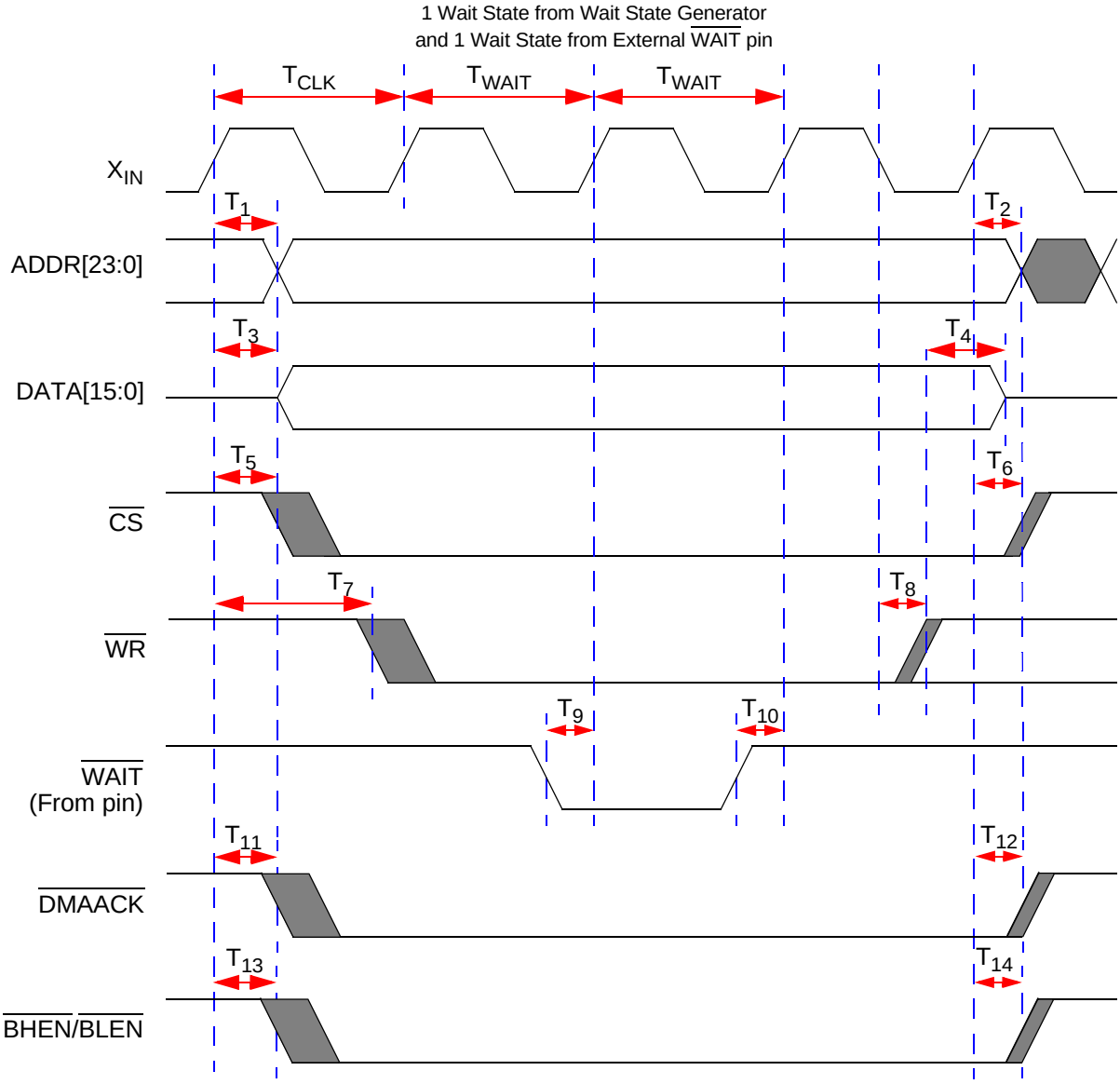


Figure 12. External Interface Timing for a Write Operation, ISA Mode

External Interface Read Timing, Normal Mode

Figure 13 and Table 16 provide timing information for the external interface performing a read operation in Normal Mode. In Figure 13, it is assumed the wait state generator has been configured to provide 2 wait states during read operations. For proper data hold time

determination, be aware that the input data is captured on chip during the rising edge of the system clock prior to the RD signal deassertion. The Read signal ($\overline{RD}$) timing is shown for both NORMAL and ISA modes.

Table 16. External Interface Timing for a Read Operation, Normal Mode

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	X_{IN} Rise to Address Valid Delay		10
T_2	X_{IN} Rise to Address Output Hold Time	3	
T_3	Data Input Valid to X_{IN} Rise Setup Time		3
T_4	$\overline{RD}$ Rise to Data Input Hold Time	0	
T_5	X_{IN} Rise to $\overline{CS}$ Assertion Delay		10
T_6	X_{IN} Rise to $\overline{CS}$ Deassertion Hold Time	3	
T_7	X_{IN} Rise to $\overline{RD}$ Assertion Delay		10
T_8	X_{IN} Rise to $\overline{RD}$ Deassertion Hold Time	3	
T_9	$\overline{WAIT}$ Input Pin Assertion to X_{IN} Rise Setup Time	1	
T_{10}	$\overline{WAIT}$ Input Pin Deassertion to X_{IN} Rise Setup Time	1	
T_{11}	X_{IN} Rise to $\overline{DMAACK}$ Assertion Delay		10
T_{12}	X_{IN} Rise to $\overline{DMAACK}$ Deassertion Hold Time	3	
T_{13}	X_{IN} Rise to $\overline{BHEN}$ or $\overline{BLEN}$ Assertion Delay		10
T_{14}	X_{IN} Rise to $\overline{BHEN}$ or $\overline{BLEN}$ Deassertion Hold Time	3	

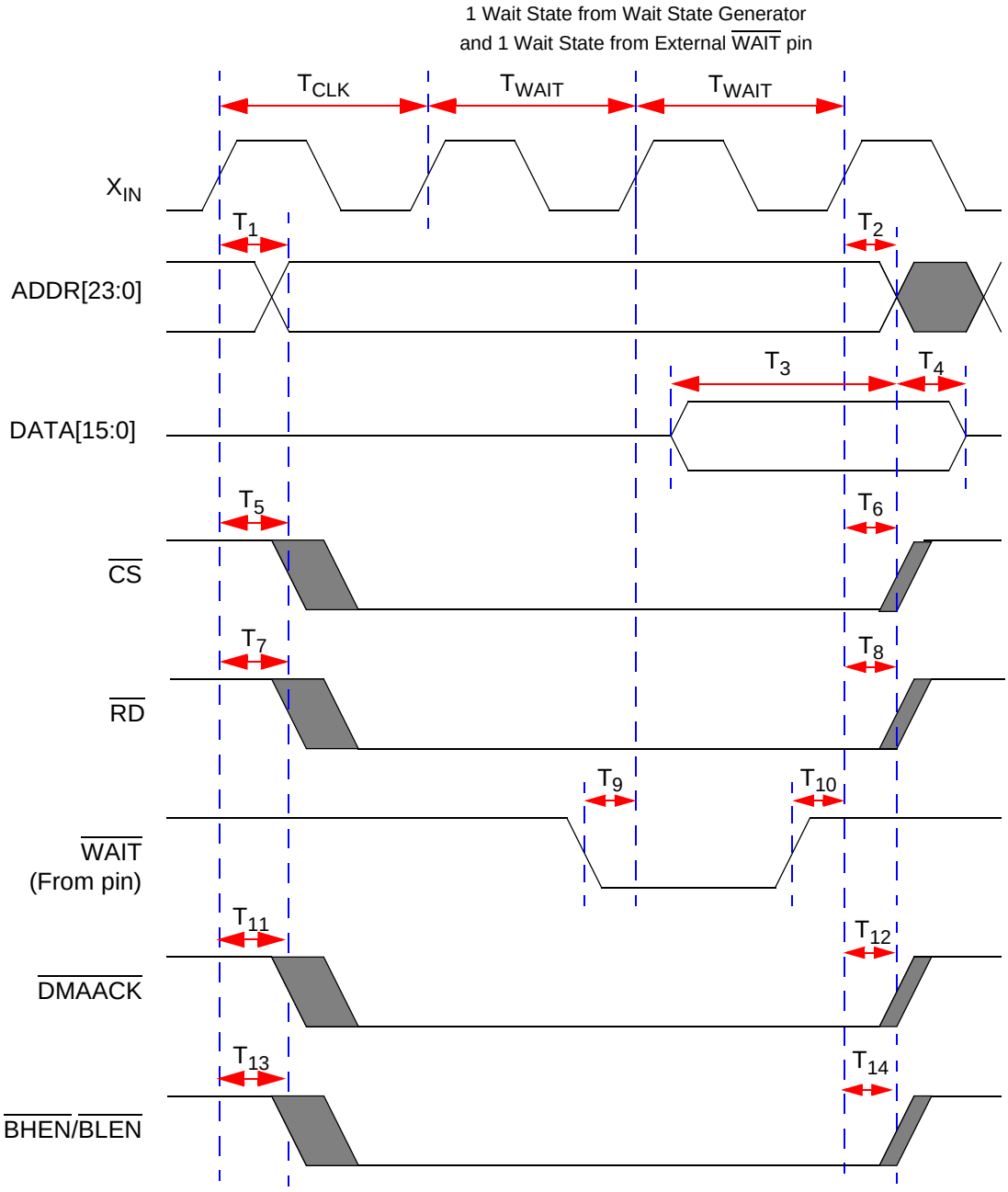


Figure 13. External Interface Timing for a Read Operation, Normal Mode

Figure 14 and Table 16 provide timing information for the External Interface performing a read operation in Normal Mode with a post read wait state. The configuration is the same as in Figure 13, with the exception of the post read wait state.

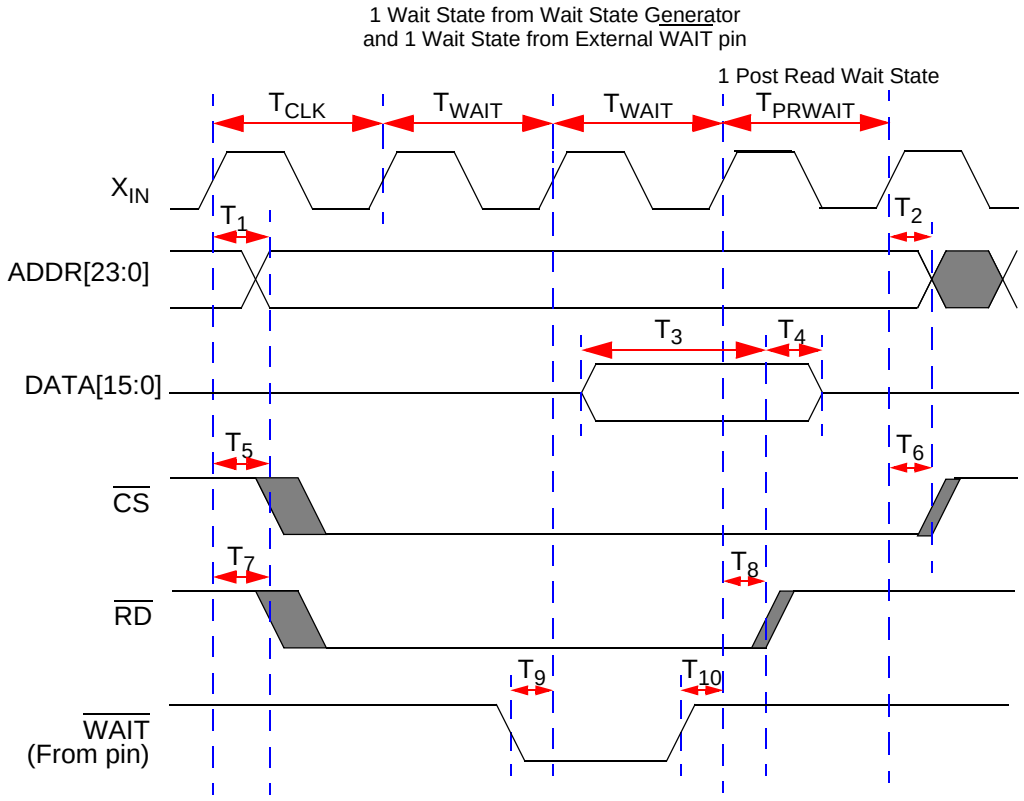


Figure 14. External Interface Timing for a Read Operation, 2 Wait States and 1 Post Read Wait State

External Interface Read Timing, ISA Mode

Figure 15 and Table 17 provide timing information for the external interface performing a read operation in ISA Mode. In Figure 15, it is assumed the wait state generator has been configured to provide 2 wait states during read operations. In Figure 15, it is also assumed that the chip select ($\overline{CS}$) signals have been configured for active Low operation. The Read signal ($\overline{RD}$) timing is shown for both NORMAL and ISA modes.

Table 17. External Interface Timing for a Read Operation, ISA Mode

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	X_{IN} Rise to Address Valid Delay		10
T_2	X_{IN} Rise to Address Output Hold Time	3	
T_3	Data Input Valid to X_{IN} Rise Setup Time		3
T_4	X_{IN} Rise to Data Input Hold Time	3	
T_5	X_{IN} Rise to $\overline{CS}$ Assertion Delay		10
T_6	X_{IN} Rise to $\overline{CS}$ Deassertion Hold Time	3	
T_7	X_{IN} Fall to $\overline{RD}$ Assertion Delay		10
T_8	X_{IN} Fall to $\overline{RD}$ Deassertion Hold Time	3	
T_9	$\overline{WAIT}$ Input Pin Assertion to X_{IN} Rise Setup Time	1	
T_{10}	$\overline{WAIT}$ Input Pin Deassertion to X_{IN} Rise Setup Time	1	
T_{11}	X_{IN} Rise to $\overline{DMAACK}$ Assertion Delay		10
T_{12}	X_{IN} Rise to $\overline{DMAACK}$ Deassertion Hold Time	3	
T_{13}	X_{IN} Rise to $\overline{BHEN}$ or $\overline{BLEN}$ Assertion Delay		10
T_{14}	X_{IN} Rise to $\overline{BHEN}$ or $\overline{BLEN}$ Deassertion Hold Time	3	

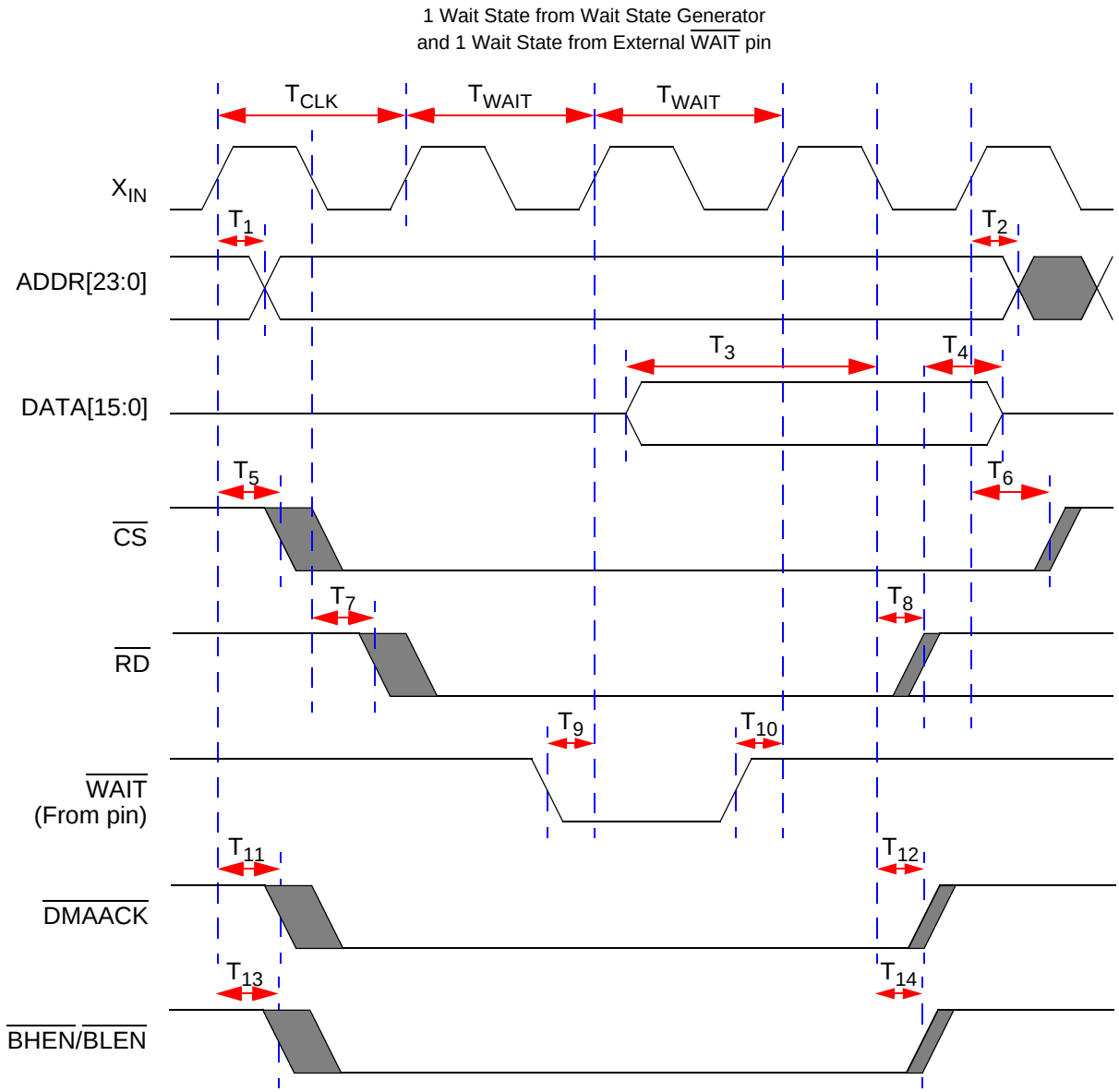


Figure 15. External Interface Timing for a Read Operation, ISA Mode

Reset and Stop Mode Recovery

The reset controller within the ZNEO® Z16F Series controls RESET and Stop Mode Recovery operation. In a typical operation, the following events causes a Reset to occur:

- Power-On Reset
- Voltage Brown-Out
- WDT time-out (when configured through the WDT_RES option bit to initiate a Reset)
- External $\overline{\text{RESET}}$ pin assertion
- OCD initiated Reset (OCDCTL[0] set to 1)
- Fault detect logic

When ZNEO Z16F Series is in Stop Mode, a Stop Mode Recovery is initiated by either of the following operations:

- WDT time-out
- GPIO port input pin transition on an enabled Stop Mode Recovery source

Reset Types

The ZNEO Z16F 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 ZNEO Z16F Series device and the source of the Reset. Table 18 lists the types of Reset and their operating characteristics.

Table 18. Reset and Stop Mode Recovery Characteristics and Latency

	Reset Characteristics and Latency		
Reset Type	**Peripheral Control Registers**	**ZNEO CPU**	**Reset Latency (Delay)**
System Reset	Reset (as applicable)	Reset	A minimum of 66 internal precision oscillator cycles.
Stop Mode Recovery	Unaffected, except RST-SRC and OSCCTL registers	Reset	A minimum of 66 internal precision oscillator cycles.

System Reset

During a System Reset, the ZNEO Z16F Series device is held in Reset for 66 cycles of the IPO. At the beginning of Reset, all GPIO pins are configured as inputs. All GPIO programmable pull-ups are disabled.

At the start of a System Reset, the motor control PWM outputs are forced to high-impedance momentarily. When the option bits that control the off-state have been properly evaluated, the PWM outputs are forced to the programmed off-state.

During Reset, the ZNEO CPU and on-chip peripherals are nonactive; however, the IPO and WDT oscillator continue to run. During the first 50 clock cycles, the internal option bit registers are initialized, after which the system clock for the core and peripherals begins operating. The ZNEO CPU and on-chip peripherals remain nonactive through the next 16 cycles of the system clock, after which the internal reset signal is deasserted.

On Reset, control registers within the register file that have a defined reset value are loaded with their reset values. Other control registers (including the Flags) and general-purpose RAM are undefined following Reset. The ZNEO CPU fetches the Reset vector at program memory address 0004h and loads that value into the program counter. Program execution begins at the Reset vector address.

Table 19 lists the System Reset sources as a function of the operating mode. The following text provides more detailed information about the individual Reset sources. Note that a POR/VBO event always has priority over all other possible reset sources to ensure that a full System Reset occurs.

Table 19. System Reset Sources and Resulting Reset Action

Operating Mode	System Reset Source	Action
NORMAL or HALT modes	POR/VBO	System Reset
	WDT time-out when configured for Reset	System Reset
	$\overline{\text{RESET}}$ pin assertion	System Reset
	Write RSTSCR[0] to 1	System Reset
	Fault detect logic reset	System Reset
Stop Mode	POR/VBO	System Reset
	$\overline{\text{RESET}}$ pin assertion	System Reset
	Fault detect logic reset	System Reset

Power-On Reset

Each device in the ZNEO Z16F Series contains an internal POR circuit. The POR circuit monitors the supply voltage and holds the device in the Reset state until the supply voltage reaches a safe operating level. After the supply voltage exceeds the POR voltage threshold

(V_{POR}) and has stabilized, the POR counter is enabled and counts 50 cycles of the IPO. At this point, the system clock is enabled and the POR counter counts a total of 16 system clock pulses. The device is held in the Reset state until the second POR counter sequence has timed out. After the ZNEO Z16F Series exits the POR state, the ZNEO CPU fetches the Reset vector. Following POR, the POR status bit in the Reset Status and Control Register is set to 1.

Figure 16 displays Power-On Reset operation. For the POR threshold voltage (V_{POR}), see Table 75 on page 343.

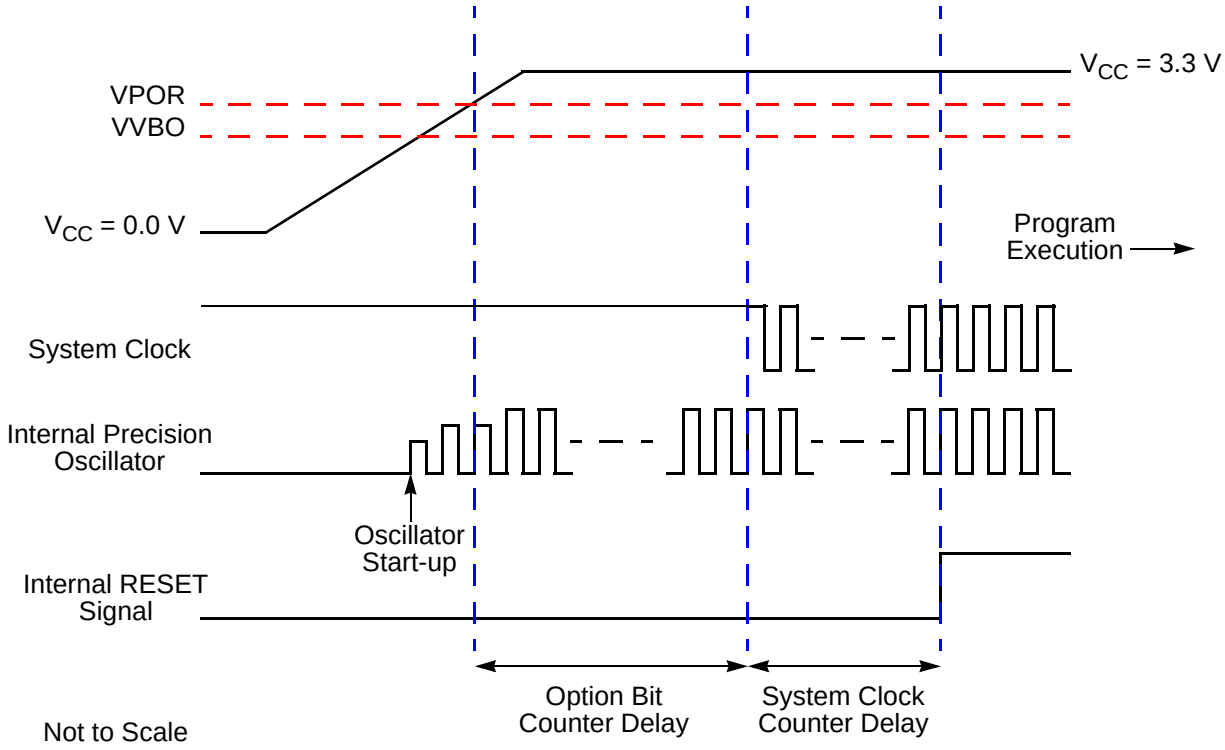


Figure 16. Power-On Reset Operation

Voltage Brown-Out Reset

The ZNEO Z16F Series provides Low Voltage Brown-Out (VBO) protection. The VBO circuit senses the supply voltage when it drops to an unsafe level (below the VBO threshold voltage) and forces the device into the Reset state. While the supply voltage remains below the POR voltage threshold (V_{POR}), the VBO holds the device in the Reset state.

When the supply voltage exceeds the V_{POR} and is stabilized, the device progresses through a full System Reset sequence, as described in the section <CrossRef> Power-On

Reset on page 57. Following Power-On Reset, the POR status bit in the reset source register is set to 1. Figure 17 displays Voltage Brown-Out operation. For VBO and POR threshold voltages (V_{VBO} and V_{POR}), see Figure 75 on page 343.

The VBO circuit is either enabled or disabled during Stop Mode. Operation during Stop Mode is controlled by the VBO_AO option bit. For information about configuring VBO_AO, see the Option Bits chapter on page 292.

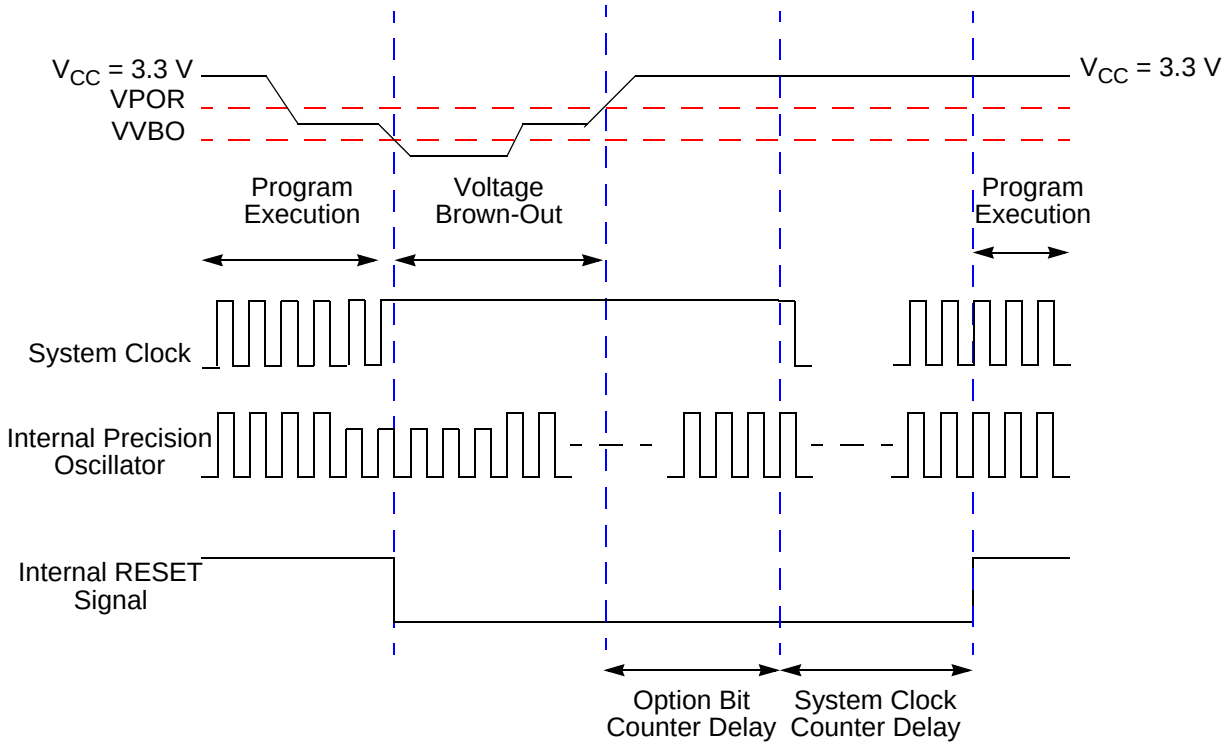


Figure 17. Voltage Brown-Out Reset Operation

Watchdog Timer Reset

If the device is in NORMAL or Halt Mode, the WDT initiates a System Reset at time-out if the WDT_RES option bit is set to 1. This setting is the default (unprogrammed) setting of the WDT_RES option bit. The WDT status bit in the Reset Status and Control Register is set to signify that the reset was initiated by the WDT.

External Pin Reset

The input-only $\overline{\text{RESET}}$ pin has a schmitt-triggered input, an internal pull-up, an analog filter and a digital filter to reject noise. After the $\overline{\text{RESET}}$ pin is asserted for at least four sys-

tem clock cycles, the device progresses through the System Reset sequence. While the $\overline{\text{RESET}}$ input pin is asserted Low, the ZNEO Z16F Series device continues to be held in the Reset state. If the $\overline{\text{RESET}}$ pin is held Low beyond the System Reset time-out, the device exits the Reset state 16 system clock cycles following $\overline{\text{RESET}}$ pin deassertion. If the $\overline{\text{RESET}}$ pin is released before the System Reset time-out, the $\overline{\text{RESET}}$ pin is driven Low by the chip until the completion of the time-out as described in the next section. In Stop Mode, the digital filter is bypassed as the system clock is disabled.

Following a System Reset initiated by the external $\overline{\text{RESET}}$ pin, the EXT status bit in the Reset Status and Control Register is set to 1.

External Reset Indicator

During System Reset, the $\overline{\text{RESET}}$ pin functions as an open drain (active Low) RESET Mode indicator in addition to the input functionality. This Reset output feature allows a ZNEO Z16F Series device to Reset other components to which it is connected, even if the Reset is caused by internal sources such as POR, VBO or WDT events and as an indication of when the reset sequence completes.

After an internal reset event occurs, the internal circuitry begins driving the $\overline{\text{RESET}}$ pin Low. The $\overline{\text{RESET}}$ pin is held Low by the internal circuitry until the appropriate delay listed in Table 18 on page 56 has elapsed.

User Reset

A System Reset is initiated by setting RSTSCR[0]. If the Write was caused by the OCD, the OCD is not Reset.

Fault Detect Logic Reset

Fault detect circuitry exists to detect *illegal* state changes which is caused by transient power or electrostatic discharge events. When such a fault is detected, a system reset is forced. Following the system reset, the FLTD bit in the Reset Status and Control Register is set.

Stop Mode Recovery

Stop Mode is entered by execution of a Stop instruction by the ZNEO CPU. For detailed information about Stop Mode, see the Low-Power Modes chapter on page 64. During Stop Mode Recovery, the device is held in Reset for 66 cycles of the internal precision oscillator.

Stop Mode Recovery only affects the contents of the the Reset Status and Control Register (see page 62) and the Oscillator Control Register (see page 333). 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 ZNEO CPU fetches the Reset vector at program memory addresses 0004h-0007h and loads that value into the program counter. Program execution begins at the Reset vector address. Following Stop Mode Recovery, the Stop bit in the Reset Status and Control Register is set to 1. Table 20 lists the Stop Mode Recovery sources and resulting actions. The following text provides more detailed information about each of the Stop Mode Recovery sources.

Table 20. Stop Mode Recovery Sources and Resulting Action

Operating Mode	Stop Mode Recovery Source	Action
Stop Mode	WDT time-out when configured for Reset	Stop Mode Recovery
	WDT time-out when configured for System Exception	Stop Mode Recovery followed by WDT System Exception
	Data transition on any GPIO Port pin enabled as a Stop Mode Recovery source	Stop Mode Recovery

Stop Mode Recovery Using WDT Time-Out

If the WDT times out during Stop Mode, the device undergoes a Stop Mode Recovery sequence. In the Reset Status and Control Register, the WDT and Stop bits are set to 1. If the WDT is configured to generate a System Exception on time-out, the ZNEO CPU services the WDT System Exception following the normal Stop Mode Recovery sequence.

Stop Mode Recovery Using a GPIO Port Pin Transition

Each of the GPIO port pins is configured as a Stop Mode Recovery input source. If 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 Reset Status and Control Register, the Stop bit is set to 1.

Caution: Short pulses on the port pin initiates Stop Mode Recovery without initiating an interrupt (if enabled for that pin).

Reset Status and Control Register

The Reset Status and Control Register (RSTSCR), shown in Table 21, records the cause of the most recent RESET or Stop Mode Recovery. All status bits are updated on each RESET or Stop Mode Recovery event. Table 22 indicates the possible states of the Reset status bits following a RESET or Stop Mode Recovery event.

Table 21. Reset Status and Control Register (RSTSCR)

Bits	7	6	5	4	3	2	1	0
Field	POR	STOP	WDT	EXT	FLT	USR	Reserved	USER_RST
RESET	See Table 22 for a description of these bits.							
R/W	R	R	R	R	R	R	R	W
Addr	FF–E050h							

Bit	Description
[7] POR	For a description of bits [7:2], see Table 22.
[6] STOP	
[5] WDT	
[4] EXT	
[3] FLT	
[2] USR	
[1]	**Reserved** These bits are reserved and must be programmed to 0.
[0] USER_RST	The USER_RST bit in this register allows software controlled RESET of the part pin. This bit is a Write Only bit that causes a System Reset with the result identified by the USR bit after being executed. 0 = No action. 1 = Causes System Reset.

Table 22. Reset Status Register Values Following Reset

Reset or Stop Mode Recovery Event	POR	STOP	WDT	EXT	FLT	USR
Power-On Reset	1	0	0	0	0	0
Reset using $\overline{\text{RESET}}$ pin assertion	0	0	0	1	0	0
Reset using WDT time-out	0	0	1	0	0	0
Reset from Fault detect logic	0	0	0	0	1	0
Stop Mode Recovery using GPIO pin transition	0	1	0	0	0	0
Stop Mode Recovery using WDT time-out	0	1	1	0	0	0
Reset using software control; write 1 to bit 0 of this register	0	0	0	0	0	1

Low-Power Modes

The ZNEO® Z16F Series products contain advanced integrated power-saving features. Power management functions are divided into three categories to include CPU operating modes, peripheral power control and programmable option bits. The highest level of power reduction is provided through a combination of all functions.

Stop Mode

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

- IPO is stopped; X_{IN} and X_{OUT} pins are driven to V_{SS}.
- System clock is stopped.
- ZNEO CPU is stopped.
- Program counter (PC) stops incrementing.
- If enabled for operation during Stop Mode, the WDT and its internal RC oscillator continue to operate.
- If enabled for operation in Stop Mode through the associated option bit, the VBO protection circuit continues to operate.
- All other on-chip peripherals are nonactive.

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_{DD} or V_{SS}), the VBO protection must be disabled and WDT must be disabled. The device is brought out of Stop Mode using Stop Mode Recovery. For detailed information about Stop Mode Recovery, see the Reset and Stop Mode Recovery section on page 56.

Caution: To prevent excess current consumption when using an external clock source in Stop Mode, the external clock must be disabled.

Halt Mode

Execution of the ZNEO CPU's HALT instruction places the device into Halt Mode. The following list represents the operating characteristics of the ZNEO CPU in Halt Mode:

- System clock is enabled and continues to operate
- ZNEO CPU is stopped
- PC stops incrementing
- WDT's internal RC oscillator continues to operate
- If enabled, the WDT continues to operate
- All other on-chip peripherals continue to operate

The ZNEO CPU is brought out of Halt Mode by any of the following operations:

- Interrupt or System Exception
- WDT time-out (System Exception or Reset)
- Power-On Reset
- VBO reset
- External $\overline{\text{RESET}}$ pin assertion
- Instantaneous Halt Mode recovery

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

Peripheral-Level Power Control

On-chip peripherals in ZNEO Z16F Series parts automatically enter a low power mode after Reset and whenever the peripheral is disabled. To minimize power consumption, unused peripherals must be disabled. See the individual peripheral chapters for specific register settings to enable or disable the peripheral.

Power Control Option Bits

User programmable option bits are available in some versions of the ZNEO Z16F Series devices that enable very low power Stop Mode operation. These options include disabling the VBO protection circuits and disabling the WDT oscillator. For detailed description of the user options that affect power management, see the Option Bits chapter on page 292.

General-Purpose Input/Output

The ZNEO® Z16F Series products contain general-purpose input/output (GPIO) pins arranged as Ports A–K. Each port contains control and data registers. The GPIO control registers are used to determine data direction, open-drain, output drive current and alternate pin functions. Each port pin is individually programmable.

GPIO Port Availability by Device

Table 23 lists the port pins available by device and package pin count.

Table 23. GPIO Port Availability by Device

Device	Pin-Count	Port A	Port B	Port C	Port D	Port E	Port F	Port G	Port H	Port J	Port K
Z16F2811	100-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	[7:0]	[7:0]
	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	—	—
Z16F6411	100-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	[7:0]	[7:0]
	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	—	—
Z16F3211	100-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	[7:0]	[7:0]
	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	—	
Z16F2810	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	—	—
	68-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]	—	—
	64-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]	—	—

Architecture

Figure 18 displays a simplified block diagram of a GPIO port pin. This diagram does not display the ability to accommodate alternate functions and variable port current drive strength.

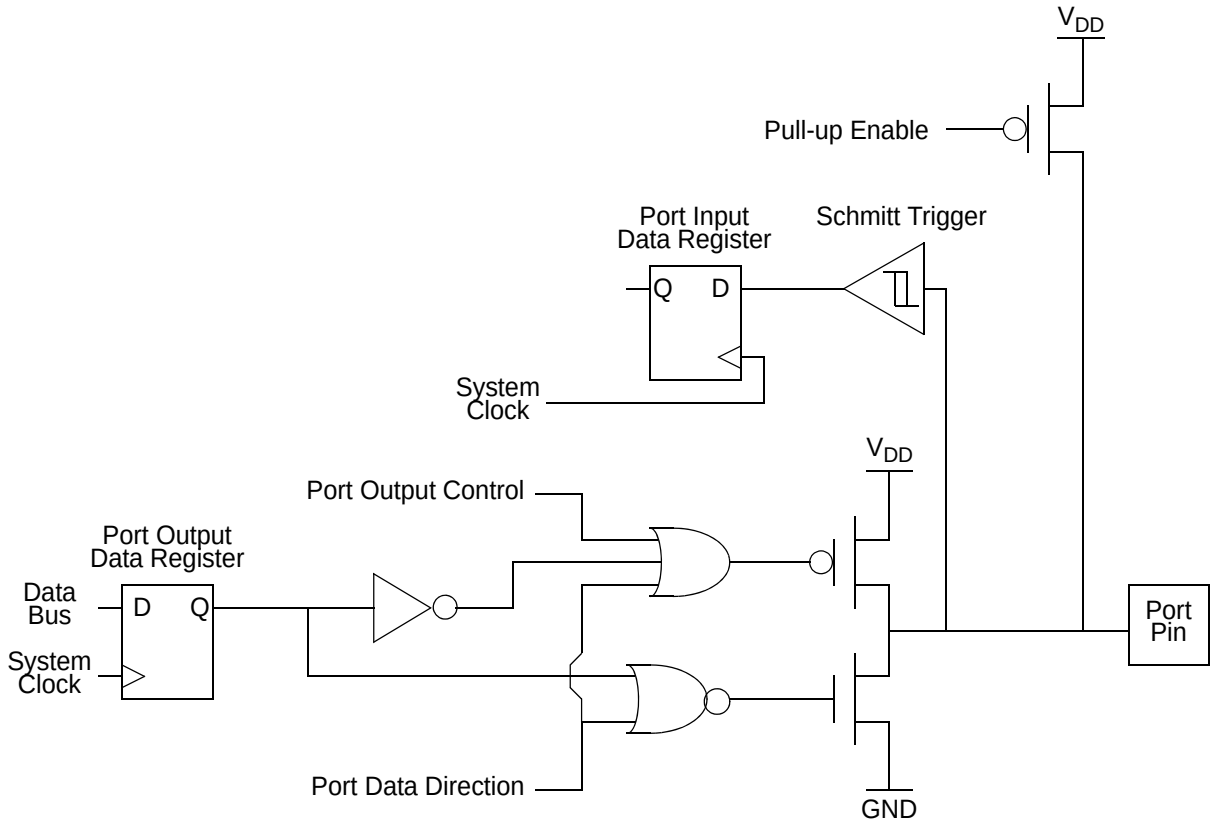


Figure 18. GPIO Port Pin Block Diagram

GPIO Alternate Functions

Many GPIO port pins are used for GPIO and to provide access to the on-chip peripheral functions such as timers, serial communication devices and external data and address bus. The Port A–K alternate function registers configure these pins for either GPIO or alternate function operation. When a pin is configured for alternate function, control of the port pin direction (I/O) is passed from the Port A–K data direction registers to the alternate function assigned to this pin. Table 24 on page 68 lists the alternate functions associated with each port pin.

For detailed information about enabling the external interface data signals, see the External Interface chapter on page 37. When the external interface data signals are enabled for an 8-bit port, the other GPIO functionality including alternate functions cannot be used.

Table 24. Port Alternate Function Mapping

Port	Pin	Alternate Function 1	Alternate Function 2	Alternate Function 3	External Interface
Port A	PA0	T0IN/T0OUT	DMA0REQ	T0INPB	
	PA1	T0OUT	DMA0ACK		
	PA2	DE0	FAULTY		
	PA3	CTS0	FAULT0		
	PA4	RXD0	CS1		
	PA5	TXD0	CS2		
	PA6	SCL	CS3		
	PA7	SDA	CS4		
Port B	PB0/T0IN0	ANA0			
	PB1/T0IN1	ANA1			
	PB2/T0IN2	ANA2			
	PB3	ANA3/OPOUT			
	PB4	ANA4			
	PB5	ANA5			
	PB6	ANA6/OPINP/CINN			
	PB7	ANA7/OPINN			
Port C	PC0	T1IN/T1OUT	DMA1REQ	CINN	
	PC1	T1OUT	DMA1ACK	COMPOUT	
	PC2	SS	CS4		
	PC3	SCK	DMA2REQ		
	PC4	MOSI	DMA2ACK		
	PC5	MISO	CS5		
	PC6	T2IN/T2OUT	PWMH0		
	PC7	T2OUT	PWML0		
Port D	PD0	PWMH1	ADDR[20]		
	PD1	PWML1	ADDR[21]		
	PD2	PWMH2	ADDR[22]		
	PD3	DE1	ADDR[16]		
	PD4	RXD1	ADDR[18]		
	PD5	TXD1	ADDR[19]		
	PD6	CTS1	ADDR[17]		
	PD7	PWML2	ADDR[23]		

Table 24. Port Alternate Function Mapping (Continued)

Port	Pin	Alternate Function 1	Alternate Function 2	Alternate Function 3	External Interface
Port E	PE0				DATA[0]
	PE1				DATA[1]
	PE2				DATA[2]
	PE3				DATA[3]
	PE4				DATA[4]
	PE5				DATA[5]
	PE6				DATA[6]
	PE7				DATA[7]
Port F	PF0	ADDR[0]			
	PF1	ADDR[1]			
	PF2	ADDR[2]			
	PF3	ADDR[3]			
	PF4	ADDR[4]			
	PF5	ADDR[5]			
	PF6	ADDR[6]			
	PF7	ADDR[7]			
Port G	PG0	ADDR[8]			
	PG1	ADDR[9]			
	PG2	ADDR[10]			
	PG3	ADDR[11]			
	PG4	ADDR[12]			
	PG5	ADDR[13]			
	PG6	ADDR[14]			
	PG7	ADDR[15]			
Port H	PH0	ANA8	WR		
	PH1	ANA9	RD		
	PH2	ANA10	$\overline{CS0}$		
	PH3	ANA11/CPINP	WAIT		

Table 24. Port Alternate Function Mapping (Continued)

Port	Pin	Alternate Function 1	Alternate Function 2	Alternate Function 3	External Interface
Port J	PJ0				DATA[8]
	PJ1				DATA[9]
	PJ2				DATA[10]
	PJ3				DATA[11]
	PJ4				DATA[12]
	PJ5				DATA[13]
	PJ6				DATA[14]
	PJ7				DATA[15]
Port K	PK0	BHEN			
	PK1	BLEN			
	PK2	$\overline{CS0}$			
	PK3	$\overline{CS1}$			
	PK4	$\overline{CS2}$			
	PK5	$\overline{CS3}$			
	PK6	$\overline{CS4}$			
	PK7	$\overline{CS5}$			

GPIO Interrupts

Many of the GPIO port pins are used as interrupt sources. Some port pins are configured to generate an interrupt request on either the rising edge or falling edge of the pin input signal. Other port pin interrupts generate an interrupt when any edge occurs (both rising and falling). For more information about interrupts using the GPIO pins, see the Interrupt Controller chapter on page 80.

GPIO Control Register Definitions

Port A-K Input Data Registers

Reading from the Port A-K Input Data registers, shown in Table 25, returns the sampled values from the corresponding port pins. The Port A-K Input Data registers are Read-only.

Table 25. Port A-K Input Data Registers (PxIN)

Bits	7	6	5	4	3	2	1	0
Field	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
Addr	FF_E100, FF_E110, FF_E120, FF_E130, FF_E140, FF_E150, FF_E160, FF_E170, FF_E180, FF_E190							

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

Port A-K Output Data Registers

The Port A-K Output Data registers, shown in Table 26, write output data to the pins.

Table 26. Port A-K Output Data Registers (P*x*OUT)

Bits	7	6	5	4	3	2	1	0
Field	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E101, FF_E111, FF_E121, FF_E131, FF_E141, FF_E151, FF_E161, FF_E171, FF_E181, FF_E191							

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

Port A-K Data Direction Registers

The Port A-K Data Direction registers, shown in Table 27, configure the specified port pins as either inputs or outputs.

Table 27. Port A-K Data Direction Registers (PxDD)

Bits	7	6	5	4	3	2	1	0
Field	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E102, FF_E112, FF_E122, FF_E132, FF_E142, FF_E152, FF_E162, FF_E172, FF_E182, FF_E192							

Bit	Description
[7:0] DD[7:0]	**Data Direction** These bits control the direction of the associated port pin. Port alternate function operation overrides the data direction register setting. 0 = Output Data in the Port A-K Output Data Register is driven onto the port pin. 1 = Input The port pin is sampled and the value written into the Port A-K Input Data Register. The output driver is high impedance.

Port A-K High Drive Enable Registers

Setting the bits in the Port A-K High Drive Enable registers, shown in Table 28, to 1, configures the specified port pins for high current output drive operation. The Port A-K High Drive Enable registers affect the pins directly and as a result, alternate functions are also affected.

Table 28. Port A-K High Drive Enable Registers (PxHDE)

Bits	7	6	5	4	3	2	1	0
Field	PHDE7	PHDE6	PHDE5	PHDE4	PHDE3	PHDE2	PHDE1	PHDE0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E103, FF_E113, FF_E123, FF_E133, FF_E143, FF_E153, FF_E163, FF_E173, FF_E183, FF_E193							

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

Port A-K Alternate Function High and Low Registers

The Port A-K Alternate Function High and Low registers, shown in Table 29 and Table 30 on page 75, 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 67. When changing alternate functions, it is recommended to use word data mode instructions to perform simultaneous Writes to the Port Alternate Function High and Low registers.

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

Table 29. Port A-K Alternate Function High Registers (PxAFH)

Bits	7	6	5	4	3	2	1	0
Field	AFH[7]	AFH[6]	AFH[5]	AFH[4]	AFH[3]	AFH[2]	AFH[1]	AFH[0]
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E104, FF_E124, FF_E134, FF_E174							

Table 30. Port A-K Alternate Function Low Registers (PxAFL)

Bits	7	6	5	4	3	2	1	0
Field	AFL[7]	AFL[6]	AFL[5]	AFL[4]	AFL[3]	AFL[2]	AFL[1]	AFL[0]
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E105, FF_E115, FF_E125, FF_E135, FF_E155, FF_E165, FF_E175, FF_E195							

Table 31. Alternate Function Enabling

AFH[*x*]	AFL[*x*]	Priority
0	0	No Alternate Function Enabled
0	1	Alternate Function 1 Enabled
1	0	Alternate Function 2 Enabled
1	1	Alternate Function 3 Enabled
Note: *x* indicates the register bits from 0 through 7.		

Port A-K Output Control Registers

Setting the bits in the Port A-K Output Control registers to 1, shown in Table 32, configures the specified port pins for open-drain operation. These registers affect the pins directly and as a result, alternate functions are also affected. Enabling the I^2C controller automatically configures the SCL and SDA pins as open-drain; independent of the setting in the output control registers that have the SCL and SDA alternate functions.

Table 32. Port A-K Output Control Registers (P*x*OC)

Bits	7	6	5	4	3	2	1	0
Field	POC7	POC6	POC5	POC4	POC3	POC2	POC1	POC0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E106, FF_E116, FF_E126, FF_E136, FF_E146, FF_E156, FF_E166, FF_E176, FF_E186, FF_E196							

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

Port A-K Pull-Up Enable Registers

Setting the bits in the Port A-K Pull-Up Enable registers to 1 enables a weak internal resistive pull-up on the specified port pins. These registers affect the pins directly and as a result, alternate functions are also affected. See Table 33.

Table 33. Port A-K Pull-Up Enable Registers (P*x*PUE)

Bits	7	6	5	4	3	2	1	0
Field	PUE7	PUE6	PUE5	PUE4	PUE3	PUE2	PUE1	PUE0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E107, FF_E117, FF_E127, FF_E137, FF_E147, FF_E157, FF_E167, FF_E177, FF_E187, FF_E197							

Bit	Description
[7:0] PUE[7:0]	**Port Pull-Up Enable** These bits function independently of the alternate function bit and enable the weak pull-up if set to 1. 0 = The weak pull-up on the port pin is disabled. 1 = The weak pull-up on the port pin is enabled.

Port A-K Stop Mode Recovery Source Enable Registers

Setting the bits in the Port A-K Stop Mode Recovery Source Enable registers to 1, shown in Table 34, 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 34. Port A-K Stop Mode Recovery Source Enable Registers (P*x*SMRE)

Bits	7	6	5	4	3	2	1	0
Field	PSMRE7	PSMRE6	PSMRE5	PSMRE4	PSMRE3	PSMRE2	PSMRE1	PSMRE0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E108, FF_E118, FF_E128, FF_E138, FF_E148, FF_E158, FF_E168, FF_E178, FF_E188, FF_E198							

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

Port A IRQ MUX1 Register

The Port IRQ MUX1 Register, shown in Table 35, selects either Port A/D pins or the comparator/DBG channel as interrupt sources.

Table 35. Port A IRQ MUX1 Register (PAIMUX1)

Bits	7	6	5	4	3	2	1	0
Field	CPIMUX	Reserved						DBGIMUX
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R	R	R	R	R	R/W
Addr	FF_E10C							

Bit	Description
[7] CPIMUX	**Comparator Interrupt MUX** 0 = Select Port A7/D7 based upon the Port A IRQ edge register as the interrupt source. 1 = Select the comparator as the interrupt source.

Bit	Description
[6:1]	**Reserved** These bits are reserved and must be programmed to 000000.
[0] DBGIMUX	**Debug Interrupt MUX** 0 = Select Port A0/D0 based on the Port A IRQ edge register as the interrupt source. 1 = Select the DBG as the interrupt source.

Port A IRQ MUX Register

The Port IRQ MUX Register, shown in Table 36, selects either Port A or Port D pins as interrupt sources.

Table 36. Port A IRQ MUX Register (PAIMUX)

Bits	7	6	5	4	3	2	1	0
Field	PAIMUX7	PAIMUX6	PAIMUX5	PAIMUX4	PAIMUX3	PAIMUX2	PAIMUX1	PAIMUX0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E10E							

Bit	Description
PAIMUX[7:0]	**Port A/D Interrupt Source** 0 = Select Port A*x* as interrupt source. 1 = Select Port D*x* as interrupt source.

Port A IRQ Edge Register

The Port IRQ Edge Register, shown in Table 37, selects either positive or negative edge as the port pin interrupt sources.

Table 37. Port A IRQ Edge Register (PAIEDGE)

Bits	7	6	5	4	3	2	1	0
Field	PAIEDGE7	PAIEDGE6	PAIEDGE5	PAIEDGE4	PAIEDGE3	PAIEDGE2	PAIEDGE1	PAIEDGE0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E10F							

Bit	Description
[7:0] PAIEDGE[7:0]	**Port A/D Interrupt Edge** 0 = Select Port A/D pin negedge as interrupt source. 1 = Select Port A/D pins posedge as interrupt source.

Port C IRQ MUX Register

The Port C IRQ MUX Register, shown in Table 38, selects either Port C pins or the DMA channels as interrupt sources.

Table 38. Port C IRQ MUX Register (PCIMUX)

Bits	7	6	5	4	3	2	1	0
Field	Reserved				PCIMUX3	PCIMUX2	PCIMUX1	PCIMUX0
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Addr	FF_E12E							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 000000.
[3:0] PCIMUX[3:0]	**Port C Interrupt MUX** 0 = Select DMA Chan[3:0] as interrupt source. 1 = Select port C pins as interrupt source.

Interrupt Controller

The interrupt controller on the ZNEO® Z16F Series products prioritize interrupt requests from on-chip peripherals and the GPIO port pins. The features of the interrupt controller includes:

- Flexible GPIO interrupts:
  - Eight selectable rising and falling edge GPIO interrupts
  - Four dual-edge interrupts
- Three levels of individually programmable interrupt priority
- Software Interrupt Requests (IRQ) assertion

The IRQs allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start an ISR. Usually this service routine is involved with 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.

System exceptions are nonmaskable requests which allow critical system functions to suspend CPU operation in an orderly manner and force the CPU to start a service routine. Usually this service routine tries to determine how critical the exception is. When the service routine is complete, the CPU returns to the operation from which it was interrupted.

The ZNEO Z16F Series 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 ZNEO CPU, refer to the ZNEO CPU Core User Manual (UM0188), available for download at www.zilog.com.

Interrupt Vector Listing

Table 39 lists all of the interrupts available in order of priority.

Table 39. Interrupt Vectors in Order of Priority

Priority	Program Memory Vector Address	Programmable Priority?	Interrupt Source
Highest	0004h	No	Reset (not an interrupt)
	0008h	No	System Exceptions
	000Ch	No	Reserved
	0010h	Yes	Timer 2
	0014h	Yes	Timer 1
	0018h	Yes	Timer 0
	001Ch	Yes	UART 0 receiver
	0020h	Yes	UART 0 transmitter
	0024h	Yes	I^2C
	0028h	Yes	SPI
	002Ch	Yes	ADC0
	0030h	Yes	Port A7 or Port D7, rising or falling input edge or Comparator output rising and falling edge (source selected in PortA IRQ MUX registers)
	0034h	Yes	Port A6 or Port D6, rising or falling input edge
	0038h	Yes	Port A5 or Port D5, rising or falling input edge
	003Ch	Yes	Port A4 or Port D4, rising or falling input edge
	0040h	Yes	Port A3 or Port D3, rising or falling input edge
	0044h	Yes	Port A2 or Port D2, rising or falling input edge
	0048h	Yes	Port A1 or Port D1, rising or falling input edge
	004Ch	Yes	Port A0 or Port D0, rising or falling input edge or OCD Interrupt (source selected in PortA IRQ MUX registers)
	0050h	Yes	PWM Timer
	0054h	Yes	UART 1 receiver
	0058h	Yes	UART 1 transmitter
	005Ch	Yes	PWM Fault
	0060h	Yes	Port C3, both input edges/DMA 3
	0064h	Yes	Port C2, both input edges/DMA 2
	0068h	Yes	Port C1, both input edges/DMA 1
Lowest	006Ch	Yes	Port C0, both input edges/DMA 0

The most significant byte (MSB) of the four byte interrupt vector is not used. The vector is stored in the three least significant byte (LSB) of the vector, as shown in Table 40.

Table 40. Interrupt Vector Placement

Vector Byte	Data
0	Reserved
1	IRQ Vector[23:16]
2	IRQ Vector[15:8]
3	IRQ Vector[7:0]

Architecture

Figure 19 displays a block diagram of the interrupt controller.

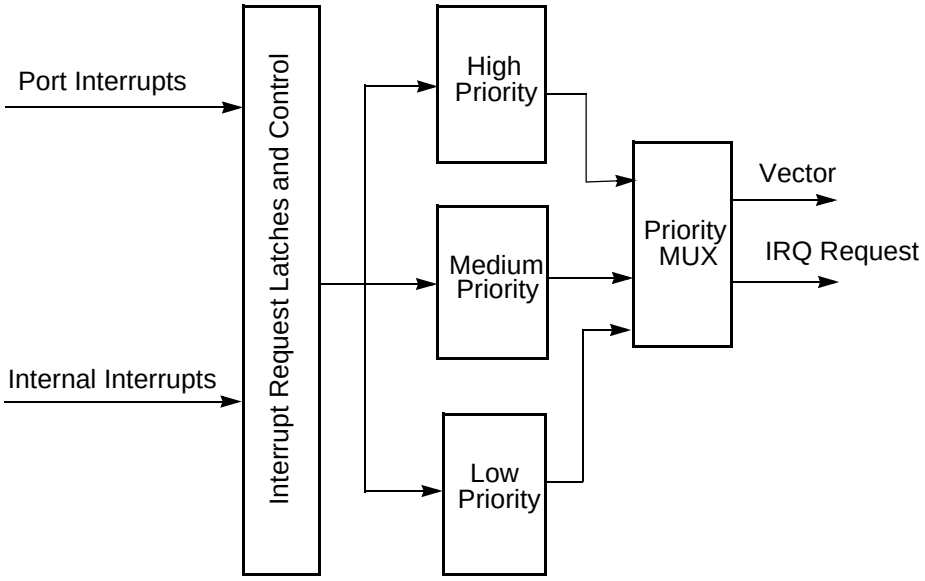


Figure 19. Interrupt Controller Block Diagram

Operation

Master Interrupt Enable

The master interrupt enable bit in the flag register globally enables or disables interrupts. This bit has been moved to the flag register (bit 0). Thus, anytime the register is loaded, it

changes the state of the IRQE bit. For the IRET instruction, the bit is set based on what has been pushed on the stack.

Interrupts are globally enabled by any of the following actions:

- Execution of an Enable Interrupt (EI) instruction
- Writing 1 to the IRQE bit in the flag register

Interrupts are globally disabled by any of the following actions:

- Execution of a Disable Interrupt (DI) instruction
- ZNEO CPU acknowledgement of an interrupt service request from the interrupt controller
- Writing 0 to the IRQE bit in the flag register
- Reset
- Execution of a TRAP instruction
- All System Exceptions

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 are enabled with identical interrupt priority (for example, all interrupts enabled as Level 2 interrupts), the interrupt priority is assigned from highest to lowest as specified in Table 39 on page 81. 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 levels (Level 1, Level 2 or Level 3), priority is assigned as specified in Table 39. Reset and System Exceptions have the highest priority.

System Exceptions

System Exceptions are generated for stack overflow, illegal instructions, divide-by-zero and divide overflow, etc. The System Exceptions are not affected by the IRQE and share a single vector.

Each exception has a bit in the system exception status register. When a system exception occurs it pushes the program counter and the flags on the stack, fetches the system exception vector from `000008h` (similar to a IRQ) and the bit associated with that exception is set in the status register. Additional exceptions from the same source are blocked until the status bit of the particular exception is cleared by writing 1 tothat status bit. Other types of exceptions occur while servicing an exception. When this happens the processor again

vectors to the system exception vector and sets the associated exception status bit. The service routine would then have to respond to the new exception.

Note: When an illegal instruction occurs, the program counter and flags are pushed onto the stack only once. If the associated exception bit is not Reset, the program counter and flags cannot be pushed a second time.

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 ZNEO CPU, the corresponding bit in the interrupt request register is cleared until the next interrupt occurs. Writing 1 to the corresponding bit in the interrupt request register clears the interrupt request.

Program code generates interrupts directly. Writing a 1 to the appropriate bit in the interrupt request set register triggers an interrupt (assuming that interrupts are enabled). When the interrupt request is acknowledged by the ZNEO CPU, the bit in the interrupt request register is automatically cleared to 0.

System Exception Status Registers

When a System Exception occurs, the System Exception Status registers are read to determine which system exception occurred. These registers are read individually or read as a 16-bit quantity.

Table 41. System Exception Register High (SYSEXCPH)

Bits	7	6	5	4	3	2	1	0
Field	SPOVF	PCOVF	DIV0	DIVOVF	ILL	Reserved		
RESET	0	0	0	0	0	0	0	0
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Addr	FF_E020h							

Bit	Description
[7] SPOVF	**Stack Pointer Overflow** If this bit is 1, a stack pointer overflow exception occurred. Writing 1 to this bit clears it to 0.
[6] PCOVF	**Program Counter Overflow** If this bit is 1, a program counter overflow exception occurred. Writing 1 to this bit clears it to 0.

Bit	Description (Continued)
[5] DIV0	**Divide by Zero** If this bit is 1, a divide operation was executed where the denominator was zero. Writing 1 to this bit clear it to 0.
[4] DIVOVF	**Divide Over Flow** If this bit is 1, a divide overflow occurred. A divide overflow happens when the result is greater than FFFFFFFFh. Writing 1 to this bit clears it to 0.
[3] ILL	**Illegal Instruction** If this bit is 1, an illegal instruction occurred. Writing 1 to this bit clears it to 0.
[2:0]	**Reserved** These bits are reserved and must be programmed to 000.

Table 42. System Exception Register Low (SYSEXCPL)

Bits	7	6	5	4	3	2	1	0
Field	Reserved					WDTOSC	PRIOSC	WDT
RESET	0	0	0	0	0	0	0	0
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Addr	FF_E021h							

Bit	Description
[7:3]	**Reserved** These bits are reserved and must be programmed to 00000.
[2] WDTOSC	**WDT Oscillator Fail** If this bit is 1, a WDT oscillator fail exception occurred. Writing 1 to this bit clears it to 0.
[1] PRIOSC	**Primary Oscillator Fail** If this bit is 1, a primary oscillator fail exception occurred. Writing 1 to this bit clears it to 0.
[0] WDT	**Watchdog Timer Interrupt** If this bit is 1, a WDT exception occurred. Writing 1 to this bit clears it to 0.

Last IRQ Register

When an interrupt occurs, the 5th bit value of the interrupt vector is stored in the Last IRQ Register, shown in Table 43. This register allows the software to determine which interrupt source was last serviced. It is used by RTOS which have a single interrupt entry point. To implement this the software must set all interrupt vectors to the entry point address. The entry point service routine then reads this register to determine which source caused the interrupt or exception and respond accordingly.

Table 43. Last IRQ Register (LASTIRQ)

Bits	7	6	5	4	3	2	1	0
Field	Always 0	IRQADR					Always 00	
RESET	0	0	0	0	0	1	0	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R	R
Addr	FF_E023h							

Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) Register, shown in Table 44, 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 ZNEO CPU. If interrupts are globally disabled (polled interrupts), the ZNEO CPU reads the Interrupt Request 0 Register to determine if any interrupt requests are pending. Writing 1 to the bits in this register clears the interrupt. The bits of this register are set by writing 1 to the Interrupt Request 0 Set Regsiter (IRQ0SET) at address FF_E031h.

Table 44. Interrupt Request 0 Register (IRQ0) and Interrupt Request 0 Set Register (IRQ0SET)

Bits	7	6	5	4	3	2	1	0
Field	T2I	T1I	T0I	U0RXI	U0TXI	I2CI	SPII	ADCI
RESET	0	0	0	0	0	0	0	0
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Addr	FF_E030h							
Field	T2I	T1I	T0I	U0RXI	U0TXI	I2CI	SPII	ADCI
RESET	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W
Addr	FF_E031h							
Note: IRQ0SET at address FF_E031h is write only and used to set the interrupts identified.								

Bit	Description
[7] T2I	**Timer 2 Interrupt Request** 0 = No interrupt request is pending for timer 2. 1 = An interrupt request from timer 2 is awaiting service. Writing 1 to this bit resets it to 0.
[6] T1I	**Timer 1 Interrupt Request** 0 = No interrupt request is pending for timer 1. 1 = An interrupt request from timer 1 is awaiting service. Writing 1 to this bit resets it to 0.

Bit	Description (Continued)
[5] T0I	**Timer 0 Interrupt Request** 0 = No interrupt request is pending for timer 0. 1 = An interrupt request from timer 0 is awaiting service. Writing 1 to this bit resets it to 0.
[4] U0RXI	**UART 0 Receiver Interrupt Request** 0 = No interrupt request is pending for the UART 0 receiver. 1 = An interrupt request from the UART 0 receiver is awaiting service. Writing 1 to this bit resets it to 0.
[3] U0TXI	**UART 0 Transmitter Interrupt Request** 0 = No interrupt request is pending for the UART 0 transmitter. 1 = An interrupt request from the UART 0 transmitter is awaiting service. Writing 1 to this bit resets it to 0.
[2] I2CI	**I^2C Interrupt Request** 0 = No interrupt request is pending for the I^2C. 1 = An interrupt request from the I^2C is awaiting service. Writing 1 to this bit resets it to 0.
[1] SPII	**SPI Interrupt Request** 0 = No interrupt request is pending for the SPI. 1 = An interrupt request from the SPI is awaiting service. Writing 1 to this bit resets it to 0.
[0] ADCI	**ADC Interrupt Request** 0 = No interrupt request is pending for ADC. 1 = An interrupt request from ADC is awaiting service. Writing 1 to this bit resets it to 0.

Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) Register, shown in Table 45, 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 ZNEO CPU. If interrupts are globally disabled (polled interrupts), the ZNEO CPU reads the Interrupt Request 1 Register to determine, if any interrupt requests are pending. Writing 1 to the bits in this register clears the interrupt. The bits of this register are set by writing 1 to the Interrupt Request 1 Set Register (IRQ1SET) at address FF_E035h.

Table 45. Interrupt Request1 Register (IRQ1) and Interrupt Request1 Set Register (IRQ1SET)

Bits	7	6	5	4	3	2	1	0
Field	PAD7I	PAD6I	PAD5I	PAD4I	PAD3I	PAD2I	PAD1I	PAD0I
RESET	0	0	0	0	0	0	0	0
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Addr	FF_E034h							
Field	PAD7I	PAD6I	PAD5I	PAD4I	PAD3I	PAD2I	PAD1I	PAD0I
RESET	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W
Addr	FF_E035h							

Note: IRQ1SET at address FF_E035h is write only and used to set the interrupts identified.

Bit	Description
[7:0] PADxI	**Port A/D Pin *x* Interrupt Request** 0 = No interrupt request is pending for GPIO Port A/D pin *x*. 1 = An interrupt request from GPIO Port A/D pin *x* is awaiting service. Writing 1 to these bits resets it to 0. Here, *x* indicates the specific GPIO port pin number (0 through 7). PAD7I and PAD0I have interrupt sources other than Port A and Port D as selected by the Port A IRQ MUX registers. PAD7I is configured to provide the comparator interrupt. PAD0I is configured to provide the OCD interrupt.

Note: The above IRQ1 bits are set any time the selected port is toggled. The setting of these bits are not affected by the associated interrupt enable bits.

Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register, shown in Table 46, 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 ZNEO CPU. If interrupts are globally disabled (polled interrupts), the ZNEO CPU reads the Interrupt Request 1 Register to determine, if any interrupt requests are pending. Writing 1 to the bits in this register clears the interrupt. The bits of this register are set bywriting 1 to the Interrupt Request 2 Set Register (IRQ2SET) at address FF_E039h.

Table 46. Interrupt Request 2 Register (IRQ2) and Interrupt Request 2 Set Register (IRQ2SET)

Bits	7	6	5	4	3	2	1	0
Field	PWMTI	U1RXI	U1TXI	PWMFI	PC3I/ DMA3I	PC2I/ DMA2I	PC1I/ DMA1I	PC0I/ DMA0I
RESET	0	0	0	0	0	0	0	0
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Addr	FF_E038h							
Field	PWMTI	U1RXI	U1TXI	PWMFI	PC3I/ DMA3I	PC2I/ DMA2I	PC1I/ DMA1I	PC0I/ DMA0I
RESET	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W
Addr	FF_E039h							

Note: IRQ2SET at address FF_E039h is write only and used to set the interrupts identified.

Bit	Description
[7] PWMTI	**PWM Timer Interrupt Request** 0 = No interrupt request is pending for the PWM timer. 1 = An interrupt request from the PWM timer is awaiting service. Writing 1 to this bit resets it to 0.
[6] U1RXI	**UART 1 Receiver Interrupt Request** 0 = No interrupt request is pending for the UART 1 receiver. 1 = An interrupt request from the UART 1 receiver is awaiting service. Writing 1 to this bit resets it to 0.
[5] U1TXI	**UART 1 Transmitter Interrupt Request** 0 = No interrupt request is pending for the UART 1 transmitter. 1 = An interrupt request from the UART 1 transmitter is awaiting service. Writing 1 to this bit resets it to 0.
[4] PWMFI	**PWM Fault Interrupt Request** 0 = No interrupt request is pending for the PWM fault. 1 = An interrupt request from the PWM fault is awaiting service. Writing 1 to this bit resets it to 0.
[3:0] PCxI/ DMAxI	**Port C Pin *x* or DMA x Interrupt Request** 0 = No interrupt request is pending for GPIO port C pin x or DMA x. 1 = An interrupt request from GPIO port C pin *x* or DMAx is awaiting service. Writing 1 to this bit resets it to 0. Where *x* indicates the specific GPIO port C pin or DMA number (0 through 3).

IRQ0 Enable High and Low Bit Registers

The IRQ0 enable high and low bit registers, shown in Tables 48 and 49, form a priority encoded enabling for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register. Table 47 describes the priority control for IRQ0.

Table 47. IRQ0 Enable and Priority Encoding

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

Note: *x* indicates the range of register bits 0 through 7.

Table 48. IRQ0 Enable High Bit Register (IRQ0ENH)

Bits	7	6	5	4	3	2	1	0
Field	T2ENH	T1ENH	T0ENH	U0RENH	U0TENH	I2CENH	SPIENH	ADCENH
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E032h							

Bit	Description
[7] T2ENH	Timer 2 Interrupt Request Enable High Bit.
[6] T1ENH	Timer 0 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 49. IRQ0 Enable Low Bit Register (IRQ0ENL)

Bits	7	6	5	4	3	2	1	0
Field	T2ENL	T1ENL	T0ENL	U0RENL	U0TENL	I2CENL	SPIENL	ADCENL
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E033h							

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

The IRQ1 Enable High and Low Bit registers, shown in Tables 51 and 52, form a priority encoded enabling for interrupts in the Interrupt Request 1 Register. Priority is generated by setting bits in each register. Table 50 describes the priority control for IRQ1.

Table 50. IRQ1 Enable and Priority Encoding

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

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

Table 51. IRQ1 Enable High Bit Register (IRQ1ENH)

Bits	7	6	5	4	3	2	1	0
Field	PAD7ENH	PAD6ENH	PAD5ENH	PAD4ENH	PAD3ENH	PAD2ENH	PAD1ENH	PAD0ENH
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E036h							

Bit	Description
[7:0] PAD*x*ENH	Port A/D Bit[*x*] Interrupt Request Enable High Bit.

Table 52. IRQ1 Enable Low Bit Register (IRQ1ENL)

Bits	7	6	5	4	3	2	1	0
Field	PAD7ENL	PAD6ENL	PAD5ENL	PAD4ENL	PAD3ENL	PAD2ENL	PAD1ENL	PAD0ENL
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E037h							

Bit	Description
[7:0] PAD*x*ENL	Port A/D Bit[*x*] Interrupt Request Enable Low Bit.

IRQ2 Enable High and Low Bit Registers

The IRQ2 Enable High and Low Bit registers, shown in Tables 54 and 55, form a priority encoded enabling for interrupts in the Interrupt Request 2 Register. Priority is generated by setting bits in each register. Table 53 describes the priority control for IRQ2.

Table 53. IRQ2 Enable and Priority Encoding

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

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

Table 54. IRQ2 Enable High Bit Register (IRQ2ENH)

Bits	7	6	5	4	3	2	1	0
Field	PWMTENH	U1RENH	U1TENH	PWMFENH	C3ENH/ DMA3ENH	C2ENH/ DMA2ENH	C1ENH/ DMA1ENH	C0ENH/ DMA0ENH
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E03Ah							

Bit	Description
[7] PWMTENH	PWM Timer 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] PWMFENH	PWM Fault Interrupt Request Enable High Bit.
[3:0] CxENH/ DMAxENH	Port C*x or DMAx* Interrupt Request Enable High Bit.

Table 55. IRQ2 Enable Low Bit Register (IRQ2ENL)

Bits	7	6	5	4	3	2	1	0
Field	PWMTENL	U1RENL	U1TENL	PWMFENL	C3ENL/ DMA3ENL	C2ENL/ DMA2ENL	C1ENL/ DMA1ENL	C0ENL/ DMA0ENL
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E03Bh							

Bit	Description
[7] PWMTENL	PWM Timer 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.

Bit	Description
[4] PWMFENL	PWM Fault Interrupt Request Enable Low Bit.
[3:0] CxENL/ DMAxENL	Port C*x* or *DMAx* Interrupt Request Enable Low Bit.

Timers

The ZNEO® Z16F Series contains three 16-bit reloadable timers used for timing, event counting or generation of pulse width modulated (PWM) signals. These timers include the following features:

- 16-bit reload counter
- Programmable prescaler with values ranging from 1 to 128
- PWM output generation (single or differential)
- Capture and compare capability
- External input pin for event counting, clock gating or capture signal
- Complementary timer output pins
- Timer interrupt

Architecture

Capture and compare capability measures the velocity from a tachometer wheel or reads sensor outputs for rotor position for brushless DC motor commutation. Figure 20 displays the architecture of the timer.

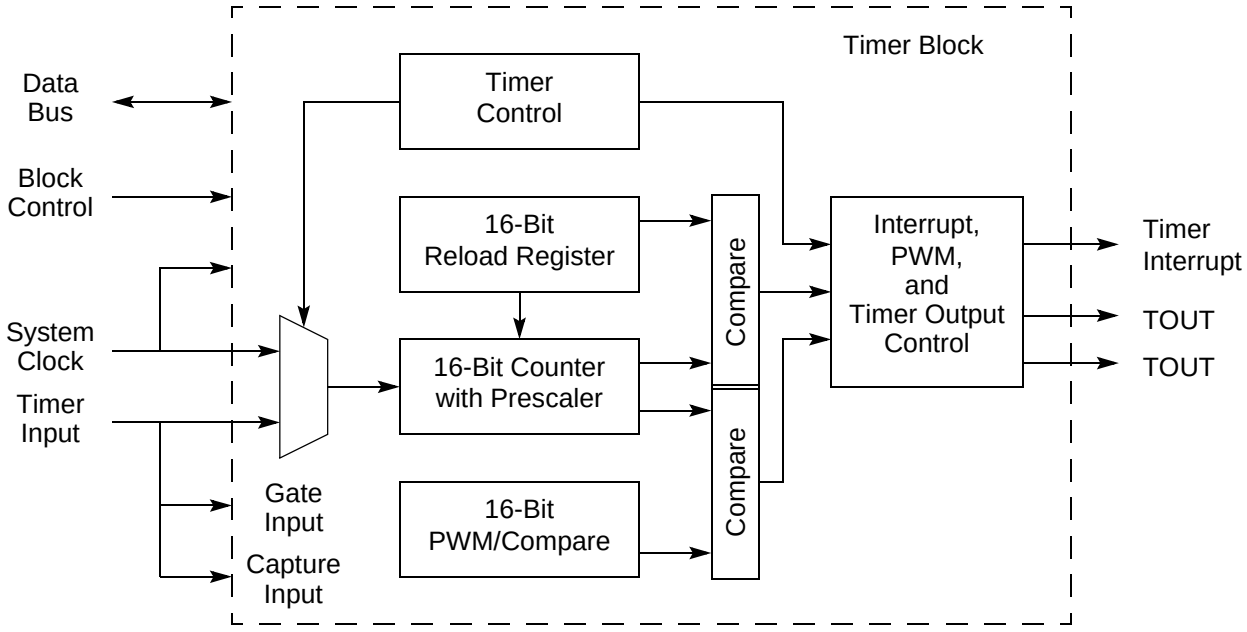


Figure 20. Timer Block Diagram

Operation

The general-purpose timer is a 16-bit up-counter. In normal operation, the timer is initialized to 0001h. When the timer is enabled, it counts up to the value contained in the Reload High and Low Byte registers, then resets to 0001h. The counter either halts or continues depending on the mode.

Minimum time-out delay (1 system clock) 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 ($2^{16} * 2^7$ system clocks) is set by loading the value 0000h into the Timer Reload High and Low byte registers and setting the prescale value to 128. When the timer reaches FFFFh, the timer rolls over to 0000h.

If the reload register is set to a value less than the current counter value, the counter continues counting until it reaches FFFFh and then resets to 0000h. Then the timer continues to count until it reaches the reload value and it resets to 0001h.

Note: When T0IN0, T0IN1 and T0IN2 functions are enabled on the PB0, PB1 and PB2 pins, each Timer 0 input will have the same effect as the single Timer 0 Input pin T0IN. For example, if the Timer 0 is in Capture Mode, any transitions on any of the PB0, PB1 and PB2 pins will cause a Capture.

Timer Operating Modes

The timers are configured to operate in the following modes:

One-Shot Mode

In One-Shot Mode, the timer counts up to the 16-bit reload value stored in the Timer Reload High and Low byte registers. The timer input is the system clock. When the timer reaches the reload value, it generates an interrupt and the count value in the Timer High and Low byte registers is reset to `0001h`. The timer is automatically disabled and stops counting.

If the timer output alternate function is enabled, the timer output pin changes state for one system clock cycle (from Low to High then back to Low if TPOL = 0) at timer Reload. If the timer output is required to make a permanent state change on One-Shot time-out, first set the TPOL bit in the Timer Control 1 Register to the start value before beginning One-Shot Mode. Then, after starting the timer, set TPOL to the opposite value.

Observe the following steps to configure a timer for One-Shot Mode and initiate the count:

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for One-Shot Mode
   - Set the prescale value
   - Set the initial output level (High or Low) using the TPOL bit for the timer output alternate function
   - Set the INTERRUPT Mode
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. Enable the timer interrupt, if required and set the timer interrupt priority by writing to the relevant interrupt registers.
5. When 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.

The timer period is calculated by the following equation (start value = 1):

$$\text{One-Shot Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

TRIGGERED One-Shot Mode

In TRIGGERED One-Shot Mode, the timer operates as follows:

1. The timer is nonactive until a trigger is received. The timer trigger is taken from the timer input pin. The TPOL bit in the Timer Control 1 Register selects whether the trigger occurs on the rising edge or the falling edge of the timer input signal.
2. Following the trigger event, the timer counts system clocks up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers.
3. After reaching the reload value, the timer outputs a pulse on the timer output pin, generates an interrupt and resets the count value in the Timer High and Low Byte registers to `0001h`. The duration of the output pulse is a single system clock. The TPOL bit also sets the polarity of the output pulse.
4. The timer now idles until the next trigger event. Trigger events, which occur while the timer is responding to a previous trigger is ignored.

Observe the following steps to configure timer 0 in TRIGGERED One-Shot Mode and initiate operation:

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for TRIGGERED One-Shot Mode
   - Set the prescale value
   - Set the initial output level (High or Low) via the TPOL bit for the timer output alternate function
   - Set the INTERRUPT Mode
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. Enable the timer interrupt, if required and set the timer interrupt priority by writing to the relevant interrupt registers.
5. When 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. Counting does not start until the appropriate input transition occurs.

The timer period is calculated by the following equation (Start Value = 1):

$$\text{Triggered One-Shot Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

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. After 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 timer output alternate function is enabled, the timer output pin changes state (from Low to High or High to Low) after timer Reload.

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

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for Continuous Mode
   - Set the prescale value
   - Set the initial output level (High or Low) through TPOL for the timer output alternate function
2. Write to the Timer High and Low Byte registers to set the starting count value (usually 0001h). This only affects the first pass in Continuous Mode. After the first timer reload in Continuous Mode, counting always begins at the reset value of 0001h.
3. Write to the Timer Reload High and Low Byte registers to set the Reload period.
4. Enable the timer interrupt, if required and set the timer interrupt priority by writing to the relevant interrupt registers.
5. When 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.

The timer period is calculated by the following equation:

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

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

COUNTER and COMPARATOR COUNTER Modes

In COUNTER Mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the associated GPIO port pin. 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.

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

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

After 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.

If the timer output alternate function is enabled, the timer output pin changes state (from Low to High or High to Low) at timer Reload.

Observe the following steps to configure a timer for COUNTER and COMPARATOR COUNTER modes and initiate the count:

1. Write to the timer control registers to:
   - Disable the timer.
   - Configure the timer for COUNTER or COMPARATOR COUNTER Mode.
   - Select either the rising edge or falling edge of the timer input or comparator output signal for the count. This also sets the initial logic level (High or Low) for the timer output alternate function. However, enabling the timer output function is not required.
2. Write to the Timer Reload High and Low Byte registers to set the starting count value. This affects only the first pass in the COUNTER modes. After the first timer Reload, 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. Enable the timer interrupt, if appropriate 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 (COUNTER Mode).

6. When 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.

PWM SINGLE and DUAL OUTPUT Modes

In PWM SINGLE OUTPUT Mode, the timer outputs a PWM output signal through a GPIO Port pin. In PWM DUAL OUTPUT Mode, the timer outputs a PWM output signal and also its complement through two GPIO port pins. The timer first counts up to the 16-bit PWM match value stored in the timer 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. When it reaches the reload value, the timer generates an interrupt. The count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes.

The timer output signal begins with value = TPOL and then transits to TPOL, when the timer value matches the PWM value. The timer output signal returns to TPOL after the timer reaches the reload value and is reset to `0001h`.

In PWM DUAL OUTPUT Mode, the timer also generates a second PWM output signal, timer output complement ($\overline{\text{TOUT}}$). A programmable deadband is configured (PWMD field) to delay (0 to 128 system clock cycles) the Low to a High (inactive to active) output transitions on these two pins. This configuration ensures a time gap between the deassertion of one PWM output to the assertion of its complement.

Observe the following steps to configure a timer for PWM SINGLE or DUAL OUTPUT Mode and initiate the PWM operation:

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for the selected PWM Mode
   - Set the prescale value
   - Set the initial logic level (High or Low) and PWM High or Low transition for the timer output alternate function with the TPOL bit
   - Set the deadband delay (DUAL OUTPUT Mode) with the PWMD field
2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001h`). The starting count value only affects the first pass in PWM Mode. After the first timer reset in PWM Mode, counting always begins at the reset value of `0001h`.
3. Write to the 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. Enable the timer interrupt, if required and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin(s) 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 determined by the following equation:

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

If an initial starting value other than 0001h is loaded into the Timer High and Low Byte registers, use the One-Shot Mode equation 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 determined by:

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

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

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

Capture Modes

There are three Capture modes which provide slightly different methods for recording the time or time interval between timer input events. These modes are Capture Mode, Capture Restart Mode and Capture Compare Mode. In all of the three modes, when the appropriate timer input transition (capture event) occurs, the timer counter value is captured and stored in the PWM High and Low Byte registers. The TPOL bit in the Timer Control 1 Register determines if the Capture occurs on a rising edge or a falling edge of the timer input signal. The TICONFIG bit determines whether interrupts are generated on capture events, reload events or both. The INCAP bit in Timer Control 0 Register clears to indicate an interrupt caused by a reload event and sets to indicate the timer interrupt is caused by an input capture event.

If the timer output alternate function is enabled, the timer output pin changes state (from Low to High or High to Low) at timer Reload. The initial value is determined by the TPOL bit.

Capture Mode

When the timer is enabled in Capture Mode, it counts continuously and resets to 0000h from FFFFh. When the Capture event occurs, the timer counter value is captured and stored in the PWM High and Low Byte registers, 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. On reaching the reload value, the timer generates an interrupt and continues counting.

Capture Restart Mode

When the timer is enabled in Capture Restart Mode, it counts continuously until the capture event occurs or the timer count reaches the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. If the Capture event occurs first, the timer counter value is captured and stored in the PWM High and Low byte registers, an interrupt is generated and the count value in the Timer High and Low Byte registers is Reset to 0001h and counting resumes. If no Capture event occurs, on reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is Reset to 0001h and counting resumes.

Capture/Compare Mode

The Capture/Compare Mode is identical to Capture Restart Mode except that counting does not start until the first appropriate external timer reload high and low byte input transition occurs. Every subsequent appropriate transition (after the first) of the timer reload high and low byte input signal captures the current count value. When the Capture event occurs, an interrupt is generated, the count value in the Timer Reload High and Low byte High and Low Byte registers is reset to 0001h and counting resumes. If no Capture event occurs, on reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is Reset to 0001h and counting resumes.

Observe the following steps to configure a timer for one of the Capture modes and initiate the count:

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for the selected Capture Mode
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the timer input
   - Configure the timer interrupt to be generated at the input capture event, the reload event or both by setting TICONFIG field
2. Write to the Timer Reload 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. Enable the timer interrupt, if appropriate 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. In Capture and Capture Restart modes, the timer begins counting. In Capture Compare Mode the timer does not start counting until the first appropriate input transition occurs.

In Capture modes, the elapsed time from timer start to Capture event is calculated using the following equation (start value = 1):

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Compare Mode

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

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

Observe the following steps to configure timer for Compare Mode and initiate the count:

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for Compare Mode
   - Set the prescale value
   - Set the initial logic level (High or Low) for the timer output alternate function, if required
2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. Enable the timer interrupt, if appropriate and set the timer interrupt priority by writing to the relevant interrupt registers.
5. When 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.

The compare time is calculated by the following equation (Start Value = 1):

$$\text{Compare Mode Time (s)} = \frac{(\text{Compare Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Gated Mode

In Gated Mode, the timer counts only when the timer input signal is in its active state as determined by the TPOL bit in the Timer Control 1 Register. When the timer input signal is active, counting begins. A timer interrupt is generated when the timer input signal transits from active to inactive state 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. On reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is Reset to `0001h` and counting continues as long as the timer input signal is active. 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 steps to configure a timer for Gated Mode and initiate the count:

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for Gated Mode
   - Set the prescale value
   - Select the active state of the timer input through the TPOL bit
2. Write to the Timer High and Low Byte registers to set the initial count value. This affects only 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. Enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the timer interrupt to be generated only at the input deassertion event, the reload event or both by setting TICONFIG field of the Timer Control 0 Register.
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.
8. The timer counts when the timer input is equal to the TPOL bit.

Reading Timer Count Values

The current count value in the timer is read while counting (enabled). This has no effect on timer operation. Normally, the count must be read with one 16-bit operation. However, 8-bit reads are done with the following method. 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 when enabled. When the timer is not enabled, a read from the timer low byte register returns the actual value in the counter.

The Timers can be cascaded by using the Cascade bit in the Timer control registers. When this bit is set for a Timer, the input source is redefined. When the Cascade bit is set for Timer 0, the input for Timer 0 is the output of the Analog Comparator. When the Cascade bit is set for Timer 1 and Timer 2, the output of Timer 0 and Timer 1 become the input for Timer 1 and Timer 2, respectively. Any Timer Mode can be used. Timer 0 can be cascaded to Timer 1 only by setting the Cascade bit for Timer 1. Timer 1 cascaded to Timer 2 only by setting the Cascade bit for Timer 2. Or all three cascaded, Timer 0 to Timer 1 or Timer 2 for really long counts by setting the Cascade bit for Timer 1 and Timer 2.

Timer Control Register Definitions

Timer 0–2 High and Low Byte Registers

The Timer 0–2 High and Low Byte (TxH and TxL) registers, shown in Tables 56 and 57, contain the current 16-bit timer count value. When the timer is enabled, a read from TxH stores the value in TxL to a temporary holding register. A read from TxL always returns this temporary register when the timer is enabled. When the timer is disabled, reads from the TxL reads the register directly.

Writing to the Timer High and Low Byte registers while the timer is enabled is not recommended. There are no temporary holding registers available for write operations, so simultaneous 16-bit writes are not possible. When either of the timer high or low byte registers are written during counting, the 8-bit written value is placed in the counter (High or Low Byte) at the next clock edge. The counter continues counting from the new value.

Table 56. Timer 0–2 High Byte Register (TxH)

Bits	7	6	5	4	3	2	1	0
Field	TH							
RESET	00h							
R/W	R/W							
Addr	FF–E300h, FF–E310h, FF–E320h							

Bit	Description
[7:0] TH	**Timer High and Low Byte** These two bytes, {TH[7:0], TL[7:0]}, contain the current 16-bit timer count value.

Table 57. Timer 0–2 Low Byte Register (TXL)

Bits	7	6	5	4	3	2	1	0
Field	TL							
RESET	01h							
R/W	R/W							
Addr	FF–E301h, FF–E311h, FF–E321h							

Bit	Description
[7:0] TL	**Timer High and Low Byte** These two bytes, {TH[7:0], TL[7:0]}, contain the current 16-bit timer count value.

Timer X Reload High and Low Byte Registers

The Timer 0–2 Reload High and Low Byte (TxRH and TxRL) registers, shown in Tables 58 and 59, 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.

Table 58. Timer 0–2 Reload High Byte Register (TxRH)

Bits	7	6	5	4	3	2	1	0
Field	TRH							
RESET	FFh							
R/W	R/W							
Addr	FF–E302h, FF–E312h, FF–E322h							

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

Table 59. Timer 0–2 Reload Low Byte Register (TxRL)

Bits	7	6	5	4	3	2	1	0
Field	TRL							
RESET	FF							
R/W	R/W							
Addr	FF–E303h, FF–E313h, FF–E323h							

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

Timer 0–2 PWM High and Low Byte Registers

The Timer 0–2 PWM High and Low Byte (TxPWMH and TxPWML) registers, shown in Tables 60 and 61, define PWM operations and store the timer counter values for the Capture modes. These TxPWMH and TxPWML registers also store the 16-bit captured timer value when operating in Capture or Capture/Compare modes.

Table 60. Timer 0–2 PWM High Byte Register (TxPWMH)

Bits	7	6	5	4	3	2	1	0
Field	PWMH							
RESET	00h							
R/W	R/W							
Addr	FF–E304h, FF–E314h, FF–E324h							

Bit	Description
[7:0] PWMH	**Pulse-Width Modulator High and Low Bytes** These two bytes, {PWMH[7:0], PWML[7:0]}, form a 16-bit value which 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).

Table 61. Timer 0–2 PWM Low Byte Register (TxPWML)

Bits	7	6	5	4	3	2	1	0
Field	PWML							
RESET	00h							
R/W	R/W							
Addr	FF–E305h, FF–E315h, FF–E315h							

Bit	Description
[7:0] PWML	**Pulse-Width Modulator High and Low Bytes** These two bytes, {PWMH[7:0], PWML[7:0]}, form a 16-bit value which 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).

Timer 0–2 Control Registers

Timer 0–2 Control 0 Register

The Timer 0–2 Control 0 Register (TxCTL0), together with the Timer 0–2 Control 1 (TxCTL1) Register, determines timer configuration and operation.

Table 62. Timer 0–2 Control 0 Register (TxCTL0)

Bits	7	6	5	4	3	2	1	0
Field	TMODE[3]	TICONFIG		CASCADE	PWMD			INCAP
RESET	0	00		0	000			0
R/W	R/W	R/W		R/W	R/W			R
Addr	FF–E306h, FF–E316h, FF–E326h							

Bit	Description
[7] TMODE[3]	**Timer Mode High Bit** This bit along with TMODE[2:0] field in T0CTL1 register determines the operating mode of the timer. This bit is the most significant bit of the timer mode selection value. For more details, see the Timer 0–2 Control 1 Register (TxCTL1) section on page 111.
[6:5] TICONFIG	**Timer Interrupt Configuration** This field configures timer interrupt definitions. These bits affect all modes. The effect per mode is explained below: ONE SHOT, Continuous, COUNTER, PWM, Compare, DUAL PWM, TRIGGERED One-Shot, COMPARATOR COUNTER: 0x Timer interrupt occurs on reload. 10 Timer interrupts are disabled. 11 Timer Interrupt occurs on reload. **Gated**: 0x Timer interrupt occurs on reload. 10 Timer interrupt occurs on inactive gate edge. 11 Timer interrupt occurs on reload. **Capture, Capture/Compare, Capture Restart**: 0x Timer interrupt occurs on reload and capture. 10 Timer interrupt occurs on capture only. 11 Timer interrupt occurs on reload only.
[4] CASCADE	**Timer Cascade** This field allows the timers to be cascaded for larger counts. Only Counter Mode must be used with this feature. 0 = The timer is not cascaded. 1 = Timer is cascaded. If timer 0 CASCADE bit is set, ANALOG COMPARATOR output is used as input. If timer 1 CASCADE bit is set, the Timer 0 output is used as the input. If timer 2 CASCADE bit is set, the timer 1 output is used as input.

Bit	Description (Continued)
[3:1] PWMD	**PWM Delay Value** This field is a programmable delay to control the number of additional system clock cycles following a PWM or Reload compare before the timer output or the timer output complement is switched to the active state. This field ensures a time gap between deassertion of one PWM output to the assertion of its complement. No delay. 2 cycles delay. 4 cycles delay. 8 cycles delay. 16 cycles delay. 32 cycles delay. 64 cycles delay. 128 cycles delay.
[0] INCAP	**Input Capture Event** 0 = Previous timer interrupt is not a result of a timer input capture event. 1 = Previous timer interrupt is a result of a timer input capture event.

Timer 0–2 Control 1 Register

The Timer 0–2 Control 1 (TxCTL1) Register enables/disables the timer, sets the prescaler value and determines the timer operating mode.

Table 63. Timer 0–2 Control 1 Register (TxCTL1)

Bits	7	6	5	4	3	2	1	0
Field	TEN	TPOL	PRES			TMODE		
RESET	0	0	000			000		
R/W	R/W	R/W	R/W			R/W		
Addr	FF–E307h, FF–E317h, FF–E327h							

Bit	Description
[7] TEN	0 = Timer is disabled. 1 = Timer is enabled. **Note:** the TEN bit is cleared automatically when the timer stops.

Bit	Description (Continued)
[6] TPOL	**Timer Input/Output Polarity** This bit is a function of the current operating mode of the timer. It determines the polarity of the input and/or output signal. When the timer is disabled, the timer output signal is set to the value of this bit. **One-Shot Mode** If the timer is enabled, the timer output signal pulses (changes state) for one system clock cycle after timer Reload. **Continuous Mode** If the timer is enabled, the timer output signal is complemented after timer Reload. **COUNTER Mode** If the timer is enabled, the timer output signal is complemented after timer reload. 0 = Count occurs on the rising edge of the timer input signal. 1 = Count occurs on the falling edge of the timer input signal. **PWM SINGLE OUTPUT Mode** When enabled, the timer output is forced to $\overline{\text{TPOL}}$ after PWM count match and forced back to TPOL after Reload. **Capture Mode** If the timer is enabled, the timer output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the timer input signal. 1 = Count is captured on the falling edge of the timer input signal. **Compare Mode** The timer output signal is complemented after timer Reload. **Gated Mode** The timer output signal is complemented after timer Reload. 0 = Timer counts when the timer input signal is High and interrupts are generated on the falling edge of the timer input. 1 = Timer counts when the timer input signal is Low and interrupts are generated on the rising edge of the timer input. **Capture/Compare Mode** If the timer is enabled, the timer output signal is complemented after timer Reload. 0 = Counting starts on the first rising edge of the timer Input signal. The current count is captured on subsequent rising edges of the timer input signal. 1 = Counting starts on the first falling edge of the timer input signal. The current count is captured on subsequent falling edges of the timer input signal. **PWM DUAL OUTPUT Mode** If enabled, the timer output is set=$\overline{\text{TPOL}}$ after PWM match and set = TPOL after Reload. If enabled the timer output complement takes on the opposite value of the timer output. The PWMD field in the T0CTL1 Register determines an optional added delay on the assertion (Low to High) transition of both timer output and the timer output complement for deadband generation.

Bit	Description (Continued)
[6] TPOL (cont'd)	**Capture Restart Mode** If the timer is enabled, the timer output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the timer input signal. 1 = Count is captured on the falling edge of the timer input signal. **ANALOG COMPARATOR COUNTER Mode** If the timer is enabled, the timer output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the timer input signal. 1 = Count is captured on the falling edge of the timer input signal. **TRIGGERED One-Shot Mode** If the timer is enabled, the timer output signal is complemented after timer Reload. 0 = The timer triggers on a Low to High transition on the input. 1 = The timer triggers on a High to Low transition on the input.
[5:3] PRES	The timer input clock is divided by 2^{PRES}, where PRES is set from 0 to 7. The prescaler is reset each time the timer is disabled. This ensures proper clock division each time the timer is restarted. 000 = Divide by 1 001 = Divide by 2 010 = Divide by 4 011 = Divide by 8 100 = Divide by 16 101 = Divide by 32 110 = Divide by 64 111 = Divide by 128
[2:0] TMODE[2:0]	This field along with the TMODE[3] bit in T0CTL0 register determines the operating mode of the timer. TMODE[3:0] selects from the following modes: 0000 = One-Shot Mode 0001 = Continuous Mode 0010 = COUNTER Mode 0011 = PWM SINGLE OUTPUT Mode 0100 = Capture Mode 0101 = Compare Mode 0110 = Gated Mode 0111 = Capture/Compare Mode 1000 = PWM DUAL OUTPUT Mode 1001 = Capture Restart Mode 1010 = COMPARATOR COUNTER Mode 1011 = TRIGGERED One-Shot Mode

Multi-Channel PWM Timer

The ZNEO® Z16F Series includes a Multi-Channel PWM optimized for motor control applications. The PWM includes the following features:

- Six independent PWM outputs or three complementary PWM output pairs
- Programmable deadband insertion for complementary output pairs
- Edge-aligned or center-aligned PWM signal generation
- PWM off-state is an option bit programmable
- PWM outputs driven to off-state on System Reset
- Asynchronous disabling of PWM outputs on system fault; outputs are forced to off-state
- Fault inputs generate pulse-by-pulse or hard shutdown
- 12-bit reload counter with 1, 2, 4 or 8 programmable clock prescaler
- High current source and sink on all PWM outputs
- PWM pairs used as general purpose inputs when outputs are disabled
- ADC synchronized with PWM period
- Synchronization for current-sense sample and hold
- Narrow pulse suppression with programmable threshold

Architecture

The PWM unit consists of a master timer to generate the modulator time base and six independent compare registers to set the PWM for each output. The six outputs are designed to provide control signals for inverter drive circuits. The outputs are grouped into pairs consisting of a high-side driver and a low-side driver output. The output pairs are programmable to operate independently or as complementary signals.

In complementary output mode, a programmable dead-time is inserted to ensure nonoverlapping signal transitions. The master count and compare values feed into modulator logic which generates the proper transitions in the output states. Output polarity and fault/off-state control logic allows programming of the default off-states which forces the outputs to a safe state in the event a fault in the motor drive is detected. Figure 21 displays the architecture of the PWM modulator.

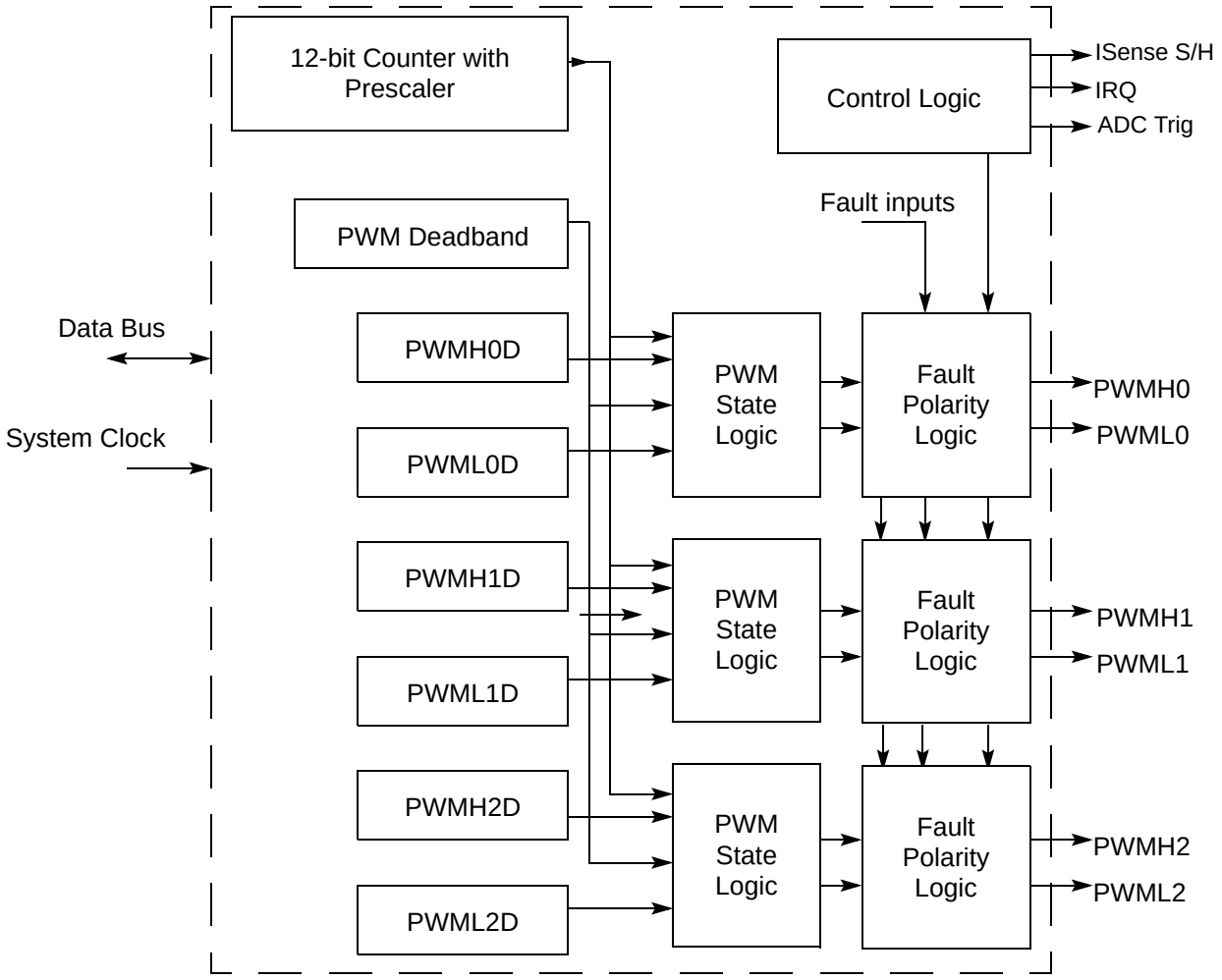


Figure 21. PWM Block Diagram

Operation

PWM Option Bits

To protect the configuration of critical PWM parameters, settings to enable output channels and the default off-state are maintained as user option bits. These values are set when the user program code is written to the part; the software cannot change these values. For more, see the Option Bits chapter on page 292.

PWM Output Polarity and Off-State

The default off-state and polarity of the PWM outputs are controlled by the option bits PWMHI and PWMLO. The PWMHI option controls the off-state and polarity for PWM high-side outputs PWMH0, PWMH1 and PWMH2. The PWMLO option controls the off-state and polarity for low-side outputs PWML0, PWML1 and PWML2.

The off-state is the value programmed in the option bit. For example, programming PWMHI to 1 makes the off-state of PWMH0, PWMH1 and PWMH2 a High logic value and the active state a Low logic value. Conversely, programming PWMHI to 0 causes the off-state to be a Low logic value. PWMLO is programmed in a similar manner.

PWM Enable

The MCEN option bit enables output pairs PWM0, PWM1 and PWM2. If the Motor Control option is not enabled, the PWM outputs remain in a high-impedance state after reset and is used as alternate functions like general purpose input. If the Motor Control option is enabled, following a Power-On Reset (POR) the PWM pins enter a high impedance state. As the internal reset proceeds, the PWM outputs are forced to the off-state as determined by the PWMHI and PWMLO off-state option bits.

PWM Reload Event

To prevent erroneous PWM pulse-widths and periods, registers that control the timing of the output are buffered. Buffering causes all of the PWM compare values to update. In other words, the registers controlling the duty cycle and clock source prescaler only take effect on a PWM reload event. A PWM reload event is configured to occur at the end of each PWM period or only every 2, 4, or 8 PWM periods by setting the RELFREQ bits in the PWM Control 1 Register (PWMCTL1). Software indicates that all new values are ready by setting the READY bit in the PWM Control 0 Register (PWMCTL0) to 1. When the READY bit is set to 1, the buffered values take effect at the next reload event.

PWM Prescaler

The prescaler decreases the PWM clock signal by factors of 1, 2, 4, or 8 with respect to the system clock. The PRES[1:0] bit field in the PWM Control 1 Register (PWMCTL1) controls prescaler operation. This 2-bit PRES field is buffered so that the prescale value only changes on a PWM Reload event.

PWM Period and Count Resolution

The PWM counter operates in two modes to allow edge-aligned and center-aligned outputs. Figures 22 and 23 illustrate edge and center-aligned PWM outputs. The mode in which the PWM operates determine the period of the PWM outputs (PERIOD). The programmed duty-cycle (PWMDC) and the programmed deadband time (PWMDB) deter-

mine the active time of a PWM output. The following sections describe the PWM TIMER modes and the registers controlling the duty-cycle and deadband time.

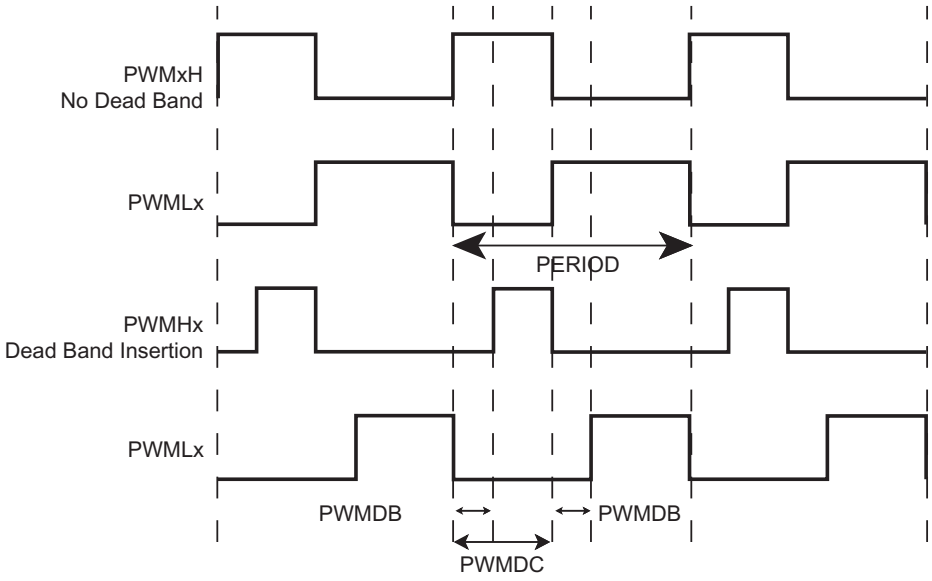


Figure 22. Edge-Aligned PWM Output

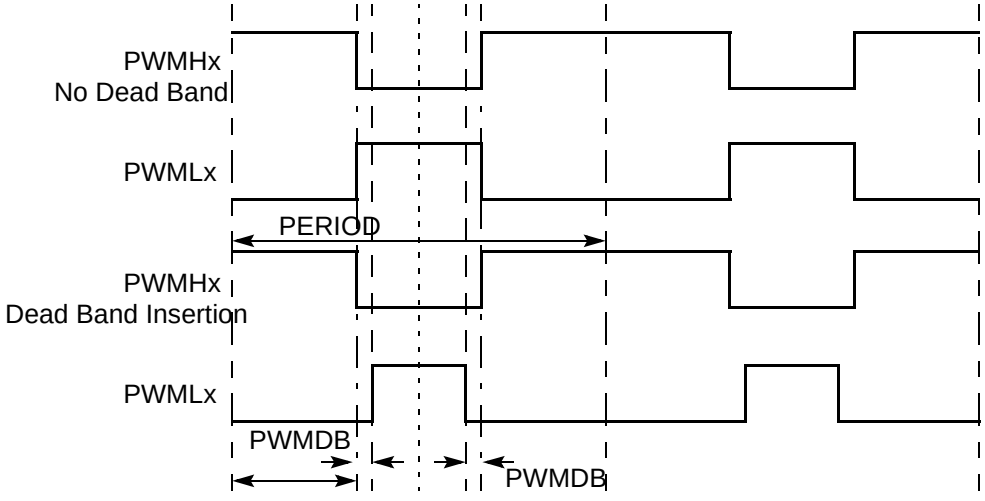


Figure 23. Center-Aligned PWM Output

EDGE-ALIGNED Mode

In EDGE-ALIGNED PWM Mode, a 12-bit up counter creates the PWM period with a minimum resolution equal to the PWM clock source period. The counter counts up to the reload value, resets to 000h and then resumes counting.

$$\text{Edge-Aligned PWM Mode Period} = \frac{\text{Prescaler} \times \text{Reload Value}}{f_{PWMclk}}$$

CENTER-ALIGNED Mode

In CENTER-ALIGNED PWM Mode, a 12-bit up/down counter creates the PWM period with a minimum resolution equal to twice the PWM clock source period. The counter counts up to the reload value and then counts down to 0.

$$\text{Center-Aligned PWM Mode Period} = \frac{2 \times \text{Prescaler} \times \text{Reload Value}}{f_{PWMclk}}$$

PWM Duty Cycle Registers

The PWM duty cycle registers (PWMH0D, PWML0D, PWMH1D, PWML1D, PWMH2D, PWML2D) contain a 16-bit signed value where bit 15 is the sign bit. The duty cycle value is compared to the current 12-bit unsigned PWM count value. If the PWM duty cycle value is set less than or equal to 0, the PWM output is deasserted for full PWM period. If the PWM duty cycle value is set to a value greater than the PWM reload value, the PWM output is asserted for full PWM period.

Independent and Complementary PWM Outputs

The six PWM outputs are configured to operate independently or as three complementary pairs. Operation as six independent PWM channels are enabled by setting the INDEN bit in the PWM Control 1 Register (PWMCTL1). In INDEPENDENT Mode, each PWM output uses its own PWM duty cycle value.

When PWM outputs are configured to operate as three complementary pairs, the PWM duty cycle values PWMH0D, PWMH1D and PWMH2D control the modulator output. In COMPLEMENTARY OUTPUT Mode deadband time is also inserted.

The POLx bits in the PWM Control 1 Register (PWMCTL1) select the relative polarity of the high- and low-side signals. As illustrated in Figures 22 and 23, when the POLx bits are cleared to 0, the PWM high-side output willstart in the on-state and transits to the off-state when the PWM timer count reaches the programmed duty cycle. The low-side PWM value starts in the off-state and transits to the on-state as the PWM timer count reaches the value in the associated duty cycle register. Alternately, setting the POLx causes the high-side output to start in the off-state and the low-side output to start in the on-state.

Manual Off-State Control of PWM Output Channels

Each PWM output is controlled directly by the modulator logic or set to the off-state. To manually set the PWM output to the off-state, set the OUTCTL bit and the associated OUTx bits in the PWM Output Control Register (PWMOUT). Off-state control operates individually by channel. For example, suppressing a single output of pair allows the complementary channel to continue operating. Similarly, if the outputs are operating independently disabling one output channel has no effect on the other PWM outputs.

Deadband Insertion

When the PWM outputs are configured to operate as complementary pairs, an 8-bit deadband value is defined in the PWM Deadband Register (PWMDB). Inserting deadband time causes the modulator to separate the deassertion of one PWM signal from the assertion of its complement. This separation is essential for many motor control applications to prevent simultaneous turn-on of the high-side and low-side drive transistors. The deadband counter directly counts system clock cycles and is unaffected by PWM prescaler settings. The width of this deadband is the number of system clock cycles specified in the PWM Deadband Register (PWMDB).

The minimum deadband duration is zero system clocks; the maximum time is 255 system clocks. Both PWM outputs of a complementary pair is deasserted during the deadband period. Generation of deadband time does not alter the PWM period but the deadband time is subtracted from the active time of the PWM outputs. Figure 22 on page 117 displays the effect of deadband insertion on the PWM output.

Minimum PWM Pulse Width Filter

The PWM modulator is capable of producing pulses as narrow as a single system clock cycle in width. The response time of external drive circuit is slower than the period of a system clock. Therefore, a filter is implemented to enforce a minimum width pulse on the PWM output pins. All output pulses, either High or Low, must be at least the minimum number of PWM clock cycles in width, as specified in the PWM Minimum Pulse Width Filter (PWMMPF) Register (for more details, see the PWM Prescaler section on page 116).

If the expected pulse width is less than the threshold, the associated PWM output does not change state until the duty cycle value has changed sufficiently to allow pulse generation of an acceptable width. The minimum pulse width filter also accounts for the duty cycle variation caused by the deadband insertion. The PWM output pulse is filtered even if the programmed duty cycle is greater than the threshold but the decrease in pulse width because of deadband insertion causes the pulse to be too narrow. The pulse width filter value is calculated as:

$$\text{roundup}(\text{PWMMPF}) = T_{\text{minPulseOut}} / (T_{\text{systemClock}} \cdot \text{PWMprescaler})$$

where *minPulseOut* is the shortest allowed pulse width on the PWM outputs (in seconds).

Synchronization of PWM and ADC

The ADC on the ZNeo is synchronized with the PWM period. Enabling the PWM ADC trigger causes the PWM to generate an ADC conversion signal at the end of each PWM period. Additionally, in CENTER-ALIGNED Mode, the PWM generates a trigger at the center of the period. Setting the ADCTRIG bit in the PWM Control 0 Register (PWMCTL0) enables the ADC synchronization.

Synchronized Current-Sense Sample and Hold

The PWM controls the current-sense input sample and hold amplifier. The signal controlling the sample/hold is configured to always sample or automatically hold when any or all of the PWM High or Low outputs are in the on state. The current-sense sample and hold is controlled by the Current-Sense Sample and Hold Control Register (CSSHR0 and CSSHR1).

PWM Timer and Fault Interrupts

The PWM generates interrupts to the ZNEO CPU during any of the following events:

- PWM Reload, in which the interrupt is generated at the end of a PWM period when a PWM register reload occurs
- PWM Fault, in which a fault condition is indicated by asserting any FAULT pins or by the assertion of the comparator

Fault Detection and Protection

The ZNEO contains hardware and software fault controls, which allow rapid deassertion of all enabled PWM output signals. A logic Low on an external fault pin ($\overline{\text{FAULT0}}$ or FAULT1) or the assertion of the over current comparator forces the PWM outputs to the predefined off-state.

Similar deassertion of the PWM outputs is accomplished in software by writing to the PWMOFF bit in the PWM Control 0 Register. The PWM counter continues to operate while the outputs are deasserted (inactive) due to one of these fault conditions.

The fault inputs are individually enabled through the PWM Fault Control Register. If a fault condition is detected and the source is enabled, the fault interrupt is generated. The

PWM Fault Status Register (PWMFSTAT) is read to determine which fault source caused the interrupt.

When a fault is detected and the PWM outputs are disabled, modulator control of the PWM outputs are reenabled either by the software or by the fault input signal deasserting. Selection of the reenable method is made using the PWM Fault Control Register (PWMFCTL). Configuration of the fault modes and reenable methods allow pulse-by-pulse limiting and hard shutdown. When configured in Automatic Restart Mode, the PWM outputs are reengaged at beginning of the next PWM cycle (master timer value is equal to 0) if all fault signals are deasserted. In software controlled restart, all fault inputs must be deasserted and the fault flags must be cleared.

The fault input pin is Schmitt-triggered. The input signal from the pin as well as the comparators pass though an analog filter to reject high-frequency noise.

The logic path from the fault sources to the PWM output is asynchronous ensuring that the fault inputs forces the PWM outputs to their off-state even if the system clock is stopped.

PWM Operation in CPU Halt Mode

When the ZNEO CPU is operating in Halt Mode, the PWM continues to operate if it is enabled. To minimize current in Halt Mode, the PWM must be disabled by clearing the PWMEN bit to 0.

PWM Operation in CPU Stop Mode

When the ZNEO CPU is operating in Stop Mode, the PWM is disabled as the system clock ceases to operate in Stop Mode. The PWM output remains in the same state as they were prior to entering the Stop Mode. In normal operation, the PWM outputs must be disabled by software prior to the CPU entering the Stop Mode. A fault condition detected in Stop Mode forces the PWM outputs to the predefined off-state.

Observing the State of PWM Output Channels

The logic value of the PWM outputs is sampled by reading the PWMIN Register. If a PWM channel pair is disabled (option bit is not set), the associated PWM outputs are forced to high impedance and are used as general purpose inputs.

PWM Control Register Definitions

The following sections describe the various PWM control registers.

PWM High and Low Byte Registers

The PWM High and Low Byte (PWMH and PWML) registers, shown in Table 64 and Table 65) contain the current 12-bit PWM count value. Reads from PWMH stores the value in PWML to a temporary holding register. A read from PWML always returns this temporary register value. It is not recommended to write to the PWM High and Low Byte registers when the PWM is enabled. There are no temporary holding registers for write operations, so simultaneous 12-bit writes are not possible. When either the PWM High and Low Byte registers are written during counting, the 8-bit written value is placed in the counter (High or Low Byte) at the next clock edge. The counter continues counting from the new value.

Table 64. PWM High Byte Register (PWMH)

Bits	7	6	5	4	3	2	1	0
Field	Reserved				PWMH			
RESET	0h				0h			
R/W	R/W				R/W			
Addr	FF_E38Ch							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3:0] PWMH	**PWM High and Low Bytes** The upper byte of two bytes {PWMH[3:0], PWML[7:0]} that contain the current 12-bit PWM count value.

Table 65. PWM Low Byte Register (PWML)

Bits	7	6	5	4	3	2	1	0
Field	PWML							
RESET	01h							
R/W	R/W							
Addr	FF_E38DH							

Bit	Description
[7:0] PWML	**PWM High and Low Bytes** The lower byte of two bytes {PWMH[3:0], PWML[7:0]} that contain the current 12-bit PWM count value.

PWM Reload High and Low Byte Registers

The PWM Reload High and Low Byte (PWMRH and PWMRL) registers, shown in Tables 66 and 67, store a 12-bit reload value, {PWMRH[3:0], PWMRL[7:0]}. The PWM reload value is held in buffer registers. The PWM reload value written to the buffer registers are not used by the PWM generator until the next PWM reload event occurs. Reads from these registers always return the values from the buffer registers.

$$\text{Edge-Aligned PWM Mode Period} = \frac{\text{Prescaler} \times \text{Reload Value}}{f_{PWMclk}}$$

$$\text{Center-Aligned PWM Mode Period} = \frac{2 \times \text{Prescaler} \times \text{Reload Value}}{f_{PWMclk}}$$

Table 66. PWM Reload High Byte Register (PWMRH)

Bits	7	6	5	4	3	2	1	0
Field	Reserved				PWMRH			
RESET	0h				Fh			
R/W	R/W				R/W			
Addr	FF_E38EH							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3:0] PWMRH	**PWM Reload Register High and Low** These two bytes form the 12-bit reload value, {PWMRH[3:0], PWMRL[7:0]}. This value sets the PWM period.

Table 67. PWM Reload Low Byte Register (PWMRL)

Bits	7	6	5	4	3	2	1	0
Field	PWMRL							
RESET	FF							
R/W	R/W							
Addr	FF_E38Fh							

Bit	Description
[7:0] PWMRL	**PWM Reload Register High and Low** These two bytes form the 12-bit reload value, {PWMRH[3:0], PWMRL[7:0]}. This value sets the PWM period.

PWM 0–2 Duty Cycle High and Low Byte Registers

The PWM 0–2 H/L (High Side/Low Side) Duty Cycle High and Low Byte (PWM*x*DH and PWM*x*DL) registers, shown in Tables 68 and 69, set the duty cycle of the PWM signal. This 14-bit signed value is compared to the PWM count value to determine the PWM output. Reads from these registers always return the values from the temporary holding registers. The PWM generator does not use the PWM duty cycle value until the next PWM reload event occurs.

$$\text{PWM Duty Cycle} = 100 \times \frac{\text{PWM Duty Cycle Value}}{\text{PWM Reload Value}}$$

Writing a negative value (DUTYH[7] = 1) forces the PWM to be OFF for the full PWM period. Writing a positive value greater than the 12-bit PWM reload value forces the PWM to be ON for the full PWM period.

Table 68. PWM 0–2 H/L Duty Cycle High Byte Register (PWMH*x*DH, PWML*x*DH)

Bits	7	6	5	4	3	2	1	0
Field	SIGN	Reserved		DUTYH				
RESET	X	XX		X_XXXX				
R/W	R/W	R/W		R/W				
Addr	FF_E390h, FF_E392h, FF_E394h, FF_E396h, FF_E398h, FF_E39Ah							

Bit	Description
[7] SIGN	**Duty Cycle Sign** 0 = Duty cycle is a positive two's complement number. 1 = Duty cycle is a negative two's complement number. Output is forced to the off-state.
[6:5]	**Reserved** These bits are reserved and must be programmed to 00.
[4:0] DUTYH	**PWM Duty Cycle High and Low Bytes** The upper byte of two bytes {DUTYH[7:0], DUTYL[7:0]} that form a 14-bit signed value; bits 5 and 6 of the High byte are always 0. The value is compared to the current 12-bit PWM count.

Table 69. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMH*x*DL, PWML*x*DL)

Bits	7	6	5	4	3	2	1	0
Field	DUTYL							
RESET	XXH							
R/W	R/W							
Addr	FF_E391h, FF_E393h, FF_E395h, FF_E397h, FF_E399h, FF_E39Bh							

Bit	Description
[7:0] DUTYL	**PWM Duty Cycle High and Low Bytes** The lower byte of two bytes {DUTYH[7:0], DUTYL[7:0]} that form a 14-bit signed value; bits 5 and 6 of the High byte are always 0. The value is compared to the current 12-bit PWM count.

PWM Control 0 Register

The PWM Control 0 Register (PWMCTL0) controls PWM operation.

Table 70. PWM Control 0 Register (PWMCTL0)

Bits	7	6	5	4	3	2	1	0
Field	PWMOFF	OUTCTL	ALIGN	Reserved	ADCTRIG	Reserved	READY	PWMEN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E380h							

Bit	Description
[7] PWMOFF	**Place PWM Outputs in Off State** 0 = Disable modulator control of PWM pins. Outputs are in predefined off state; not dependent on the Reload event. 1 = Reenable modulator control of PWM pins at next PWM Reload event.
[6] OUTCTL	**PWM Output Control** 0 = PWM outputs are controlled by the pulse-width modulator. 1 = PWM outputs selectively disabled (set to off-state) according to values in the OUT*x* bits of the PWMOUT Register.
[5] ALIGN	**PWM Edge Alignment** 0 = PWM outputs are edge aligned. 1 = PWM outputs are center aligned.
[4]	**Reserved** This bit is reserved and must be programmed to 0.

Bit	Description (Continued)
[3] ADCTRIG	**ADC Trigger Enable** 0 = No ADC trigger pulses. 1 = ADC trigger enabled.
[2]	**Reserved** This bit is reserved and must be programmed to 0.
[1] READY	**Values Ready for Next Reload Event** 0 = PWM values (prescale, period and duty cycle) are not ready. Do not use values in holding registers at next PWM reload event. 1 = PWM values (prescale, period and duty cycle) are ready. Transfer all values from temporary holding registers to working registers at next PWM reload event.
[0] PWMEN	**PWM Enable** 0 = PWM is disabled and enabled PWM output pins are forced to default off-state. PWM master counter is stopped. Certain control registers may only be written in this state. 1 = PWM is enabled and PWM output pins are enabled as outputs.

PWM Control 1 Register

The PWM Control 1 (PWMCTL1) Register controls portions of PWM operation.

Table 71. PWM Control 1 Register (PWMCTL1)

Bits	7	6	5	4	3	2	1	0
Field	RLFREQ[1:0]		INDEN	POL45	POL23	POL10	PRES[1:0]	
RESET	00		0	0	0	0	00	
R/W	R/W		R/W	R/W	R/W	R/W	R/W	
Addr	FF_E381h							

Bit	Description
[7:6] RLFREQ[1:0]	**Reload Event Frequency** This bit field is buffered. Changes to the reload event frequency takes effect at the end of the current PWM period. Reads always return the bit values from the temporary holding register. 00 = PWM reload event occurs at the end of every PWM period. 01 = PWM reload event occurs once every two PWM periods. 10 = PWM reload event occurs once every four PWM periods. 11 = PWM reload event occurs once every eight PWM periods.
[5] INDEN	**Independent PWM Mode Enable** 0 = PWM outputs operate as three complementary pairs. 1 = PWM outputs operate as six independent channels.

Bit	Description (Continued)
[4] POL2	**PWM2 Polarity** 0 = Noninverted polarity for channel pair PWM2. 1 = Invert output polarity for channel pair PWM2.
[3] POL1	**PWM1 Polarity** 0 = Noninverted polarity for channel pair PWM1. 1 = Invert output polarity for channel pair PWM1.
[2] POL0	**PWM0 Polarity** 0 = Noninverted polarity for channel pair PWM0. 1 = Invert output polarity for channel pair PWM0.
[1:0] PRES	**PWM Prescaler** The prescaler divides down the PWM input clock (either the system clock or the PWMIN external input). This field is buffered. Changes to this field take effect at the next PWM reload event. Reads always return the values from the temporary holding register. 00 = Divide by 1. 01 = Divide by 2. 10 = Divide by 4. 11 = Divide by 8.

PWM Deadband Register

The PWM Deadband (PWMDB) Register, shown in Table 72, stores the 8-bit PWM deadband value. The deadband value determines the number of PWM input cycles to use for the deadband time for complementary PWM output pairs. When counting PWM input cycles, the PWM input signal is used directly (no prescaler). The minimum deadband value is 1. Maximum deadband time is programmed by setting a value of 00h.

Table 72. PWM Deadband Register (PWMDB)

Bits	7	6	5	4	3	2	1	0
Field	PWMDB[7:0]							
RESET	01h							
R/W	R/W							
Addr	FF_E382h							

Bit	Description
[7:0] PWMDB[7:0]	**PWM Deadband** Sets the PWM deadband period for which both PWM outputs of a complementary PWM output pair are deasserted.

Note: This register can only be written when PWMEN is cleared.

PWM Minimum Pulse Width Filter

The value in the PWMMPF Register, shown in Table 73, determines the minimum width pulse, either High or Low, generated by the PWM module. The minimum pulse width period is calculated as:

$$T_{minPulseOut} = \frac{PWMDB + PWMMPF}{T_{systemClock} \cdot PwmPrescale}$$

Table 73. PWM Minimum Pulse Width Filter (PWMMPF)

Bits	7	6	5	4	3	2	1	0
Field	PWMMPF[7:0]							
RESET	00h							
R/W	R/W							
Addr	FF_E383h							

Bit	Description
[7:0] PWMDB[7:0]	**Minimum Pulse Filter** Sets the minimum allowed output pulse width in PWM clock cycles.

Note: This register can only be written when PWMEN is cleared.

PWM Fault Mask Register

The PWM Fault Mask Register, shown in Table 74, enables individual fault sources. When an input is asserted, PWM behavior is determined by the PWM Fault Control Register (PWMFCTL). The Comparator 0–3 outputs generate PWM faults and the associated fault system exception. The bits in this register can only be set; all other writes are ignored.

Table 74. PWM Fault Mask Register (PWMFM)

Bits	7	6	5	4	3	2	1	0
Field	Reserved		DBGMSK	Reserved		F1MASK	C0MASK	FMASK
RESET	00		0	000		0	0	0
R/W	R		R/W	R		R/W	R/W	R/W
Addr	FF_E384h							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5] DBGMSK	**Debug Entry Fault Mask** 0 = Entering CPU DEBUG Mode generates a PWM Fault. 1 = Entering CPU DEBUG Mode does not generate a PWM Fault.
[4:3]	**Reserved** These bits are reserved and must be programmed to 00.
[2] F1MASK	**Fault 1 Fault Mask** 0 = Fault 1 generates a PWM Fault. 1 = Fault 1 does not generate a PWM Fault.
[1] C0MASK	**Comparator Fault Mask** 0 = Comparator generates a PWM Fault. 1 = Comparator does not generate a PWM Fault.
[0] F0MASK	**Fault Pin Mask** 0 = Fault 0 pin generates a PWM Fault. 1 = Fault 0 pin does not generate a PWM Fault.

Note: This register can only be written when PWMEN is cleared.

PWM Fault Status Register

The PWM Fault Status (PWMFSTAT) Register, shown in Table 75, provides the status of fault inputs and timer reloads. The fault flags indicate the fault source, which is active. If a fault source is masked, the flag in this register is not set when the source is asserted. The reload flag is set when the timer compare values are updated. Clear flags by writing a 1 to the flag bits. Fault flag bits are cleared only if the associated fault source has deasserted.

Table 75. PWM Fault Status Register (PWMFSTAT)

Bits	7	6	5	4	3	2	1	0
Field	RLDFlag	Reserved	DBGFLAG	Reserved		F1FLAG	C0FLAG	FFLAG
RESET	U	0	U	00		U	U	U
R/W	R/W1C*	R	R/W1C*	R		R/W1C*	R/W1C*	R/W1C*
Addr	FF_E385h							

Bit	Description
[7] RLDFlag	**Reload Flag** This bit is set and latched when a PWM timer reload occurs. Writing a 1 to this bit clears the flag.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] DBGFLAG	**Debug Flag** This bit is set and latched when DEBUG Mode is entered. Writing a 1 to this bit clears the flag.
[4:3] Reserved	**Reserved** These bits are reserved and must be programmed to 00.
[2] F1FLAG	**Fault 1 Flag** This bit is set and latched when fault1 is asserted. Writing a 1 to this bit clears the flag.
[1] C0FLAG	**Comparator 0 Flag** This bit is set and latched when comparator is asserted. Writing a 1 to this bit clears the flag.
[0] FFLAG	**Fault Flag** This bit is set and latched when the fault0 input is asserted. Writing a 1 to this bit clears the flag.

Note: *For this register, W1C means you must write a 1 to clear the flag.

PWM Fault Control Register

The PWM Fault Control (PWMFCTL) Register, shown in Table 76, determines how the PWM recovers from a fault condition. Settings in this register select either an automatic or a software-controlled PWM restart.

Table 76. PWM Fault Control Register (PWMFCTL)

Bits	7	6	5	4	3	2	1	0
Field	Reserved	DBGRST	CMP1INT	CMP1RST	CMPINT	CMPRST	Fault0INT	Fault0RST
RESET	0	0	0	0	0	0		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E388h							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] DBGRST	**Debug Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasstered and a new PWM period begins. 1 = Software controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs.
[5] CMP1INT	**Comparator 1 Interrupt** 0 = Interrupt on comparator assertion disabled. 1 = Interrupt on comparator assertion enabled.
[4] CMP1RST	**Comparator 1 Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasstered. **Software Controlled Recovery** 1 = PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs.
[3 CMP0INT	**Comparator 0 Interrupt** 0 = Interrupt on comparator 0 assertion disabled. 1 = Interrupt on comparator 0 assertion enabled.
[2] CMP0RST	**Comparator 0 Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasstered. **Software Controlled Recovery** 1 = PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs.

Note: This register can only be written when PWMEN is cleared.

Bit	Description (Continued)
[1] Fault0INT	**Fault 0 Interrupt** 0 = Interrupt on fault 0 pin assertion disabled. 1 = Interrupt on Fault0 pin assertion enabled.
[0] Fault0RST	**Fault 0 Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasstered. **Software Controlled Recovery** 1 = PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs.

Note: This register can only be written when PWMEN is cleared.

PWM Input Sample Register

The PWM pin value is sampled by reading the PWM Input Sample Register, shown in Table 77.

Table 77. PWM Input Sample Register (PWMIN)

Bits	7	6	5	4	3	2	1	0
Field	Reserved	FAULT	IN2L	IN2H	IN1L	IN1H	IN0L	IN0H
RESET	0	0	0	0	0	0	0	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E386h							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] FAULT	**Sample Fault0 pin** 0 = A Low-level signal was read on the fault pin. 1 = A High-level signal was read on the fault pin.
[5:0] IN2L/IN2H/ IN1L/IN1H/ IN0L/IN0H	**Sample PWM Pins** 0 = A Low-level signal was read on the pins. 1 = A High-level signal was read on the pins.

PWM Output Control Register

The PWM Output Control (PWMOUT) Register, shown in Table 78, enables modulator control of the six PWM output signals. Output control is enabled by the OUTCTL bit in the PWMCTL0 Register. The PWM continues to operate but has no effect on the disabled PWM pins. If a fault condition is detected, all PWM outputs are forced to their selected off states.

Table 78. PWM Output Control Register (PWMOUT)

Bits	7	6	5	4	3	2	1	0
Field	Reserved	Reserved	OUT2L	OUT2H	OUT1L	OUT1H	OUT0L	OUT0H
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E387h							

Bit	Description
[7,6]	**Reserved** These bits are reserved and must be programmed to 00.
[5,3,1] OUT2L/ OUT1L/ OUT0L	**PWM 2L/1L/0L Output Configuration** 0 = PWM 2L/1L/0L output signal is enabled and controlled by PWM. 1 = PWM 2L/1L/0L output signal is in low-side off-state.
[4,2,0] OUT2H/ OUT1H/ OUT0H	**PWM 2H/1H/0H Output Configuration** 0 = PWM 2H/1H/0H output signal is enabled and controlled by PWM. 1 = PWM 2H/1H/0H output signal is in high-side off-state.

Current-Sense Sample and Hold Control Registers

The Current-Sense Sample/Hold Control Register, shown in Table 79, defines the behavior of the dedicated current sense sample and hold inputs to the ADC from the operational amplifier. These input hold the current input value whenever all high-side outputs or all low-side outputs are in the on-state. The register bits control which PWM outputs must be asserted to activate the internal hold signal. Disabling the HEN, LEN, NHEN and NLEN bits allows software control of the input sample/hold by writing the SHPOL bit.

Table 79. Current-Sense Sample and Hold Control Register (CSSHR0 and CSSHR1)

Bits	7	6	5	4	3	2	1	0
Field	SHPOL	HEN	NHEN	LEN	NLEN	SHPWM2	SHPWM1	SHPWM0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E38Ah and FF_E38Bh							

Bit	Description
[7] SHPOL	**Sample Hold Polarity** 0 = Hold when terms are active. 1 = Hold when terms are not active.
[6] HEN	**High Side Active Enable** 0 = Ignore Product of PWMH0, PWMH1, PWMH2 in sample/hold equation. 1 = Hold when PWMH0, PWMH1, PWMH2 are all active.
[5] NHEN	High Side Inactive Enable 0 = Ignore product of $\overline{\text{PWMH0}}$, $\overline{\text{PWMH1}}$, $\overline{\text{PWMH2}}$ in sample/hold equation. 1 = Hold when are all active.
[4] LEN	**Low Side Active Enable** 0 = Ignore product of PWML0, PWML1, PWML2 in sample/hold equation. 1 = Hold when PWML0, PWML1, PWML2 are all active.
[3] NLEN	**Low Side Inactive Enable** 0 = Ignore product of $\overline{\text{PWML0}}$, $\overline{\text{PWML1}}$, $\overline{\text{PWML2}}$ in sample/hold equation. 1 = Hold when $\overline{\text{PWML0}}$, $\overline{\text{PWML1}}$, $\overline{\text{PWML2}}$ are all active.
[2] SHPWM2	**PWM channel2 Sample/Hold Enable** 0 = Channel 2 terms are not used in sample/hold equation. 1 = Channel 2 terms are used in sample/hold equation.
[1] SHPWM1	**PWM Channel1 Sample/Hold Equation** 0 = Channel 1 terms are not used in sample/hold equation. 1 = Channel 1 terms are used in sample/hold equation.
[0] SHPWM0	**PWM Channel0 Sample/Hold Equation** 0 = Channel 0 terms are not used in sample/hold equation. 1 = Channel 0 terms are used in sample/hold equation.

LIN-UART

The Local Interconnect Network Universal Asynchronous Receiver/Transmitter (LIN-UART) is a full-duplex communication channel capable of handling asynchronous data transfers in standard UART applications as well as providing LIN protocol support.

Features of the LIN-UART include:

- 8-bit asynchronous data transfer
- Selectable even and odd-parity generation and checking
- Option of one or two stop bits
- Selectable MULTIPROCESSOR (9-Bit) Mode with three configurable interrupt schemes
- Separate transmit and receive interrupts or DMA requests
- Framing, parity, overrun and break detection
- 16-bit Baud Rate Generator (BRG), which functions as a general-purpose timer with interrupt
- Driver enable output for external bus transceivers
- LIN protocol support for both Master and Slave modes:
  - Break generation and detection
  - Selectable slave autobaud
  - Check Tx versus Rx data when sending
- Configurable digital noise filter on receive data line

Architecture

The LIN-UART consists of three primary functional blocks: transmitter, receiver and BRG. The LIN-UART's transmitter and receiver function independently but use the same baud rate and data format. The basic UART operation is enhanced by the noise filter and IrDA blocks. Figure 24 displays the LIN-UART architecture.

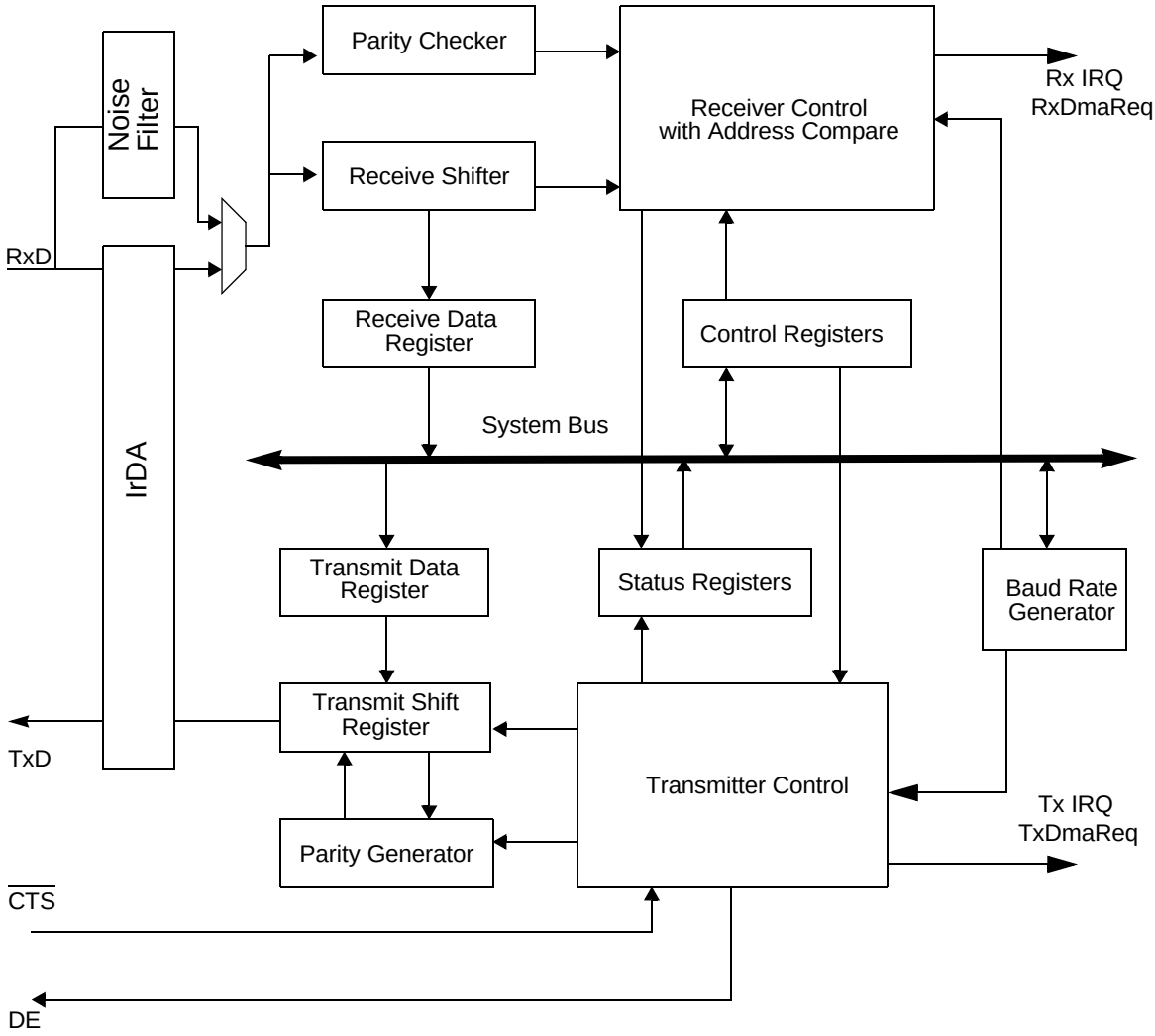


Figure 24. LIN-UART Block Diagram

Operation

Data Format for Standard UART Modes

The LIN-UART always transmits and receives data in an 8-bit data format, with the first bit being the least-significant bit. An even- or odd-parity bit or multiprocessor address/ data bit is optionally added to the data stream. Each character begins with an active Low start bit and ends with either 1 or 2 active High stop bits. Figures 25 and 26 display the

asynchronous data format employed by the LIN-UART, without parity and with parity, respectively.

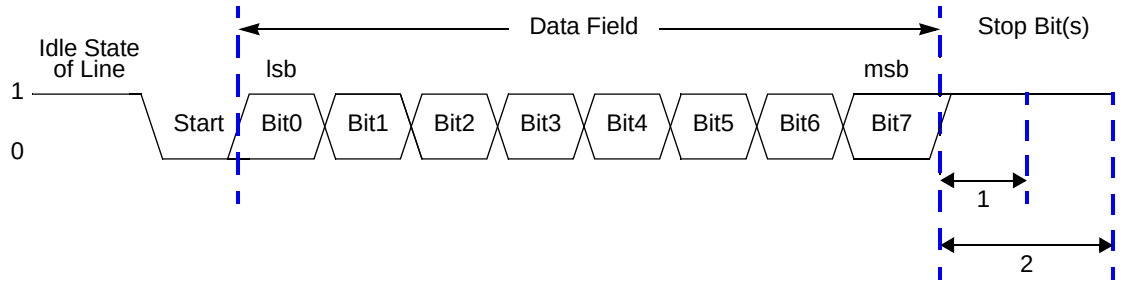


Figure 25. LIN-UART Asynchronous Data Format without Parity

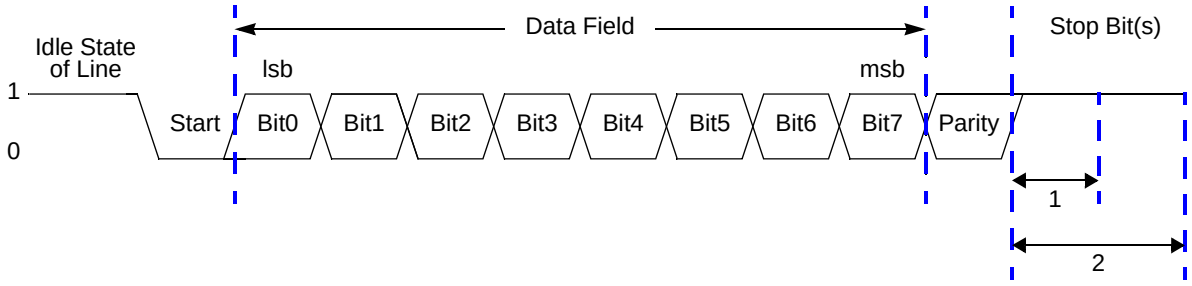


Figure 26. LIN-UART Asynchronous Data Format with Parity

Transmitting Data using the Polled Method

Observe the following steps to transmit data using the polled operating method:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. If Multiprocessor Mode is required, write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-Bit) Mode functions by setting the Multiprocessor Mode select bit (MPEN) to enable Multiprocessor Mode.
4. Write to the LIN-UART Control 0 Register to:
   a. Set the transmit enable bit (TEN) to enable the LIN-UART for data transmission.

b. If parity is required and Multiprocessor Mode is not enabled, set the parity enable bit (PEN) and select either even- or odd parity (PSEL).

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

5. Check the TDRE bit in the LIN-UART Status 0 Register to determine if the Transmit Data Register is empty (indicated by a 1). If this register is 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. If in Multiprocessor Mode, write the LIN-UART Control 1 Register to select the outgoing address bit by setting the multiprocessor bit transmitter (MPBT) if sending an address byte; clear it if sending a data byte.
7. Write the data byte to the LIN-UART Transmit Data Register. The transmitter automatically transfers the data to the transmit shift register and transmits the data.
8. If Multiprocessor Mode is required and Multiprocessor Mode is enabled, make any changes to the multiprocessor bit transmitter (MPBT) value.
9. To transmit additional bytes, return to Step 4.

Transmitting Data Using Interrupt-Driven Method

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

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Execute a DI instruction to disable interrupts.
4. Write to the interrupt control registers to enable the LIN-UART transmitter interrupt and set the appropriate priority.
5. If Multiprocessor Mode is required, write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-Bit) Mode functions by setting the Multiprocessor Mode select bit (MPEN) to enable Multiprocessor Mode.
6. Write to the LIN-UART Control 0 Register to:
   a. Set the transmit enable bit (TEN) to enable the LIN-UART for data transmission
   b. Enable parity, if Multiprocessor Mode is not enabled and select either even or odd parity.

c. Set or clear the CTSE bit to enable or disable control from the remote receiver through the $\overline{CTS}$ pin.

7. Execute an EI instruction to enable interrupts.

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

1. If operating in Multiprocessor Mode, write the LIN-UART Control 1 Register to select the outgoing address bit by setting the multiprocessor bit transmitter (MPBT) if sending an address byte; clear it if sending a data byte.
2. Write the data byte to the LIN-UART Transmit Data Register. The transmitter automatically transfers the data to the transmit shift register and transmits the data.
3. Execute the IRET instruction to return from the interrupt service routine and waits for the Transmit Data Register to again become empty.

If a transmit interrupt occurs and there is no transmit data ready to send, the interrupt service routine executes the IRET instruction. When the application contains data to transmit, software sets the appropriate interrupt request bit in the interrupt controller to initiate a new transmit interrupt. Another alternative would be for software to write the data to the Transmit Data Register instead of invoking the ISR.

Receiving Data Using the Polled Method

Observe the following steps to configure the LIN-UART for polled data reception:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Write to the LIN-UART Control 1 Register to enable Multiprocessor Mode functions.
4. Write to the LIN-UART Control 0 Register to:
   a. Set the receive enable bit (REN) to enable the LIN-UART for data reception.
   b. Enable parity, if Multiprocessor Mode is not enabled and select either even or odd parity.
5. Check the RDA bit in the LIN-UART Status 0 Register to determine if the Receive Data Register contains a valid data byte (indicated by a 1). If RDA is set to 1 to indicate available data, continue to Step 6. If the Receive Data Register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.

6. Read data from the LIN-UART Receive Data Register. If operating in MULTIPROCESSOR (9-Bit) Mode, further actions are required depending on the Multiprocessor Mode bits MPMD[1:0].
7. Return to Step 5 to receive additional data.

Receiving Data Using the Interrupt-Driven Method

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

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Execute a DI instruction to disable interrupts.
4. Write to the interrupt control registers to enable the LIN-UART receiver interrupt and set the appropriate priority.
5. Clear the LIN-UART receiver interrupt in the applicable interrupt request register.
6. Write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-Bit) Mode functions:
   a. Set the Multiprocessor Mode select (MPEN) to enable Multiprocessor Mode.
   b. Set the Multiprocessor Mode bits, MPMD[1:0], to select the appropriate address matching scheme.
   c. Configure the LIN-UART to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for ZNEO devices without a DMA block).
7. Write the device address to the Address Compare Register (automatic multiprocessor modes only).
8. Write to the LIN-UART Control 0 Register to:
   a. Set the receive enable bit (REN) to enable the LIN-UART for data reception
   b. Enable parity, if Multiprocessor Mode is not enabled and select either even- or odd-parity.
9. Execute an EI instruction to enable interrupts.

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

1. Check the LIN-UART Status 0 Register to determine whether the source of the interrupt is error, break, or received data.
2. If the interrupt was due to data available, read the data from the LIN-UART Receive Data Register. If operating in MULTIPROCESSOR (9-Bit) Mode, further actions are required depending on the Multiprocessor Mode bits MPMD[1:0].
3. Execute the IRET instruction to return from the ISR and await more data.

Clear To Send Operation

The clear to send ($\overline{\text{CTS}}$) pin, if enabled by the CTSE bit of the LIN-UART Control 0 Register, performs flow control on the outgoing transmit data stream. The $\overline{\text{CTS}}$ 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 operation is typically performed during the Stop bit transmission. If $\overline{\text{CTS}}$ deasserts in the middle of a character transmission, the current character is sent completely.

External Driver Enable

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

Driver Enable is a programmable polarity signal which envelopes the entire transmitted data frame including parity and stop bits as illustrated in Figure 27. The DE signal asserts when a byte is written to the LIN-UART Transmit Data Register. The DE signal asserts at least one bit period and no greater than two bit periods before the Start bit is transmitted. This allows a set-up time to enable the transceiver. The DE 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 LIN-UART Control Register 1 sets the polarity of the DE signal.

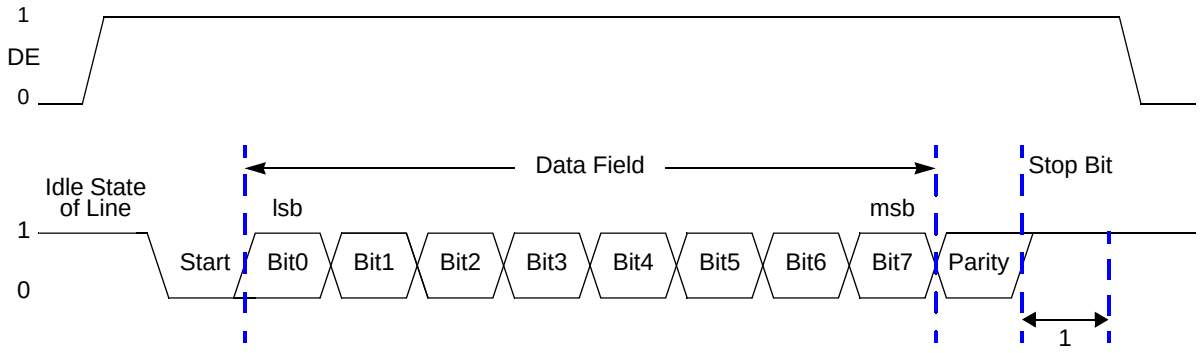


Figure 27. LIN-UART Driver Enable Signal Timing (shown with 1 Stop Bit and Parity)

The DE to Start bit setup time is calculated as follows:

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

LIN-UART Special Modes

The LIN-UART features the following two special modes:

- Multiprocessor Mode
- LIN Mode

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

Multiprocessor (9-Bit) Mode

The LIN-UART features a MULTIPROCESSOR (9-Bit) Mode which uses an extra (9^{th}) bit for selective communication when a number of processors share a common UART bus. In Multiprocessor Mode (also referred to as 9-Bit Mode), the multiprocessor bit (MP) is transmitted immediately following the 8 bits of data and immediately preceding the Stop bit(s), as illustrated in Figure 28.

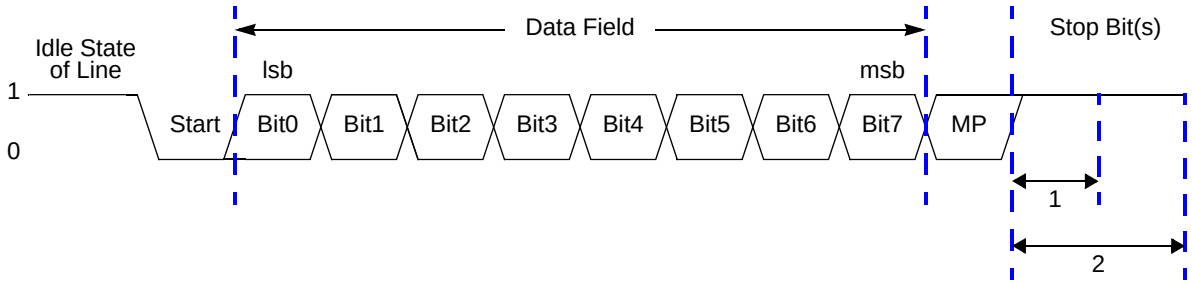


Figure 28. LIN-UART Asynchronous Multiprocessor Mode Data Format

In MULTIPROCESSOR (9-Bit) Mode, the Parity (9th) bit location becomes the MULTIPROCESSOR control bit. The LIN-UART Control 1 and Status 1 registers provide MULTIPROCESSOR (9-Bit) Mode control and status information. If an automatic address matching scheme is enabled, the LIN-UART Address Compare Register holds the network address of the device.

Multiprocessor (9-Bit) Mode Receive Interrupts

When Multiprocessor Mode is enabled, the LIN-UART processes only frames addressed to it. You can determine whether a frame of data is addressed to the LIN-UART is made in hardware, software or a combination of the two, depending on the multiprocessor configuration bits. In general, the address compare feature reduces the load on the CPU because it is not required to access the LIN-UART when it receives data directed to other devices on the multi-node network. The following 3 MULTIPROCESSOR modes are available in the hardware:

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

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

The first scheme is enabled by writing `01b` to MPMD[1:0]. In this mode, all incoming address bytes cause an interrupt, while data bytes never cause an interrupt. The ISR checks the address byte which triggered the interrupt. If it matches the LIN-UART address, the software clears MPMD[0]. At this point, each new incoming byte interrupts the CPU. The software determines the end of the frame and checks for it by reading the MPRX bit of the LIN-UART Status 1 Register for each incoming byte. If MPRX=1, a new frame has begun. If the address of this new frame is different from the LIN-UART's address, then

MPMD[0] must be set to 1 by software, causing the LIN-UART interrupts to go inactive until the next address byte. If the new frame's address matches the LIN-UART's, then the data in the new frame is processed.

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

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

LIN Protocol Mode

The LIN protocol as supported by the LIN-UART module is defined in revision 2.0 of the LIN specification package. The LIN protocol specification covers all aspects of transferring information between LIN master and slave devices using message frames including error detection and recovery, Sleep Mode and wake up from Sleep Mode. The LIN-UART hardware in LIN Mode provides character transfers to support the LIN protocol including Break transmission and detection, WAKE-UP transmission and detection and slave autobauding. Part of the error detection of the LIN protocol is for both master and slave devices to monitor their receive data when transmitting. If the receive and transmit data streams do not match, the LIN-UART asserts the PLE bit (physical layer error bit in Status 0 Register). The message frame time-out aspect of the protocol is left to software, requiring the use of an additional general purpose timer. The LIN Mode of the LIN-UART does not provide any hardware support for computing/verifying the checksum field or to verify the contents of the identifier field. These fields are treated as data and are not interpreted by the hardware.

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

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

- Parity error (PE bit in Status 0 Register) is redefined as the Physical Layer Error (PLE) bit. The PLE bit indicates thatreceive data does not match transmit data when the LIN-UART is transmitting. This applies to both Master and Slave operating modes.
- The break detect interrupt (BRKD bit in Status 0 Register) indicates when a Break is detected by the slave (break condition for at least 11 bit times). Software uses this interrupt to start a timer checking for message frame time-out. The duration of the break is read in the RxBreakLength[3:0] field of the Mode Status Register.
- The break detect interrupt (BRKD bit in Status 0 Register) indicates when a wake-up message has been received if the LIN-UART is in a LINSLEEP state.
- In LIN Slave Mode, if the BRG counter overflows while measuring the autobaud period (Start bit to beginning of bit 7 of autobaud character) an overrun error is indicated (OE bit in the Status 0 Register). In this case, software sets the LinState field back to `10b`, where the slave ignores the current message and waits for the next Break signal. The baud reload high and low registers are not updated by hardware if this autobaud error occurs. The OE bit is also set if a data overrun error occurs.

LIN System Clock Requirements

The LIN master provides the timing reference for the LIN network and is required to have a clock source with a tolerance of ±0.5%. A slave with autobaud capability is required to have a baud clock matching the master oscillator within ±14%. The slave nodes autobaud to lock onto the master timing reference with an accuracy of ±2%. If a slave does not contain autobaud capability, it must include a baud clock which deviates from the masters by no more than ±1.5%. These accuracy requirements must include effects such as voltage and temperature drift during operation.

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

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

LIN Mode Initialization and Operation

A LIN protocol mode is selected by setting either the LIN master (LMST) or LIN slave (LSLV) and optionally (for LIN slave) the autobaud enable (ABEN) bits in the LIN Control Register. To access the LIN Control Register, the mode select (MSEL) field of the LIN-UART Mode Select/Status Register must be `010b`. The LIN-UART Control 0 Register must be initialized with TEN = 1, REN = 1, all other bits = 0.

In addition to the LMST, LSLV and ABEN bits in the LIN Control Register, a LinState[1:0] field exists that defines the current state of the LIN logic. This field is initially

set by the software. In the LIN Slave Mode, the LinState field is updated by hardware as the slave moves through the Wait for Break, AutoBaud and Active states.

The noise filter is also required to be enabled and configured when interfacing to a LIN bus.

LIN Master Mode Operation

LIN Master Mode is selected by setting the bits LMST = 1, LSLV = 0, ABEN = 0, LinState[1:0] = 11b. If the LIN bus protocol indicates the bus is required go into the LIN Sleep state, the LinState[1:0] bits must be set = 00b by the software.

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

The Synch character is transmitted by writing a 55h to the Transmit Data Register (TDRE must be 1 before writing). The Synch character is not transmitted by the hardware until after the Break is complete.

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

If the master is sending the response portion of the message, these data and checksum characters are written to the Transmit Data Register when the TDRE bit asserts.

If the Transmit Data Register is written after TDRE asserts, but before TXE asserts, the hardware inserts one or two Stop bits between each character as determined by the Stop bit in the Control 0 Register. Additional idle time occurs between characters if TXE asserts before the next character is written.

LIN Sleep Mode

While the LIN bus is in the Sleep state, the CPU is in either low power Stop Mode, in Halt Mode, or in normal operational state. Any device on the LIN bus issues a Wake-up message (transmits an 80h character) if it requires the master to initiate a LIN message frame. Following the Wake-up message, the master wakes up and initiates a new message.

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

If the CPU is in HALT or OPERATIONAL Mode, the LIN-UART (if enabled) times the duration of the Wake-up and provides an interrupt following the end of the break sequence if the duration is ≥ 4 bit times. The total duration of the Wake-up message in bit times is

obtained by reading the RxBreakLength field in the mode status register. After a Wake-up message is detected, the LIN-UART is placed (by software) into either Lin Master or Lin Slave Wait for Break states as appropriate. If the break duration exceeds fifteen bit times, the RxBreakLength field contains the value Fh.

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

LIN Slave Operation

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

Following the Break the LIN-UART hardware automatically transits to the autobaud state, where it autobauds by timing the duration of the first 8 bit times of the Synch character as defined in the standard. At the end of the autobaud period, the duration measured by the BRG counter (auto baud period divided by 8) is automatically transferred to the baud reload high and low registers if the ABEN bit of the LIN Control Register is set. If the BRG counter overflows before reaching the start of bit 7 in the autobaud sequence the autobaud overrun error interrupt occurs, the OE bit in the Status 0 Register is set and the baud reload registers are not updated. To autobaud within 2% of the master's baud rate, the slave system clock must be minimum 100 times the baud rate. To avoid an autobaud overrun error, the system clock must not be greater than 2^{19} times the baud rate (16 bit counter following 3-bit prescaler when counting the 8 bit times of the autobaud sequence).

Following the Synch character, the LIN-UART hardware transits to the active state, where the Identifier character is received and the characters of the response section of the message are sent or received. The slave remains in the active state until a Break is received or the software forces a state change. When it is in active State (autobaud has completed), a Break of 10 or more bit times is recognized and a transition to the autobaud state is caused.

LIN-UART Interrupts

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

Transmitter Interrupts

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

Receiver Interrupts

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

- A data byte is received and is available in the LIN-UART Receive Data Register. This interrupt is disabled independent of the other receiver interrupt sources using the RDAIRQ bit (this feature is useful in devices, which support DMA). The received data interrupt occurs after the receive character is placed in the Receive Data Register. To avoid an overrun error, the software responds to this received data available condition before the next character is completely received.

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

- A break is received.
- A receive data overrun or LIN slave autobaud overrun error is detected.
- A data framing error is detected.
- A parity error is detected (physical layer error in LIN Mode).

LIN-UART Overrun Errors

When an overrun error condition occurs, the LIN-UART prevents overwriting of the valid data currently in the Receive Data Register. The break detect and overrun status bits are not displayed until the valid data is read.

When the valid data is read, the OE bit of the Status 0 Register is updated to indicate the overrun condition (and Break Detect, ifapplicable). The RDA bit is set to 1 to indicate that the Receive Data Register contains a data byte. However, because the overrun error occurred, this byte may not contain valid data and must be ignored. The BRKD bit indicates if the overrun is caused due to a break condition on the line. After reading the status

byte indicating an overrun error, the Receive Data Register must be read again to clear the error bits in the LIN-UART Status 0 Register.

In LIN Mode, an overrun error is signaled for receive data overruns as described above and in the LIN Slave, if the BRG counter overflows during the autobaud sequence (the ATB bit will also be set in this case). There is no data associated with the autobaud overflow interrupt, however the Receive Data Register must be read to clear the OE bit. In this case software must write 10b to the LinState field, forcing the LIN slave back to Wait for Break state.

LIN-UART Data and Error Handling Procedure

Figure 29 displays the recommended procedure for use in LIN-UART receiver interrupt service routines.

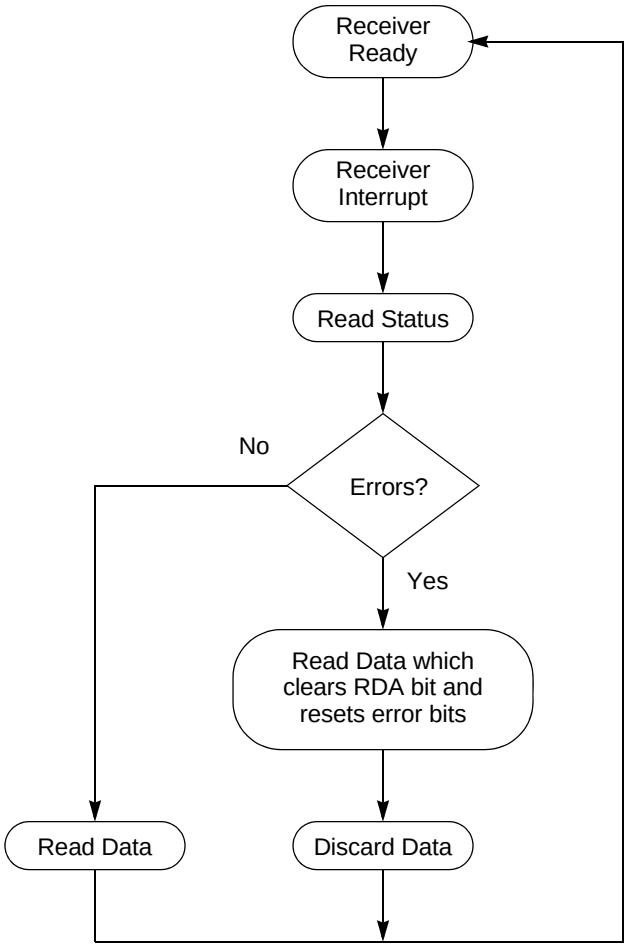


Figure 29. LIN-UART Receiver Interrupt Service Routine Flow

Baud Rate Generator Interrupts

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

LIN-UART DMA Interface

The DMA engine is configured to move UART transmit and/or receive data. This reduces processor overhead, especially when moving blocks of data. The DMA interface on the LIN-UART consists of the TxDmaReq and RxDmaReq outputs and the TxDmaAck and RxDmaAck inputs. Any of the DMA channels are configured to process the UART DMA requests.

If transmit data is to be moved by the DMA, the transmit interrupt must be disabled in the interrupt controller. If receive data is to be moved by the DMA, the RDAIRQ bit in the LIN-UART Control 1 Register must be set. This disables receive data interrupts when still enabling error interrupts. The receive interrupt must be enabled in the interrupt controller to process error condition interrupts.

LIN-UART Baud Rate Generator

The LIN-UART baud rate generator creates a lower frequency baud rate clock for data transmission. The input to the BRG is the system clock. The LIN-UART Baud Rate High and Low Byte registers combine to create a 16-bit baud rate divisor value (BRG[15:0]) which sets the data transmission rate (baud rate) of the LIN-UART. The LIN-UART data rate is calculated using the following equation for normal UART operation:

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

The LIN-UART data rate is calculated using the following equation for LIN Mode UART operation:

$$\text{UART Data Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{\text{UART Baud Rate Divisor Value}}$$

When the LIN-UART is disabled, the BRG functions as a basic 16-bit timer with interrupt on time-out. Observe the following steps to configure BRG as a timer with interrupt on time-out:

1. Disable the LIN-UART receiver by clearing the REN bit in the LIN-UART Control 0 Register to 0 (TEN bit is asserted, transmit activity may occur).

2. Load the appropriate 16-bit count value into the LIN-UART Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and associated interrupt by setting the BRGCTL bit in the LIN-UART Control 1 Register to 1. Enable the UART receive interrupt in the interrupt controller.

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

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

Noise Filter

A noise filter circuit is included to filter noise on a digital input signal, such as UART receive data before the data is sampled by the block. This filter is a requirement for protocols in a noisy environment.

The noise filter includes following features:

- Synchronizes the receive input data to the system clock.
- Noise filter enable (NFEN)input selects whether the noise filter is bypassed (NFEN = 0) or included (NFEN = 1) in the receive data path.
- Noise filter control (NFCTL[2:0])input selects the width of the up/down saturating counter digital filter. The available widths range is from 4 to11 bits.
- The digital filter output has hysteresis.
- Provides an active low saturated state output (FiltSatB), used to indicate presence of noise.

Architecture

Figure 30 displays how the noise filter is integrated with the LIN-UART for use on a LIN network.

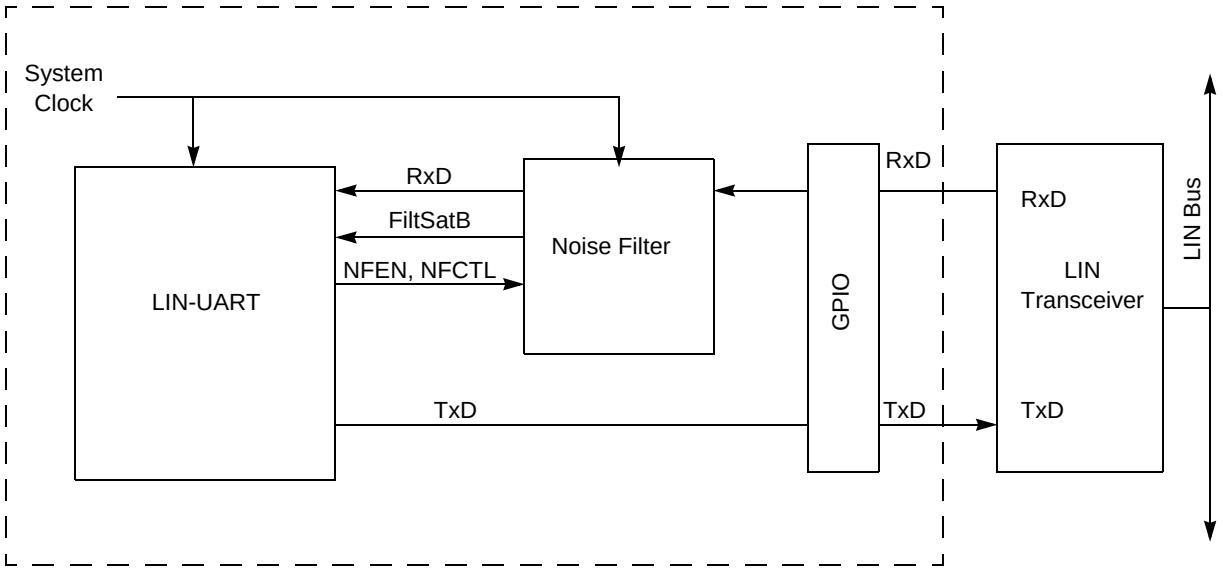


Figure 30. Noise Filter System Block Diagram

Operation

Figure 31 displays the operation of the noise filter with and without noise. The noise filter in this example is a 2-bit up/down counter, which saturates at `00b` and `11b`. A 2-bit counter is shown for convenience, the operation of wider counters is similar. The output of the filter switches from 1 to 0 when the counter counts down from `01b` to `00b` and switches from 0 to 1 when the counter counts up from `10b` to `11b`. The noise filter delays the received data by three system clock cycles.

The FiltSatB signal is checked when the filtered RxD is sampled in the center of the bit time. The presence of noise (FiltSatB = 1 at center of bit time) does not mean the sampled data is incorrect, just that the filter is not in its saturated state of all 1's or all 0's. If FiltSatB = 1 when RxD is sampled during a receive character, the NE bit in the ModeStatus[4:0] field is set. An indication of the level of noise in the network is obtained by observing this bit.

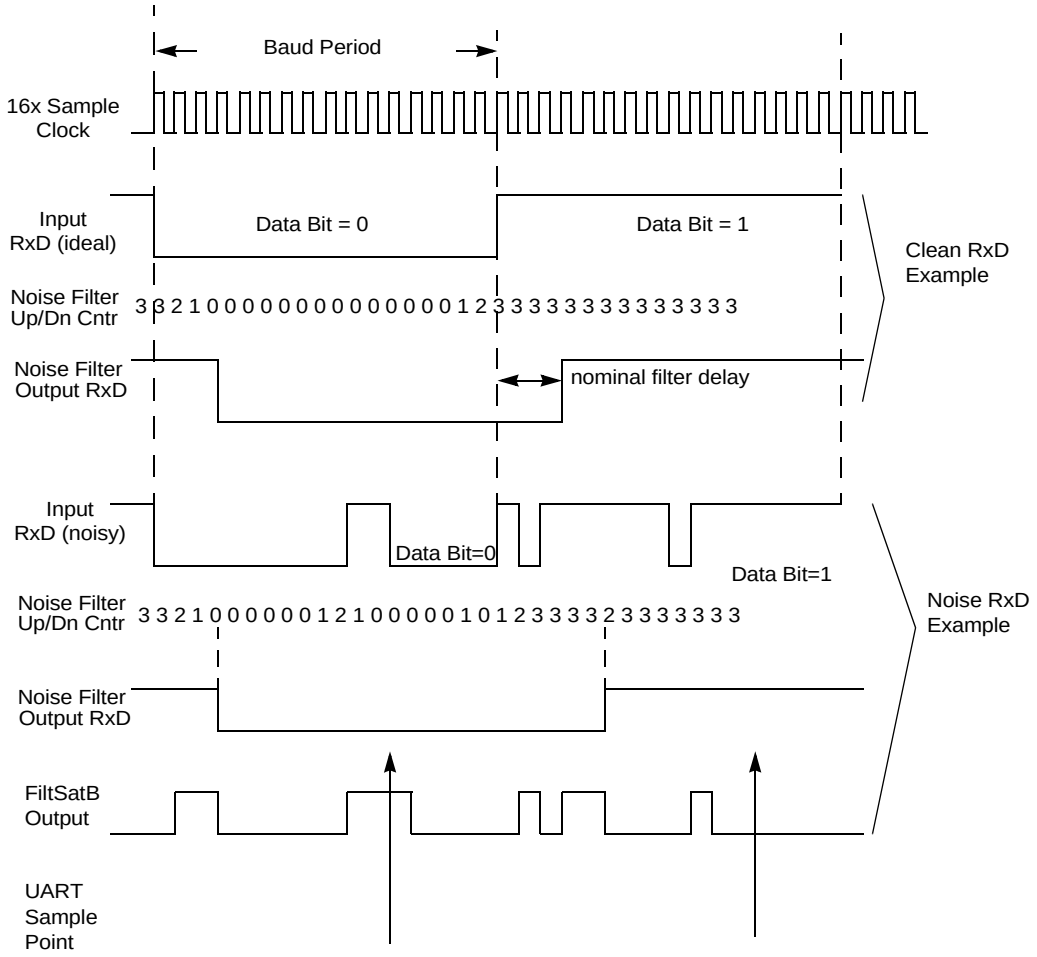


Figure 31. Noise Filter Operation

LIN-UART Control Register Definitions

The LIN-UART control registers support the LIN-UART, the associated Infrared encoder/decoder and the noise filter. For detailed information about the infrared operation, see the Infrared Encoder/Decoder chapter on page 172.

LIN-UART Transmit Data Register

Data bytes written to the LIN-UART Transmit Data Register, shown in Table 80, are shifted out on the TXD pin. The Write-only LIN-UART Transmit Data Register shares a Register File address with the Read-only LIN-UART Receive Data register.

Table 80. LIN-UART Transmit Data Register (UxTXD)

Bits	7	6	5	4	3	2	1	0
Field	TXD							
RESET	X							
R/W	W							
Addr	FF–E200h, FF–E210h							

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

LIN-UART Receive Data Register

Data bytes received through the RXD pin are stored in the LIN-UART Receive Data Register, shown in Table 81. The Read-only LIN-UART Receive Data Register shares a register file address with the Write-only LIN-UART Transmit Data Register.

Table 81. LIN-UART Receive Data Register (UxRXD)

Bits	7	6	5	4	3	2	1	0
Field	RXD							
RESET	X							
R/W	R							
Addr	FF–E200h, FF–E210h							

Bit	Description
[7:0] RXD	**Receive Data** LIN-UART receiver data byte from the RXD pin

LIN-UART Status 0 Register

The LIN-UART Status 0 Register identifies the current LIN-UART operating configuration and status. Table 82 describes the Status 0 Register for standard UART Mode. Table 83 describes the Status 0 Register for LIN Mode.

Table 82. LIN-UART Status 0 Register, Standard UART Mode (UxSTAT0)

Bits	7	6	5	4	3	2	1	0
Field	RDA	PE	OE	FE	BRKD	TDRE	TXE	CTS
RESET	0	0	0	0	0	1	1	X
R/W	R	R	R	R	R	R	R	R
Addr	FF–E201H, FF–E211H							

Bit	Description
[7] RDA	**Receive Data Available** This bit indicates that the LIN-UART Receive Data Register has received data. Reading the LIN-UART Receive Data Register clears this bit. 0 = The LIN-UART Receive Data Register is empty. 1 = There is a byte in the LIN-UART Receive Data Register.
[6] PE	**Parity Error** This bit indicates that a parity error has occurred. Reading the Receive Data Register clears this bit. 0 = No parity error occurred. 1 = A parity error occurred.
[5] OE	**Overrun Error** This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the Receive Data Register has not been read. Reading the Receive Data Register clears this bit. 0 = No overrun error occurred. 1 = An overrun error occurred.
[4] FE	**Framing Error** This bit indicates that a framing error (no Stop bit following data reception) is detected. Reading the Receive Data Register clears this bit. 0 = No framing error occurred. 1 = A framing error occurred.
[3] BRKD	**Break Detect** This bit indicates that a break has occurred. If the data bits, parity/multiprocessor bit and Stop bit(s) are all zeros then this bit is set to 1. Reading the Receive Data Register clears this bit. 0 = No break occurred. 1 = A break occurred.

Bit	Description
[2] TDRE	**Transmitter Data Register Empty** This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit. 0 = Do not write to the Transmit Data Register. 1 = The Transmit Data Register is ready to receive an additional byte to be transmitted.
[1] TXE	**Transmitter Empty** This bit indicates that the transmit shift register is empty and character transmission is finished. 0 = Data is currently transmitting. 1 = Transmission is complete.
[0] CTS	**$\overline{\text{CTS}}$ Signal** When this bit is read it returns the $\overline{\text{CTS}}$ signal level. If LBEN = 1, the $\overline{\text{CTS}}$ input signal is replaced by the internal receive data Available signal to provide flow control in LOOPBACK Mode. CTS only affects transmission if the CTSE bit = 1.

Table 83. LIN-UART Status 0 Register, LIN Mode (UxSTAT0)

Bits	7	6	5	4	3	2	1	0
Field	RDA	PLE	OE	FE	BRKD	TDRE	TXE	ATB
RESET	0	0	0	0	0	1	1	0
R/W	R	R	R	R	R	R	R	R
Addr	FF–E201H, FF–E211H							

Bit	Description
[7] RDA	**Receive Data Available** This bit indicates that the Receive Data Register has received data. Reading the Receive Data Register clears this bit. 0 = The Receive Data Register is empty. 1 = There is a byte in the Receive Data Register.
[6] PLE	**Physical Layer Error** This bit indicates that transmit and receive data do not match when a LIN slave or master is transmitting. This type of instance can be caused by a fault in the physical layer or multiple devices driving the bus simultaneously. Reading the Status 0 Register or the Receive Data Register clears this bit. 0 = Transmit and receive data match. 1 = Transmit and receive data do not match.

Bit	Description (Continued)
[5] OE	**Receive Data and Autobaud Overrun Error** This bit is set just as in normal UART operation if a receive data overrun error occurs. This bit is also set during LIN slave autobaud if the BRG counter overflows before the end of the autobaud sequence, indicating that the receive activity was not an autobaud character or the master baud rate is too slow. The ATB status bit will also be set in this case. This bit is cleared by reading the Receive Data Register. 0 = No autobaud or data overrun error occurred. 1 = An autobaud or data overrun error occurred.
[4] FE	**Framing Error** This bit indicates that a framing error (no Stop bit following data reception) is detected. Reading the Receive Data Register clears this bit. 0 = No framing error occurred. 1 = A framing error occurred.
[3] BRKD	**Break Detect** This bit is set in LIN Mode if (a) in LinSleep state and a break of at least 4 bit times occurred (Wake-up event) or (b) in Slave Wait Break state and a break of at least 11 bit times occurred (Break event), or (c) in Slave Active state and a break of at least 10 bit times occurs. Reading the Status 0 Register or the Receive Data Register clears this bit. 0 = No LIN break occurred. 1 = A LIN break occurred.
[2] TDRE	**Transmitter Data Register Empty** This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit. 0 = Do not write to the Transmit Data Register. 1 = The Transmit Data Register is ready to receive an additional byte to be transmitted.
[1] TXE	**Transmitter Empty** This bit indicates that the transmit shift register is empty and character transmission is finished. 0 = Data is currently transmitting. 1 = Transmission is complete.
[0] ATB	**LIN Slave AutoBaud Complete** This bit is set in LIN Slave Mode when an autobaud character is received. If the ABIEN bit is set in the LIN Control Register then a receive interrupt is generated when this bit is set. Reading the Status 0 Register clears this bit. This bit will be 0 in LIN Master Mode.

LIN-UART Mode Select and Status Register

The LIN-UART Mode Select and Status Register contains mode select and status bits. The four Mode Status options in Table 84 are further described in Tables 85 through 88.

Table 84. LIN-UART Mode Select and Status Register (UxMDSTAT)

Bits	7	6	5	4	3	2	1	0
Field	MSEL			Mode Status				
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R	R	R
Addr	FF–E204h, FF–E214h							

Bit	Description
[7:5] MSEL	**Mode Select** This R/W field determines which control register is accessed when performing a Write or Read to the UART Control 1 Register address. This field also determines which status is returned in the mode status field when reading this register. 000 = Multiprocessor and normal UART control/status 001 = Noise Filter control/status 010 = LIN Protocol control/status 011–110: Reserved 111 = LIN-UART hardware revision (allows hardware revision to be read in the mode status field)
[4:0] Mode Status	**Mode Status** This read-only field returns status corresponding to the mode selected by MSEL as follows: 000: MULTIPROCESSOR and NORMAL UART Mode status = {NE, 0, 0, NEWFRM, MPRX} 001: Noise filter status = {NE, 0,0,0,0} 010: LIN Mode status = {NE, RxBreakLength[3:0]} 011–110: Reserved = {0, 0, 0, 0, 0} 111: LIN-UART hardware revision

Table 85. Multiprocessor Mode Status Field (MSEL = 000b)

NE	**Noise Event** This bit is asserted if digital noise is detected on the receive data line when the data is sampled (center of bit time). If this bit is set, it does not mean that the receive data is corrupted (in extreme cases), just that one or more of the noise filter data samples near the center of the bit time did not match the average data value.
NEWFRM	**New Frame** Status bit denoting the start of a new frame. Reading the LIN-UART Receive Data Register Resets this bit to 0. 0 = The current byte is not the first data byte of a new frame. 1 = The current byte is the first data byte of a new frame.
MPRX	**Multiprocessor Receive** Returns the value of the last multiprocessor bit received. Reading from the LIN-UART Receive Data Register Resets this bit to 0.

Table 86. Digital Noise Filter Mode Status Field (MSEL = 001b)

NE	**Noise Event** This bit is asserted if digital noise is detected on the receive data line while the data is sampled (center of bit time). If this bit is set, it does not mean that the receive data is corrupted (in extreme cases), just that one or more of the noise filter data samples near the center of the bit time did not match the average data value.

Table 87. LIN Mode Status Field (MSEL = 010b)

NE	**Noise Event** This bit is asserted if some noise level is detected on the receive data line when the data is sampled (center of bit time). If this bit is set, it does not indicate that the receive data is corrupt (in extreme cases), just that one or more of the 16x data samples near the center of the bit time did not match the average data value.
RxBreak-Length	**LIN Mode Received Break Length** This field is read following a break (LIN WAKE-UP or BREAK) so software determines the measured duration of the break. If the break exceeds 15 bit times the value saturates at 1111b.

Table 88. Hardware Revision Mode Status Field (MSEL = 111b)

LIN-UART Hardware Revision	This field indicates the hardware revision of the LIN-UART block. 00_xxx LIN UART hardware rev 01_xxx Reserved 10_xxx Reserved 11_xxx Reserved

LIN-UART Control 0 Register

The LIN-UART Control 0 Register, shown in Table 89, configures the basic properties of the LIN-UART's transmit and receive operations.

Table 89. LIN-UART Control 0 Register (UxCTL0)

Bits	7	6	5	4	3	2	1	0
Field	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF–E202h, FF–E212h							

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

Bit	Description (Continued)
[2] SBRK	**Send Break** 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. In standard UART Mode, the duration of the break is determined by how long software leaves this bit asserted. Also the duration of any required Stop bits following the break must be timed by software before writing a new byte to be transmitted to the Transmit Data Register. In LIN Mode, the master sends a Break character by asserting SBRK. The duration of the break is timed by hardware and the SBRK bit is deasserted by hardware when the Break is completed. The duration of the Break is determined by the TxBreakLength field of the LIN Control Register. One or two Stop bits are automatically provided by the hardware in LIN Mode as defined by the Stop bit. 0 = No break is sent. 1 = The output of the transmitter is 0.
[1] STOP	**Stop Bit Select** 0 = The transmitter sends one stop bit. 1 = The transmitter sends two stop bits.
[0] LBEN	**Loop Back Enable** 0 = Normal operation. 1 = All transmitted data is looped back to the receiver within the IrDA module.

LIN-UART Control 1 Registers

Multiple LIN-UART control registers, shown in Tables 90 through 92, are accessible by a single bus address. The register selected is determined by the mode select (MSEL) field. These registers provide additional control over the LIN-UART operation.

Multiprocessor Control Register (LIN-UART Control 1 Register with MSEL = 000b)

When MSEL = 000b, this register provides control for UART Multiprocessor Mode, IRDA Mode, baud rate timer mode as well as other features which applies to multiple modes.

Table 90. MultiProcessor Control Register (UxCTL1 with MSEL = 000b)

Bits	7	6	5	4	3	2	1	0
Field	MPMD[1]	MPEN	MPMD[0]	MPBT	DEPOL	BRGCTL	RDAIRQ	IREN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF–E203h, FF–E213h with MSEL = 000b							

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

Bit	Description (Continued)
[2] BRGCTL	**Baud Rate Generator Control** This bit causes different LIN-UART behaviors, depending on whether the LIN-UART receiver is enabled (REN = 1 in the LIN-UART Control 0 Register). When the LIN-UART receiver is not enabled, this bit determines whether the baud rate generator issues interrupts. 0 = BRG is disabled. Reads from the Baud Rate High and Low Byte registers return the BRG Reload Value. 1 = BRG is enabled and counting. The BRG 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. When the LIN-UART receiver is enabled, this bit allows reads from the baud rate registers to return the BRG count value instead of the Reload Value. 0 = Reads from the Baud Rate High and Low Byte registers return the BRG Reload Value. 1 = Reads from the Baud Rate High and Low Byte registers return the current BRG count value. Unlike the timers, there is no mechanism to latch the High byte when the Low byte is read.
[1] RDAIRQ	**Receive Data Interrupt Enable** 0 = Received data and receiver errors generates an interrupt request to the interrupt controller. 1 = Received data does not generate an interrupt request to the interrupt controller. Only receiver errors generate an interrupt request.
[0] IREN	**Infrared Encoder/Decoder Enable** 0 = Infrared encoder/decoder is disabled. LIN-UART operates normally. 1 = Infrared encoder/decoder is enabled. The LIN-UART transmits and receives data through the Infrared encoder/decoder.

Noise Filter Control Register (LIN-UART Control 1 Register with MSEL = 001b)

When MSEL = 001b, this register, shown in Table 91, provides control for the digital noise filter.

Table 91. Noise Filter Control Register (UxCTL1 with MSEL = 001b)

Bits	7	6	5	4	3	2	1	0
Field	NFEN	NFCTL			Reserved			
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R	R	R
Addr	FF–E203h, FF–E213h with MSEL = 001b							

Bit	Description
[7] NFEN	**Noise Filter Enable** 0 = Noise filter is disabled. 1 = Noise filter is enabled. Receive data is preprocessed by the noise filter.
[6:4] NFCTL	**Noise Filter Control** This field controls the delay and noise rejection characteristics of the noise filter. The wider the counter the more delay that is introduced by the filter and the wider the noise event that is filtered. 000 = 4-bit up/down counter 001 = 5-bit up/down counter 010 = 6-bit up/down counter 011 = 7-bit up/down counter 100 = 8-bit up/down counter 101 = 9-bit up/down counter 110 = 10-bit up/down counter 111 = 11-bit up/down counter
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

LIN Control Register (LIN-UART Control 1 Register with MSEL = 010b)

When MSEL = 010b, this register, shown in Table 92, provides control for the LIN Mode of operation.

Table 92. LIN Control Register (UxCTL1 with MSEL = 010b)

Bits	7	6	5	4	3	2	1	0
Field	LMST	LSLV	ABEN	ABIEN	LINSTATE[1:0]		TxBreakLength	
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF–E203h, FF–E213h with MSEL = 010b							

Bit	Description
[7] LMST	**LIN Master Mode** 0 = LIN Master Mode not selected. 1 = LIN Master Mode selected (if MPEN, PEN, LSLV = 0)
[6] LSLV	**LIN Slave Mode** 0 = LIN Slave Mode not selected. 1 = LIN Slave Mode selected (if MPEN, PEN, LMST = 0)
[5] ABEN	**Autobaud Enable** 0 = Autobaud not enabled. 1 = Autobaud enabled if in LIN Slave Mode.
[4] ABIEN	**Autobaud Interrupt Enable** 0 = Interrupt following autobaud does not occur. 1 = Interrupt following autobaud enabled if in LIN Slave Mode. When the autobaud character is received, a receive interrupt is generated and the ATB bit is set in the Status 0 Register.
[3:2] LINSTATE[1:0]	**LIN State Machine** The LinState is controlled by both hardware and software. Software force a state change at any time if necessary. In normal operation, software moves the state in and out of Sleep state. For a LIN Slave, software changes the state from Sleep to Wait for Break after which hardware cycles through the Wait for Break, Autobaud and Active states. Software changes the state from one of the active states to Sleep state if the LIN bus goes into Sleep Mode. For a LIN Master, software changes state from Sleep to Active where it remains until software sets it back to the Sleep state. After configuration software does not alter the LinState field during operation. 00 = Sleep State (either LMST or LSLV is set) 01 = Wait for Break state (only valid for LSLV = 1) 10 = Autobaud state (only valid for LSLV = 1) 11 = Active state (either LMST or LSLV is set)

Bit	Description (Continued)
[1:0] TxBreakLength	**TxBreakLength** Used in LIN Mode by the master to control the duration of the transmitted Break. 00 = 13 bit times 01 = 14 bit times 10 = 15 bit times 11 = 16 bit times

LIN-UART Address Compare Register

The LIN-UART Address Compare Register, shown in Table 93, stores the multi-node network address of the LIN-UART. When the MPMD[1] bit of LIN-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 occur only in the event of a match.

Table 93. LIN-UART Address Compare Register (UxADDR)

Bits	7	6	5	4	3	2	1	0
Field	COMP_ADDR							
RESET	00h							
R/W	R/W							
Addr	FF–E205h, FF–E215h							

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

LIN-UART Baud Rate High and Low Byte Registers

The LIN-UART Baud Rate High and Low Byte registers, shown in Tables 94 and 95, combine to create a 16-bit baud rate divisor value (BRG[15:0]) which sets the data transmission rate (baud rate) of the LIN-UART.

Table 94. LIN-UART Baud Rate High Byte Register (UxBRH)

Bits	7	6	5	4	3	2	1	0
Field	BRH							
RESET	1							
R/W	R/W							
Addr	FF–E206h, FF–E216h							

Table 95. LIN-UART Baud Rate Low Byte Register (UxBRL)

Bits	7	6	5	4	3	2	1	0
Field	BRL							
RESET	1							
R/W	R/W							
Addr	FF–E207h, FF–E217h							

The LIN-UART data rate is calculated using the following equation for standard UART modes. For LIN protocol, the baud rate registers must be programmed with the baud period rather than 1/16 baud period.

Note: The UART must be disabled when updating the baud rate registers because High and Low registers must be written independently.

The LIN-UART data rate is calculated using the following equation for standard UART operation:

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

The LIN-UART data rate is calculated using the following equation for LIN Mode UART operation:

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

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for standard UART operation:

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

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for LIN Mode UART operation:

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

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

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

For reliable communication, the LIN-UART baud rate error must never exceed 5 percent. Table 96 provides information about baud rate errors for popular baud rates and commonly used crystal oscillator frequencies for normal UART Mode of operation.

When the LIN-UART is disabled, the baud rate generator functions 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 LIN-UART receiver by clearing the REN bit in the LIN-UART Control 0 Register to 0 (TEN bit is asserted, transmit activity may occur).
2. Load the appropriate 16-bit count value into the LIN-UART Baud Rate High and Low Byte registers.
3. Enable the baud rate generator timer function and associated interrupt by setting the BRGCTL bit in the LIN-UART Control 1 Register to 1. Enable the UART receive interrupt in the interrupt controller.

When configured as a general purpose timer, the 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 96. LIN-UART Baud Rates

20.0MHz System Clock				10.0MHz 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	N/A	N/A	N/A
625.0	2	625.0	0.00	625.0	1	625.0	0.00
250.0	5	250.0	0.00	250.0	3	208.33	–16.67
115.2	11	113.64	–1.19	115.2	5	125.0	8.51
57.6	22	56.82	–1.36	57.6	11	56.8	–1.36
38.4	33	37.88	–1.36	38.4	16	39.1	1.73
19.2	65	19.23	0.16	19.2	33	18.9	0.16
9.60	130	9.62	0.16	9.60	65	9.62	0.16
4.80	260	4.81	0.16	4.80	130	4.81	0.16
2.40	521	2.399	–0.03	2.40	260	2.40	–0.03
1.20	1042	1.199	–0.03	1.20	521	1.20	–0.03
0.60	2083	0.60	0.02	0.60	1042	0.60	–0.03
0.30	4167	0.299	–0.01	0.30	2083	0.30	0.2

Table 96. LIN-UART Baud Rates (Continued)

5.5296 MHz System Clock				3.579545 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	345.6	38.24	250.0	1	223.72	–10.51
115.2	3	115.2	0.00	115.2	2	111.9	–2.90
57.6	6	57.6	0.00	57.6	4	55.9	–2.90
38.4	9	38.4	0.00	38.4	6	37.3	–2.90
19.2	18	19.2	0.00	19.2	12	18.6	–2.90
9.60	36	9.60	0.00	9.60	23	9.73	1.32
4.80	72	4.80	0.00	4.80	47	4.76	–0.83
2.40	144	2.40	0.00	2.40	93	2.41	0.23
1.20	288	1.20	0.00	1.20	186	1.20	0.23
0.60	576	0.60	0.00	0.60	373	0.60	–0.04
0.30	1152	0.30	0.00	0.30	746	0.30	–0.04

Table 96. LIN-UART Baud Rates (Continued)

1.8432 MHz System Clock			
Desired Rate (kHz)	**BRG Divisor (Decimal)**	**Actual Rate (kHz)**	**Error (%)**
1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A
250.0	N/A	N/A	N/A
115.2	1	115.2	0.00
57.6	2	57.6	0.00
38.4	3	38.4	0.00
19.2	6	19.2	0.00
9.60	12	9.60	0.00
4.80	24	4.80	0.00
2.40	48	2.40	0.00
1.20	96	1.20	0.00
0.60	192	0.60	0.00
0.30	384	0.30	0.00

Infrared Encoder/Decoder

The ZNEO® Z16F Series products contain two fully-functional, high-performance UART-to-infrared encoder/decoders (endecs). Each infrared endec is integrated with an on-chip UART to allow easy communication between the ZNEO 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 32 displays the architecture of the infrared endec.

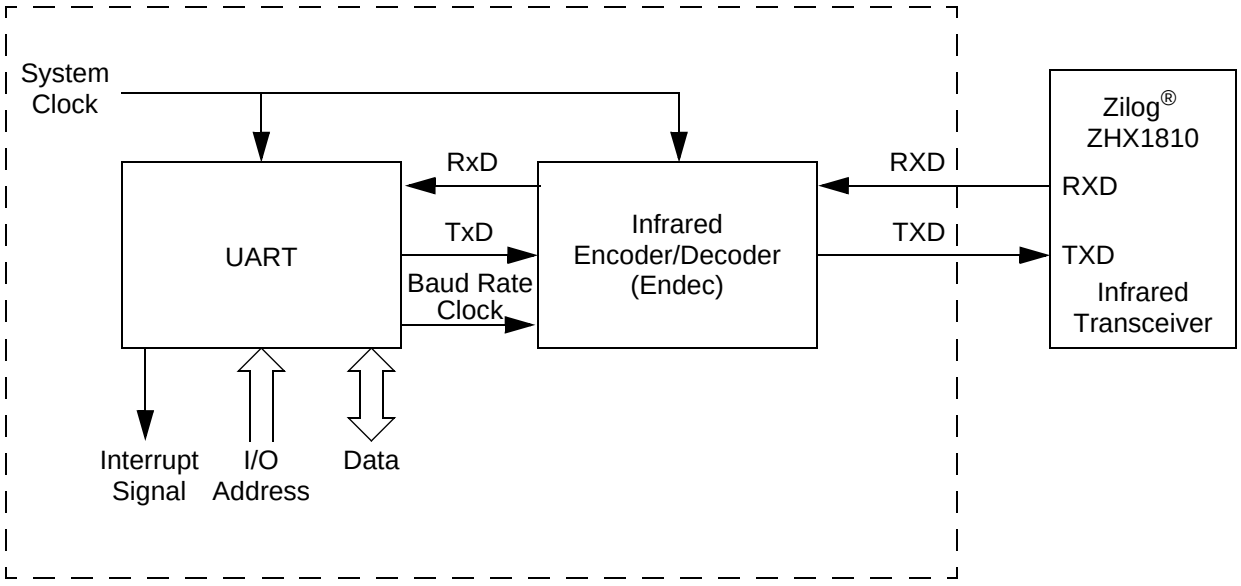


Figure 32. Infrared Data Communication System Block Diagram

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. Similarly, 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 that simultaneous data transmission and reception is not allowed.

The baud rate is set by the UART's baud rate generator and supports IrDA standard baud rates from 9600 baud to 115.2 Kbaud. Higher baud rates are possible, but do not meet IrDA specifications. The UART must be enabled to use the infrared endec. The infrared endec data rate is calculated using the following equation:

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

Transmitting IrDA Data

The data to be transmitted using the infrared transceiver is first sent to the UART. The UART's transmit signal (TXD) and baud rate clock are used by the IrDA to generate the modulation signal (IR_TXD) that drives the infrared transceiver. Each UART/infrared data bit is 16-clocks wide. If the data to be transmitted is 1, the IR_TXD signal remains Low for the full 16-clock period. If the data to be transmitted is 0, a 3-clock high pulse is output following a 7-clock low period. After the 3-clock high pulse, a 6-clock low pulse is output to complete the full 16-clock data period. Figure 33 displays IrDA data transmission. When the infrared endec is enabled, the UART's TXD signal is internal to the ZNEO Z16F Series products while the IR_TXD signal is output through the TXD pin.

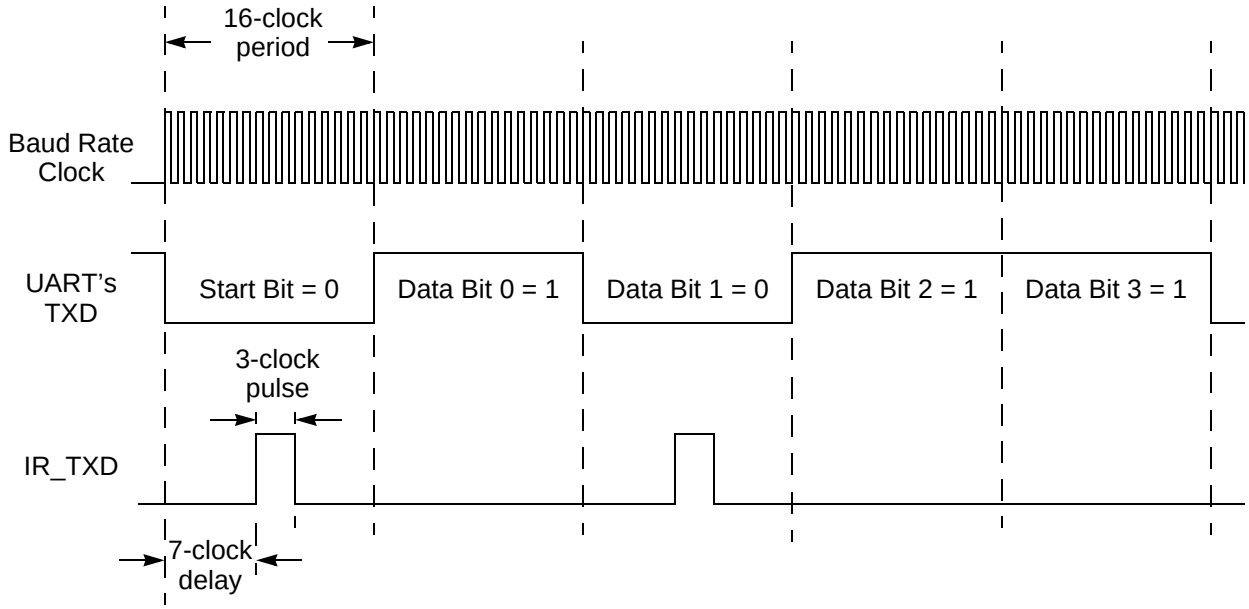


Figure 33. Infrared Data Transmission

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 34 displays data reception. When the infrared endec is enabled, the UART's RXD signal is internal to the ZNEO Z16F Series products when the IR_RXD signal is received through the RXD pin.

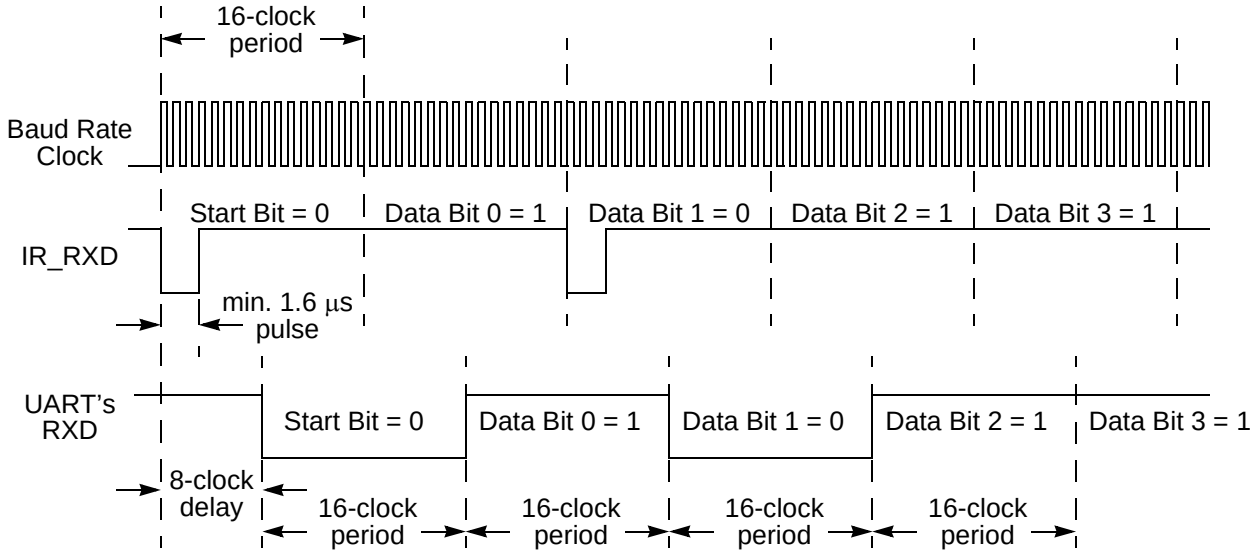


Figure 34. Infrared Data Reception

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 (or in other

words, 24 baud clock periods since the previous pulse was detected). This open window allows the endec a sampling of minus 4 baudrate clocks to plus 8 baudrate 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 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

Caution: All infrared endec configuration and status information is set by the UART control registers as defined in the beginning of the LIN-UART Control Register Definitions section on page 153.

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.

Enhanced Serial Peripheral Interface

The Enhanced Serial Peripheral Interface (ESPI) supports SPI (Serial Peripheral Interface) and Inter IC Sound (I^2S) modes of operation.

The features of the ESPI include:

- Full-duplex, synchronous, character-oriented communication
- Four-wire interface ($\overline{SS}$, SCK, MOSI, MISO)
- Transmit and receive buffer registers to enable high throughput
- Transfer rates up to maximum of one-fourth the system clock frequency in Slave Mode
- Error detection
- Dedicated programmable baud rate generator (BRG)
- Data transfer control through polling, interrupt, or DMA

Architecture

The ESPI is a full-duplex, synchronous, character-oriented channel that supporting a four-wire interface (serial clock, transmit and receive data and Slave select). The ESPI block consists of a shift register, transmit and receive data buffer registers, a baud rate (clock) generator, control/status registers and a control state machine. Transmit and receive transfers are in sync as there is a single shift register for both transmit and receive data. Figure 35 displays a block diagram of the ESPI.

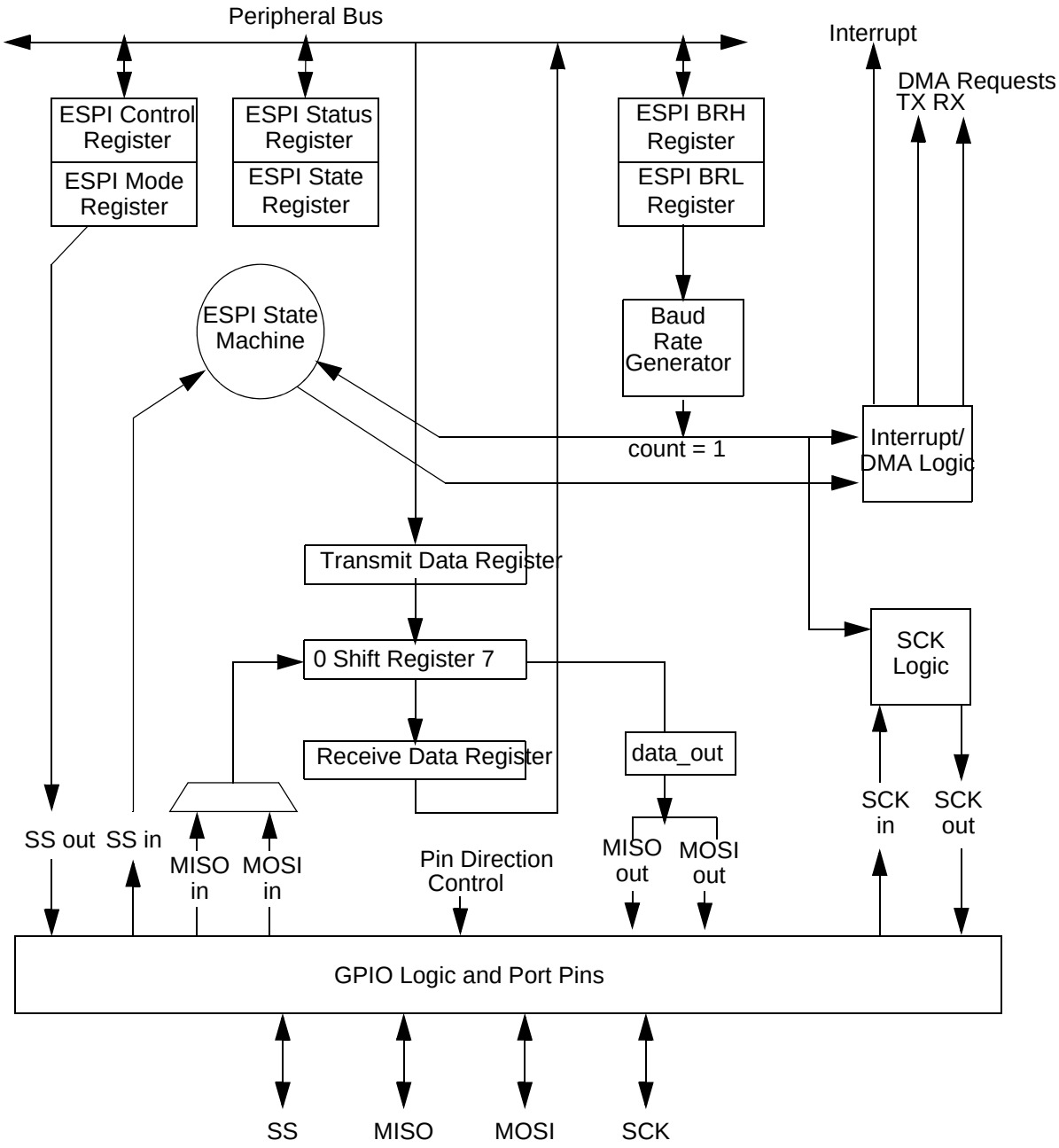


Figure 35. ESPI Block Diagram

ESPI Signals

The four ESPI signals are:

- Master-In/Slave-Out (MISO)
- Master-Out/Slave-In (MOSI)
- Serial clock (SCK)
- Slave select ($\overline{SS}$)

The following paragraphs describe these signals in both Master and Slave modes. The appropriate GPIO pins must be configured using the GPIO alternate function registers.

Master-In/Slave-Out

The MISO pin is configured as an input in a master device and as an output in a slave device. Data is transferred to most significant bit first. The MISO pin of a slave device is placed in a high-impedance state if the slave is not selected. When the ESPI is not enabled, this signal is in a high-impedance state. The direction of this pin is controlled by the MMEN bit of the ESPI Control Register.

Master-Out/Slave-In

The MOSI pin is configured as an output in a master device and as an input in a slave device. Data is transferred to most significant bit first. When the ESPI is not enabled, this signal is in a high-impedance state. The direction of this pin is controlled by the MMEN bit of the ESPI Control Register.

Serial Clock

The SCK synchronizes data movement both in and out of the shift register via the MOSI and MISO pins. In Master Mode (MMEN = 1), the ESPI's baud rate generator creates the serial clock and drives it out via its SCK pin to the slave devices. In Slave Mode, the SCK pin is an input. Slave devices ignore the SCK signal unless their $\overline{SS}$ pin is asserted.

The master and slave are each capable of exchanging a character of data during a sequence of NUMBITS clock cycles; see the NUMBITS field in the ESPI Mode Register (ESPIMODE) on page 197. In both master and slave ESPI devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. SCK phase and polarity is determined by the PHASE and CLKPOL bits in the ESPI Control Register (ESPICTL) on page 194.

Slave Select

The $\overline{SS}$ signal is a bidirectional framing signal with several modes of operation to support SPI and other synchronous serial interface protocols. The Slave Select Mode is selected by the SSMD field of the ESPI Mode Register. The direction of the $\overline{SS}$ signal is controlled by the SSIO bit of the ESPI Mode Register. The $\overline{SS}$ signal is an input on slave devices and is an output on the active master device. Slave devices ignore transactions on the bus unless their slave select input is asserted. In SPI Master Mode, additional GPIO pins are required to provide Slave Selects if there is more than one slave device.

ESPI Register Overview

The ESPI Control/Status Registers are summarized in Table 97. These registers are accessed by either Word (16-bit) or Byte operations.

Table 97. ESPI Registers

Word Address	Even Address	Odd Address
XXXXX0	Data	Transmit Data Command
XXXXX2	Control	Mode
XXXXX4	Status	State
XXXXX6	Baud Rate High	Baud Rate Low

Comparison with Basic SPI Block

The ESPI module includes many enhancements when compared to the simpler SPI module in other Z8 Encore!® parts. This section highlights the differences between the ESPI module and the SPI module as follows:

- Transmit and receive data buffer register added to support higher performance.
- Multiple interrupt sources (transmit data, receive data, errors). SPI module only has data transfer complete interrupt.
- DMA Controller interface (separate transmit and receive interfaces).
- Register addresses redefined to facilitate 16-bit transfers on the ZNEO® Z16F Series.
- Transmit data command register – new register to facilitate DMA interface and improve performance with 16-bit transfers. SSV and TEOF is set on same cycle on which the data register is written.
- Control register:
  - IRQE changed to DIRQE. This allows data interrupts to be disabled when using DMA but still allow error interrupts.

  - STR bit on the SPI module replaced with ESPIEN1. SPIEN replaced with ESPIEN0. This enhancement allows unidirectional transfers which minimizes software or DMA overhead.
  - BIRQ replaced with BRGCTL.
- Mode register:
  - Added SSMD field which adds support for loop back and I2S modes.
  - Moved SSV bit to the transmit data command register as described above.
  - Added slave select polarity (SSPO) to support active High and Low slave select on $\overline{SS}$ pin.
- Status register:
  - IRQ split into TDRE and RDRF (separate transmit and receive interrupts).
  - Replace overrun error with separate transmit under-run and receive overrun.
- State register.
  - Replaced SCKEN bit with SCKI.
  - Replaced TCKEN with SDI.

Operation

During transfer, data is sent and received simultaneously by both master and slave devices. Separate signals are required to transmit data, receive data and the serial clock. When a transfer occurs, a multibit (typically 8-bit) character is shifted out one data pin and a multi-bit character is simultaneously shifted in on a second data pin. An 8-bit shift register in the master and an 8-bit shift register in the slave is connected as a circular buffer. The ESPI shift register is buffered to support back-to-back character transfers in high performance applications.

A transaction is initiated when the Transmit Data Register is written in the master device. The value from the data register is transferred into the shift register and the transaction begins. After the transmit data is loaded into the shift register, the Transmit Data register Empty (TDRE) status bit asserts, indicating that Transmit Data Register is written with the next value. At the end of each character transfer, the shift register value (receive data) is loaded into the Receive Data Register. At that point the Receive Data register Full (RDRF) status bit asserts. When software or DMA reads the receive data from the Receive Data Register, the RDRF signal deasserts.

The master sources the SCK and $\overline{SS}$ signal during the transfer.

Internal data movement (either by software or DMA) to/from the ESPI block is controlled by the Transmit Data Register empty (TDRE) and Receive Data Register full (RDRF) signals. These signals are read only bits in the ESPI Status Register. When either the TDRE or RDRF bits assert, an interrupt is sent to the interrupt controller if the data interrupt

request enable (DIRQE) bit is set. The TDRE and RDRF signals also generate transmit and receive DMA requests.

In many cases the software application is only moving information in one direction. In such a case, either the TDRE or RDRF interrupts/DMA requests is disabled to minimize software/DMA overhead. Unidirectional data transfer is supported by setting the ESPIEN1, 0 bits in the control register to 10 or 01. If the DMA engine is being used to move the data, the transmit and receive data interrupts are disabled through the DIRQE bit of the control register. In this case error interrupts still occurs and must be handled directly by the software.

Throughput

In Master Mode, the maximum supported SCK rate is one-half the system clock frequency. This rate is achieved by programming the value 0001h into the baud rate high/low register pair. Though each character is transferred at this rate, it is unlikely that software interrupt routines or DMA keeps up with this rate. In SPI Mode, the transfer will automatically pause between characters until the current receive character is read and the next transmit data value is written.

In Slave Mode, the transfer rate is controlled by the master. As long as the TDRE and RDRF interrupt or DMA requests are serviced before the next character transfer completes the slave will keep up with the master. In Slave Mode, the baud rate is restricted to a maximum of one-fourth of the system clock frequency to allow for synchronization of the SCK input to the internal system clock.

ESPI Clock Phase and Polarity Control

The ESPI supports four combinations of SCK phase and polarity using two bits in the ESPI Control Register. The clock polarity bit, CLKPOL, selects an active High or active Low clock and has no effect on the transfer format. The clock phase bit, PHASE, selects one of two fundamentally different transfer formats. The data is output a half-cycle before the receive clock edge which provides a half cycle of setup and hold time. Table 98 lists the ESPI clock phase and polarity operation parameters.

Table 98. ESPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation

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

Transfer Format with Phase Equals Zero

Figure 36 displays the timing diagram for an SPI type transfer in which PHASE = 0. For SPI transfers the clock only toggles during the character transfer. The two SCK waveforms show polarity with CLKPOL = 0 and with CLKPOL = 1. The diagram is interpreted as either a Master or Slave timing diagram as the SCK MISO and MOSI pins are directly connected between the master and the slave.

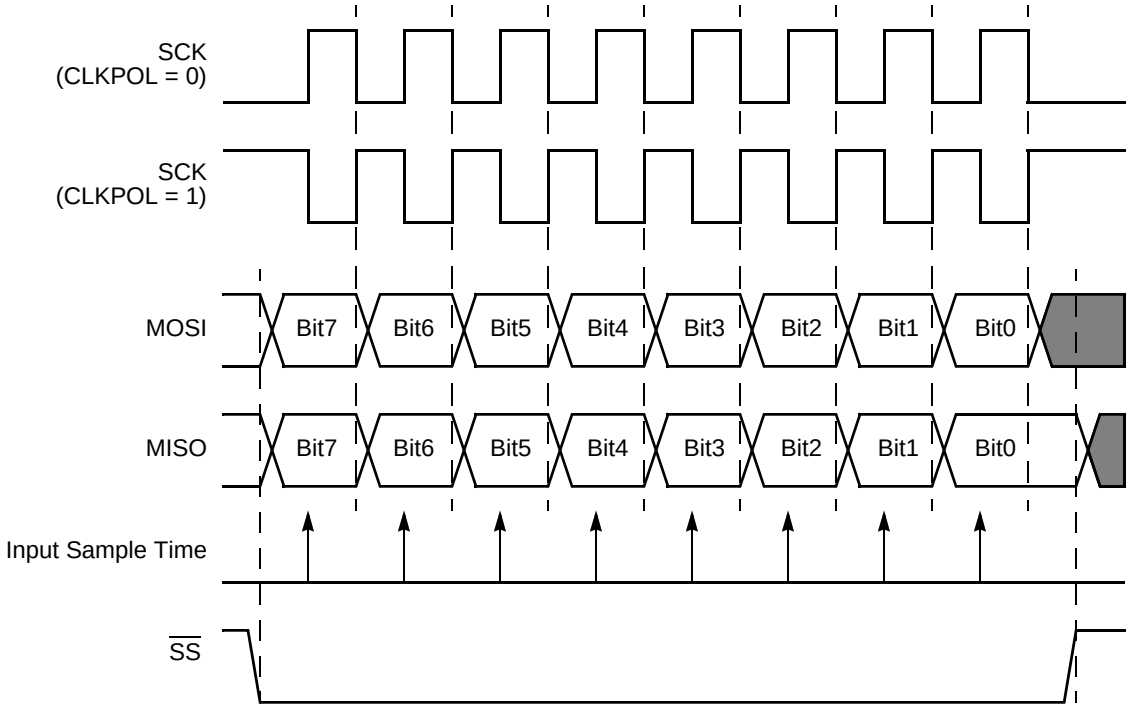


Figure 36. ESPI Timing when PHASE = 0

Transfer Format with Phase Equals One

Figure 37 displays the timing diagram for an SPI type transfer in which PHASE = 1. For SPI transfers the clock only toggles during the character transfer. Two waveforms are depicted for SCK, one for CLKPOL = 0 and another for CLKPOL = 1.

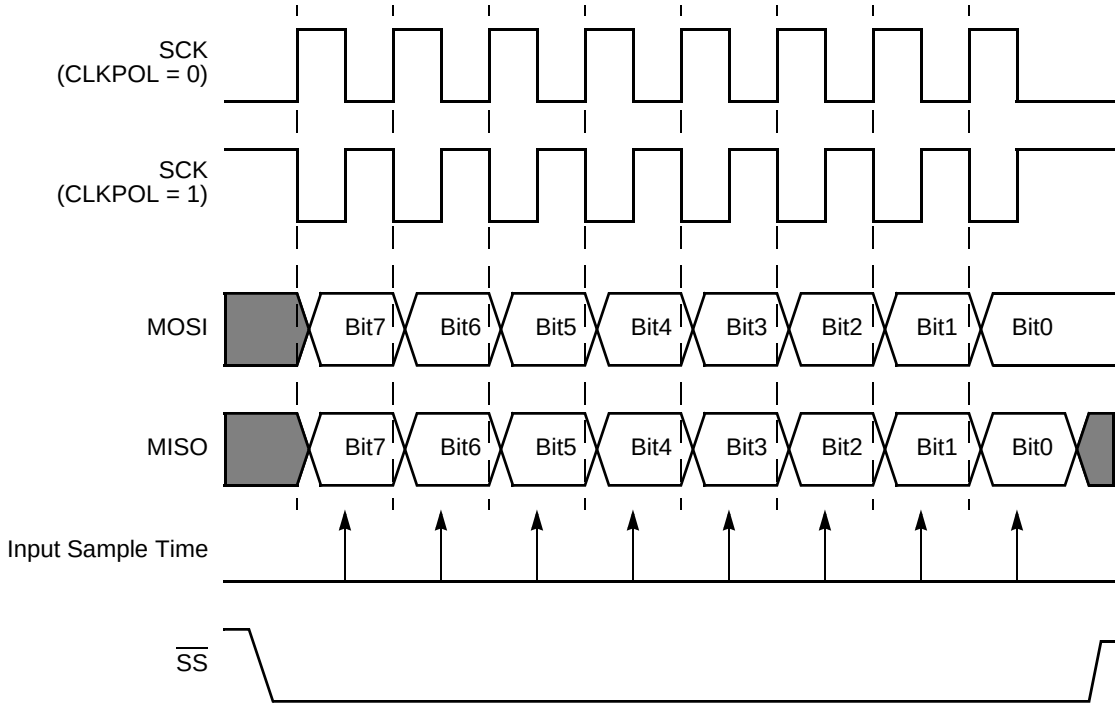


Figure 37. ESPI Timing when PHASE = 1

Modes of Operation

This section describes the different modes of data transfer supported by the ESPI block. The mode is selected by the Slave Select Mode (SSMD) field of the mode register.

SPI Mode

This mode is selected by setting the SSMD field of the mode register to 000. In this mode, software or DMA controls the assertion of the $\overline{SS}$ signal directly via the SSV bit of the SPI transmit data command register. Either DMA or software is used to control an SPI Mode transaction. Prior to or simultaneously with writing the first transmit data byte, software or DMA sets the SSV bit. Software sets the SSV bit either by performing a byte write to the transmit data command register prior to writing the first transmit character to the data register or by performing a word write to the data register address which loads the first transmit character and simultaneously sets the SSV bit.

The DMA sets the SSV bit via the command field of the descriptor. The SSV bit is written on the DMA command bus prior to or in sync with the first data byte. $\overline{SS}$ will remain asserted while one or more characters are transferred. There are two mechanisms for deasserting $\overline{SS}$ at the end of the transaction. One method is used by DMA and also by software, is to set the TEOF bit of the transmit data command register when the last TDRE interrupt

or DMA request is being serviced (set TEOF before or simultaneously with writing the last data byte). When the last bit of the last character is transmitted, the hardware will automatically deassert the SSV and TEOF bits. The second method is for software to directly clear the SSV bit after the transaction completes. If software clears the SSV bit directly, it is not necessary for software to also set the TEOF bit on the last transmit byte. After writing the last transmit byte, the end of the transaction is detected by waiting for the last RDRF interrupt or monitoring the TFST bit in the ESPI Status register.

The transmit underrun and receive overrun errors do not occur in an SPI Mode master. If the RDRF and TDRE requests have not been serviced before the current byte transfer completes, SCLK is paused until the data register is read and written. The transmit underrun and receive overrun errors will occur in a slave if the slave's software/DMA does not keep up with the master data rate. If a transmit underrun occurs in Slave Mode, the shift register in the slave is loaded with all 1s.

In the SPI Mode, the SCK is active only for the data transfer with one SCK period per bit transferred. If the SPI bus has multiple slaves, the slave select lines to all or one of the slaves must be controlled independently by software using GPIO pins.

Figure 38 displays multiple character transfer in SPI Mode. Note that while character 'n' is being transferred using the shift register, software/DMA responds to the receive request for character n-1 and the transmit request for character n+1.

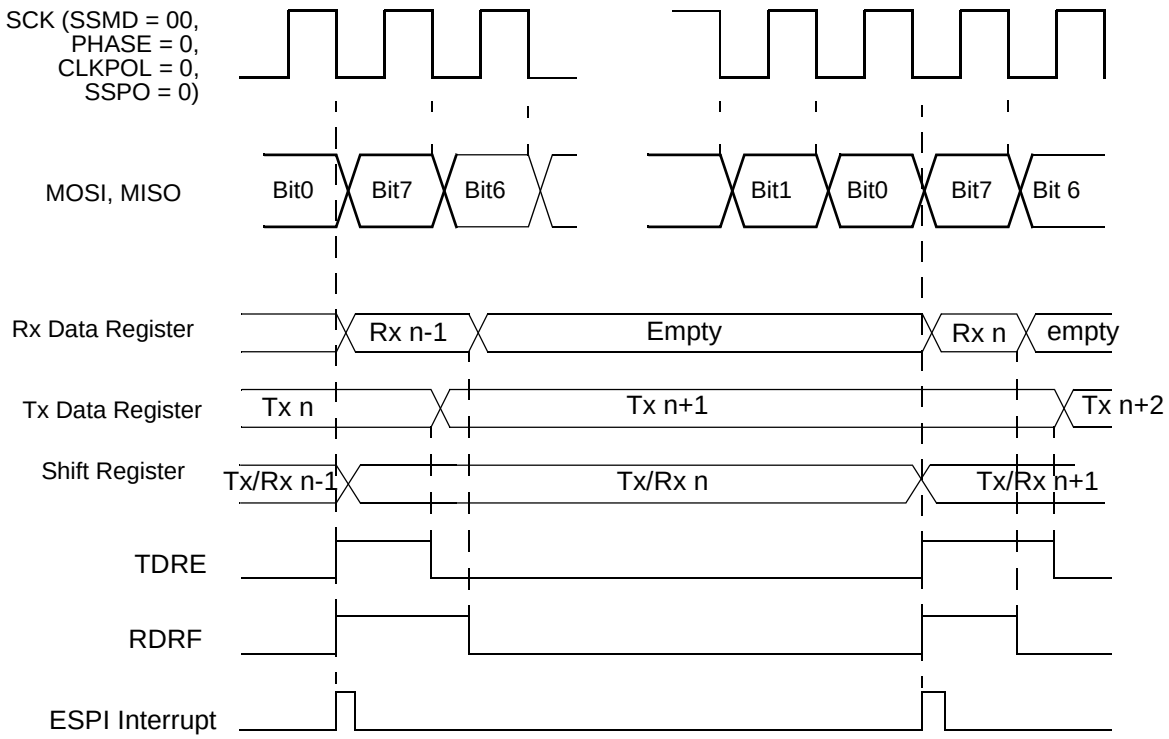


Figure 38. SPI Mode (SSMD = 000)

I2S (Inter-IC Sound) Mode

This mode is selected by setting the SSMD field of the mode register to 010. The PHASE and CLKPOL bits of the control register must be set to 0. This mode is illustrated in Figure 39 with $\overline{SS}$ alternating between consecutive frames. A frame consists of a fixed number of data bytes as defined in the DMA buffer descriptor or by software. I^2S (Inter-IC Sound) mode is typically used to transfer left or right channel audio data.

The SSV indicates whether the corresponding bytes are left or right channel data. The SSV value must be updated when servicing the TDRE interrupt/request for the first byte in a left or write channel frame. This update is accomplished by performing a word write when writing the first byte of the audio word, which updates both the ESPI data and transmit data command words or by doing a byte write to update SSV followed by a byte write to the data register. The $\overline{SS}$ signal leads the data by one SCK period.

If a DMA Channel is controlling data transfer, each sequence of left (or right) channel byte is considered a frame with a buffer descriptor. The SSV bit is defined in the buffer descriptor command field and is automatically written to the transmit data command register just prior to or in synchronous with the first data byte of the frame being written. Note that the

number of bits per frame is a value other than an integral number of 8-bits by setting NUMBITS to a value other than 0.

Example. To send 20 bits/frame, set NUMBITS = 5 and read/write 4 bytes per frame. The transmit data must be left justified and the receive data must be right justified.

The transaction is terminated when the master has no more data to transmit. After the last bit is transferred, SCLK stops and $\overline{SS}$ and SSV returns to their default states. If TEOF is not set on the last byte, a transmit underrun error occurs at this point.

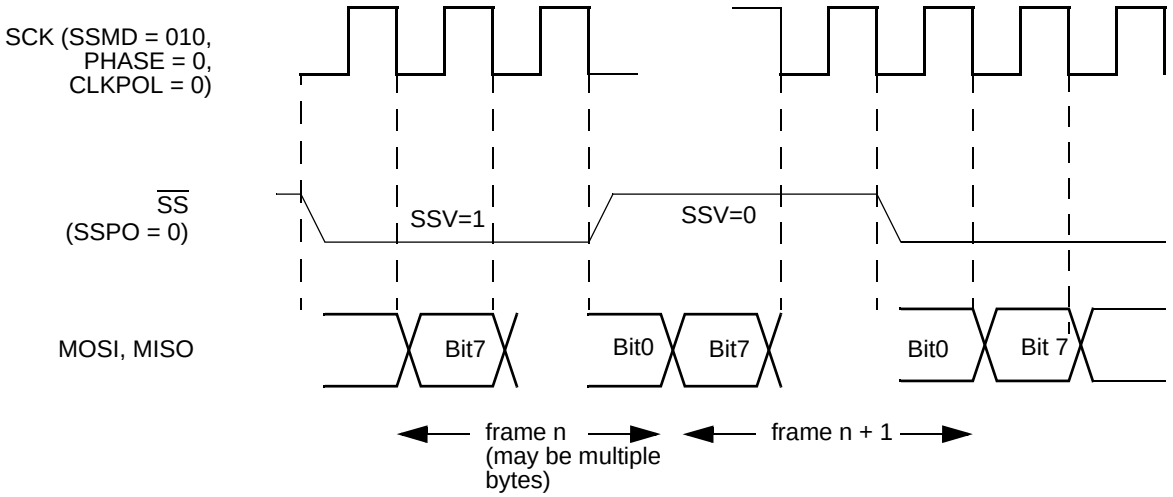


Figure 39. I²S Mode (SSMD = 010)

SPI Protocol Configuration

This section describes in detail how to configure the ESPI block for the SPI protocol. In the SPI protocol the master sources the SCK and asserts slave select signals to one or more slaves. The slave select signals are typically active Low.

SPI Master Operation

The ESPI block is configured for Master Mode operation by setting the MMEN bit = 1 in the ESPICTL Register. The SSMD field of the ESPI Mode register is set to 000 for SPI protocol mode. The PHASE, CLKPOL and WOR bits in the ESPICTL Register and the NUMBITS field in the ESPI Mode Register mustbe consistent with the Slave SPI devices. Typically for an SPI master SSIO = 1 and SSPO = 0. The appropriate GPIO pins are configured for the ESPI alternate function on the MOSI, MISO and SCK pins. The GPIO for the ESPI $\overline{SS}$ pin is configured in alternate function mode as well though software uses any GPIO pin(s) to drive one or more slave select lines. If the ESPI $\overline{SS}$ signal is not used to

drive a slave select the SSIO bit must still be set to 1 in a single master system. Figure 40 and Figure 41 displays the ESPI block configured as an SPI master.

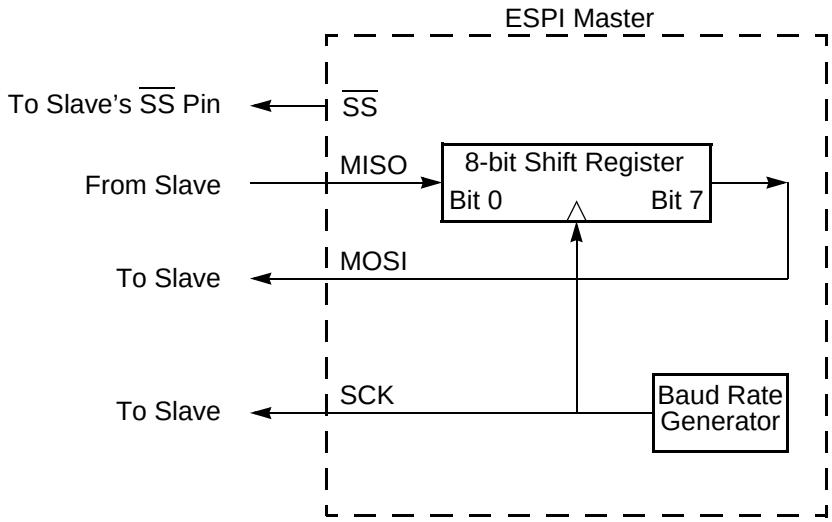


Figure 40. ESPI Configured as an SPI Master in a Single Master, Single Slave System

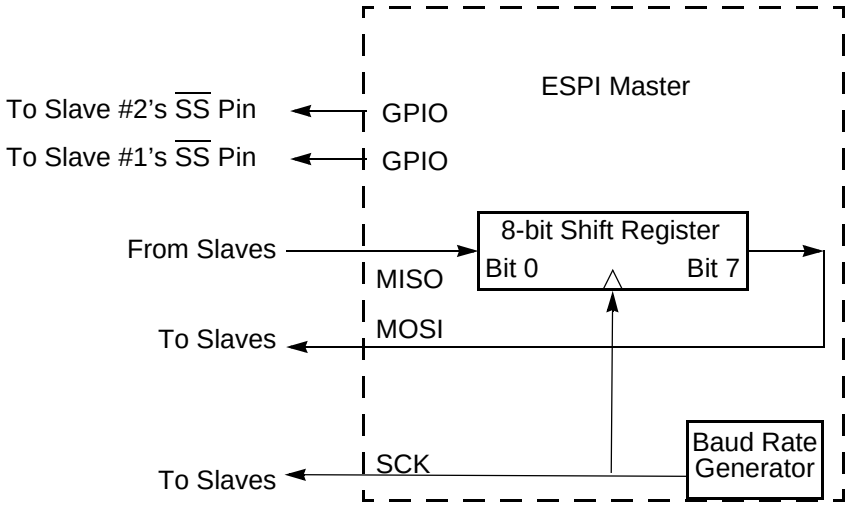


Figure 41. ESPI Configured as an SPI Master in a Single Master, Multiple Slave System

Multi-Master SPI Operation

In a multi-master SPI system, all SCK pins are tied together, all MOSI pins are tied together and all MISO pins are tied together. All SPI pins must be configured in open-drain mode to prevent bus contention. At any time, only one SPI device is configured as the master and all other devices on the bus are configured as slaves. The master asserts the $\overline{SS}$ pin on the selected slave. Then, the active master drives the clock and transmit data on the SCK and MOSI pins to the SCK and MOSI pins on the slave (including those slaves which are not enabled). The enabled slave drives data out its MISO pin to the MISO master pin.

When the ESPI is configured as a master in a multi-master SPI system, the $\overline{SS}$ pin must be configured as an input. The $\overline{SS}$ input signal on a device configured as a master must remain High. If the $\overline{SS}$ signal on the active master goes Low (indicating another master is accessing this device as a slave), a collision error flag is set in the ESPI Status Register. The slave select outputs on a master in a multi-master system must come from GPIO pins.

SPI Slave Operation

The ESPI block is configured for Slave Mode operation by setting the MMEN bit = 0 in the ESPICTL Register and setting the SSIO bit = 0 in the ESPIMODE Register. The SSMD field of the ESPI Mode Register is set to 00 for SPI protocol mode. The PHASE, CLKPOL and WOR bits in the ESPICTL Register and the NUMBITS field in the ESPIMODE Register must be set to be consistent with the other SPI devices. Typically for an SPI slave SSPO = 0.

If the slave has data to send to the master, the data must be written to the data register before the transaction starts (first edge of SCK when $\overline{SS}$ is asserted). If the data register is not written prior to the slave transaction, the MISO pin outputs all 1s.

Due to the delay resulting from synchronization of the $\overline{SS}$ and SCK input signals to the internal system clock, the maximum SCK baud rate which is supported in Slave Mode is the system clock frequency divided by 8. This rate is controlled by the SPI master.

Figure 42 displays the ESPI configuration in SPI Slave Mode.

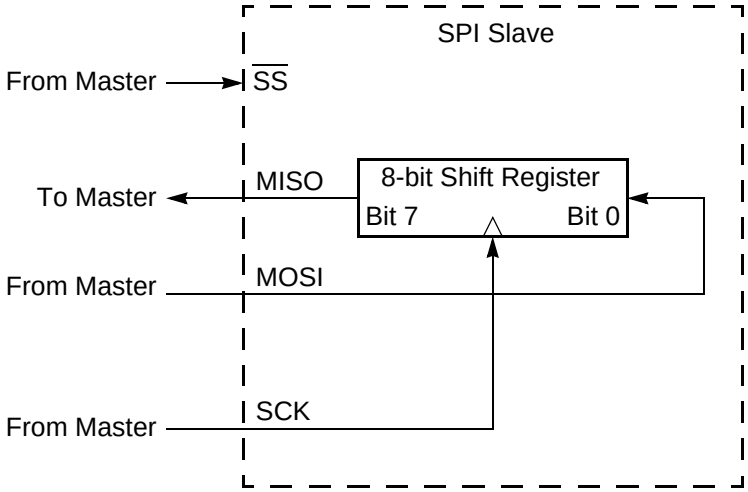


Figure 42. ESPI Configured as an SPI Slave

Error Detection

Error events detected by the ESPI block are described in this section. Error events generate an ESPI interrupt and set a bit in the ESPI Status Register. The error bits of the ESPI Status register are read/write 1 to clear.

Transmit Underrun

A transmit underrun error occurs for a master with SSMD = 10 or 11 when a character transfer completes and TDRE = 1. In these modes when a transmit underrun occurs the transfer is aborted (SCK will halt and SSV will be deasserted). For a master in SPI Mode (SSMD = 00), a transmit underrun is not signaled because SCK will pause and wait for the data register to be written.

In Slave Mode, a transmit underrun error occurs if TDRE = 1 at the start of a transfer. When a transmit underrun occurs in Slave Mode, ESPI transmits a character of all 1s.

A transmit underrun sets the TUND bit in the ESPI Status Register to 1. Writing 1 to TUND clears this error flag.

Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one master is trying to communicate at the same time (a multi-master collision) in SPI Mode. The mode fault is detected when the enabled master's $\overline{SS}$ input pin is asserted. For this to happen the control and mode registers must be configured with MMEN = 1, SSIO = 0 ($\overline{SS}$ is an input) and $\overline{SS}$ input = 0. A mode fault sets the COL bit in the ESPI Status Register to 1. Writing a 1 to COL clears this error flag.

Receive Overrun

A receive overrun error occurs when a transfer completes and the RDRF bit is still set from the previous transfer. A receive overrun sets the ROVR bit in the ESPI Status Register to 1. Writing 1 to ROVR clears this error flag. The Receive Data Register is not overwritten and will contain the data from the transfer which initially set the RDRF bit. Subsequent received data is lost until the RDRF bit is cleared.

Slave Mode Abort

In Slave Mode of operation if the $\overline{SS}$ pin deasserts before all bits in a character have been transferred, the transaction is aborted. When this condition occurs the ABT bit is set in the ESPI Status register. A slave abort error resets the slave control logic to the idle state.

A slave abort error is also asserted in Slave Mode, if BRGCTL = 1 and a BRG time-out occurs. When BRGCTL = 1 is in Slave Mode, it functions as a WDT monitoring the SCK signal. The BRG counter is reloaded every time a transition on SCK occurs while $\overline{SS}$ is asserted. The baud rate reload registers must be programmed with a value longer than the expected time between $\overline{SS}$ assertion and the first SCK edge, between SCK transitions while $\overline{SS}$ is asserted and between the last SCK edge and $\overline{SS}$ deassertion. A time-out indicates the master is stalled or disabled. Writing 1 to ABT clears this error flag.

ESPI Interrupts

ESPI has a single interrupt output which is asserted when any of the TDRE, TUND, COL, ABT, ROVR, or RDRF bits are set in the ESPI Status Register. The interrupt is a pulse (duration of one system clock) generated when any one of the source bits initially set. The TDRE and RDRF interrupts are enabled/disabled through the Data Interrupt Request Enable (DIRQE) bit of the ESPI Control Register.

A transmit interrupt is asserted by the TDRE status bit when the ESPI block is enabled and the DIRQE bit is set. The TDRE bit in the status register is cleared automatically when the Transmit Data Register is written or the ESPI block is disabled. When the Transmit Data register value is loaded into the shift register to start a new transfer, the TDRE bit will be set again causing a new transmit interrupt. If information is being received but not transmitted the transmit interrupts are eliminated by selecting Receive Only Mode (ESPIEN1,0 = 01). A master operates in Receive Only mode; however, a write to the ESPI (Transmit) Data Register is still required to initiate the transfer of a character.

A receive interrupt is generated by the RDRF status bit when the ESPI block is enabled; the DIRQE bit is set and a character transfer completes. At the end of the character transfer, the contents of the shift register is transferred into the Receive Data register, causing the RDRF bit to assert. The RDRF bit is cleared when the Receive Data register is read. If information is being transmitted but not received by the software application, the receive interrupt is eliminated by selecting Transmit Only mode (ESPIEN1,0 = 10) in either Master or Slave modes.

ESPI error interrupts occur if any of the TUND, COL, ABT and ROVR bits in the ESPI Status register are set. These bits are cleared by writing a 1 to the corresponding bit.

If the ESPI is disabled (ESPIEN1,0 = 00), an ESPI interrupt is generated by a BRG time-out. This timer function must be enabled by setting the BRGCTL bit in the ESPICTL Register. This timer interrupt does not set any of the bits of the ESPI Status register.

DMA Interface

The assertion of the TDRE and RDRF signals generate transmit and receive DMA requests (SPITxReq, SPIRxReq), allowing data movement to be handled by a DMA Controller rather than directly by software. The DMA acknowledges these requests through the SPITxAck and SPIRxAck signals). Inputs allow the SSV and TEOF bits of the Transmit Data Command register to be controlled by the DMA. The SPITxReqEOF and SPIRxReqEOF outputs to the DMA provides an indication that $\overline{SS}$ has deasserted (transaction complete).

If the software application is moving data in only one direction, the ESPIEN1,0 bits are set to 10 or 01, allowing a single DMA Channel to control the ESPI data transfer. For a master, the valid options are transmit only or transmit-receive. For a slave, all options are valid. When a slave is operating in Receive Only Mode, it will transmit characters of all 1s.

DMA Descriptors

For ESPI Transmit DMA descriptors, the 4-bit CMDSTAT field of the descriptor is in the format shown in Table 99. The SSV bit in the Master's transmit buffer descriptor CMDSTAT field controls the ESPI $\overline{SS}$ output. The SSV bit in the descriptor is transferred to the SSV bit in the ESPI Data Command register with the first byte of the buffer. If the EOF bit is set in the DMA descriptor control word, the end of frame signal from the DMA (EOFSync) will assert coincident with writing the last byte in the buffer to the ESPI Data register, setting the TEOF bit of the ESPI Data Command register. After this last byte has been transferred, the Master's $\overline{SS}$ output will deassert and the SSV and TEOF bits in the Data Command register will be cleared. The CMDSTAT field in ESPI Receive DMA Descriptors has no function.

Table 99. ESPI Tx DMA Descriptor Command Field

Reserved	Reserved	Reserved	SSV

For ESPI DMA descriptors, the 4-bit frame status field of the descriptor has the format shown in Tables 100 and 101.

Table 100. ESPI Tx DMA Descriptor Status Field

0	0	COL	TUND

Table 101. ESPI Rx DMA Descriptor Status Field

0	RSS	ABT	ROVR

In Table 101, RSS is the value of SS associated with the last byte written. This value is useful in I^2S Mode to distinquish left/right channel data. For a description of the COL, TUND, ABT and ROVR bits listed in the above tables, see the ESPI Status Register (ESPISTAT) on page 198.

ESPI Baud Rate Generator

In ESPI Master Mode, the BRG creates a lower frequency serial clock (SCK) for data transmission synchronization between the Master and the external Slave. The input to the BRG is the system clock. The ESPI Baud Rate High and Low Byte registers combine to form a 16-bit reload value, BRG[15:0], for the ESPI BRG. The ESPI baud rate is calculated using the following equation:

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

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

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

1. Disable the ESPI by clearing the ESPIEN1,0 bits in the ESPI Control register.
2. Load the appropriate 16-bit count value into the ESPI Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and associated interrupt by setting the BRGCTL bit in the ESPI Control register to 1.

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

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

ESPI Control Register Definitions

ESPI Data Register

The ESPI Data Register, shown in Table 102, addresses both the outgoing Transmit Data register and the incoming Receive Data register. Reads from the ESPI Data register return the contents of the Receive Data register. The Receive Data register is updated with the contents of the shift register at the end of each transfer. Writes to the ESPI Data register load the Transmit Data register unless TDRE = 0. Data is shifted out starting with bit 7. The last bit received resides in bit position 0.

With the ESPI configured as a Master, writing a data byte to this register initiates the data transmission. With the ESPI configured as a Slave, writing a data byte to this register loads the shift register in preparation for the next data transfer with the external Master. In either the Master or Slave modes, if TDRE = 0, writes to this register are ignored.

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

Table 102. ESPI Data Register (ESPIDATA)

Bits	7	6	5	4	3	2	1	0
Field	DATA							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E260h							

Bit	Description
[7:0] DATA	**Data** Transmit and/or receive data. Writes to the ESPIDATA Register load the shift register. Reads from the ESPIDATA Register return the value of the Receive Data Register.

ESPI Transmit Data Command Register

The ESPI Transmit Data Command Register, shown in Table 103, provides control of the $\overline{SS}$ pin when it is configured as an output (Master Mode). The TEOF and SSV bits are controlled by the DMA interface as well as by a bus write to this register.

Table 103. ESPI Transmit Data Command Register (ESPITDCR)

Bits	7	6	5	4	3	2	1	0
Field	Reserved						TEOF	SSV
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R/W	R/W
Addr	FF_E261h							

Bit	Description
[7:2]	**Reserved** These bits are reserved and are read-only.
[1] TEOF	**Transmit End of Frame** This bit is used in Master Mode to indicate that the data in the Transmit Data Register is the last byte of the transfer or frame. When the last byte has been sent $\overline{SS}$ (and SSV) change state and TEOF automatically clears. 0 = The data in the Transmit Data Register is not the last character in the message. 1 = The data in the Transmit Data Register is the last character in the message.
[0] SSV	**Slave Select Value** When SSIO = 1, writes to this register controls the value output on the $\overline{SS}$ pin. See the SSMD field of the ESPI Mode Register section on page 197 for more details.

ESPI Control Register

The ESPI Control Register, shown in Table 104, configures the ESPI for transmit and receive operations.

Table 104. ESPI Control Register (ESPICTL)

Bits	7	6	5	4	3	2	1	0
Field	DIRQE	ESPIEN1	BRGCTL	PHASE	CLKPOL	WOR	MMEN	ESPIEN0
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E262h							

Bit	Description
[7] DIRQE	**Data Interrupt Request Enable** This bit is used to disable or enable data (TDRE and RDRF) interrupts. Disabling the data interrupts is required when controlling data transfer by DMA or polling. Error interrupts are not disabled. To block all ESPI interrupt sources, clear the ESPI interrupt enable bit in the Interrupt Controller. 0 = TDRE and RDRF assertions do not cause an interrupt. Use this setting if controlling data transfer through DMA or by software polling of TDRE and RDRF. The TUND, COL, ABT and ROVR bits cause an interrupt. 1 = TDRE and RDRF assertions will cause an interrupt. TUND, COL, ABT and ROVR will also cause interrupts. Use this setting if controlling data transfer through interrupt handlers.
[6,0] ESPIEN1, ESPIEN0	**ESPI Enable and Direction Control** 00 = ESPI block is disabled. BRG is used as a general purpose timer by setting BRGCTL = 1. 01 = Receive Only Mode. Use this setting if the software application is receiving data but not sending. The TDRE asserts; however, the transmit interrupt and DMA requests do not assert. In Slave Mode, the transmitted data will be all 1s. In Master Mode software must still write to the Transmit Data register to initiate the transfer. 10 = Transmit Only Mode. Use this setting in Master or Slave Mode when the software application is sending data but not receiving. RDRF will assert, but receive interrupt and DMA requests do not occur. 11 = Transmit/Receive Mode. Use this setting if the software application is both sending and receiving information. Both TDRE and RDRF will be active.
[5] BRGCTL	**Baud Rate Generator Control** The function of this bit depends upon ESPIEN1,0. When ESPIEN1,0 = 00, this bit allows enabling the BRG to provide periodic interrupts. If the ESPI is disabled (ESPIEN1, ESPIEN0 = 00): 0 = The BRG timer function is disabled. Reading the Baud Rate High and Low registers returns the BRG reload value. 1 = The BRG timer function and time-out interrupt are enabled. Reading the Baud Rate High and Low registers returns the BRG Counter value. If the ESPI is enabled: 0 = Reading the Baud Rate High and Low registers returns the BRG reload value. If MMEN = 1, the BRG is enabled to generate SCK. If MMEN = 0, the BRG is disabled. 1 = Reading the Baud Rate High and Low registers returns the BRG Counter value. If MMEN = 1, the BRG is enabled to generate SCK. If MMEN = 0, the BRG is enabled to provide a Slave SCK time-out. See the Slave Mode Abort section on page 190 for further details.
[4] PHASE	**Phase Select** Sets the phase relationship of the data to the clock. For more information about operation of the PHASE bit, see the ESPI Clock Phase and Polarity Control section on page 181.
[3] CLKPOL	**Clock Polarity** 0 = SCK idles Low (0). 1 = SCK idles High (1).

Bit	Description (Continued)
[2] WOR	**Wire-OR (Open-Drain) Mode Enabled** 0 = ESPI signal pins not configured for open-drain. 1 = All four ESPI signal pins (SCK, SS, MISO, MOSI) configured for open-drain function. This setting is used for Multi-Master and/or Multi-Slave configurations.
[1] MMEN	**ESPI Master Mode Enable** This bit controls the data I/O pin selection and SCK direction. 0 = Data-out on MISO, data-in on MOSI (used in SPI Slave Mode), SCK is an input. 1 = Data-out on MOSI, data-in on MISO (used in SPI Master Mode), SCK is an output.

Caution: If reading the counter one byte at a time while the BRG is counting keep in mind that the values will not be in sync. It is recommended to read the counter using word (2-byte) reads.

ESPI Mode Register

The ESPI Mode Register, shown in Table 105, configures the character bit width and mode of the ESPI I/O pins.

Table 105. ESPI Mode Register (ESPIMODE)

Bits	7	6	5	4	3	2	1	0
Field	SSMD			NUMBITS[2:0]			SSIO	SSPO
RESET	000			000			0	0
R/W	R/W			R/W			R/W	R/W
Addr	FF_E263h							

Bit	Description
[7:5] SSMD	**Slave Select Mode** This field selects the behavior of $\overline{SS}$ as a framing signal. For a detailed description of these modes, see the Slave Select section on page 179. **000 = SPI Mode** When SSIO = 1, the $\overline{SS}$ pin is driven directly from the SSV bit in the Transmit Data Command register. The Master software or DMA must set SSV (or a GPIO output if the $\overline{SS}$ pin is not connected to the appropriate Slave) to the asserted state prior to or on the same clock cycle with which the Transmit Data Register is written with the initial byte. At the end of a frame (after the last RDRF event), SSV is deasserted by software. Alternatively, SSV is automatically deasserted by hardware if the TEOF bit in the Transmit Data Command register is set when the last transmit byte is loaded. In SPI Mode, SCK is active only for data transfer (one clock cycle per bit transferred). **001 = LOOPBACK Mode** When ESPI is configured as Master (MMEN = 1) the outputs are deasserted and data is looped from shift register out to shift register in. When ESPI is configured as a Slave (MMEN = 0) and $\overline{SS}$ in asserts, MISO (Slave output) is tied to MOSI (Slave input) to provide an a remote loop back (echo) function. **010 = I^2S Mode** In this mode, the value from SSV will be output by the Master on the $\overline{SS}$ pin one SCK period before the data and will remain in that state until the start of the next frame. Typically this mode is used to send back-to-back frames with $\overline{SS}$ alternating on each frame. A frame boundary is indicated in the Master when SSV changes. A frame boundary is detected in the Slave by $\overline{SS}$ changing state. The $\overline{SS}$ framing signal will lead the frame by one SCK period. In this mode SCK will run continuously, starting with the initial $\overline{SS}$ assertion. Frames will run back-to-back as long as software/DMA continue to provide data. The I^2S protocol (Inter IC Sound) is used to carry left and right channel audio data with the $\overline{SS}$ signal indicating which channel is being sent. In Slave Mode, the change in state of $\overline{SS}$ (Low to High or High to Low) will trigger the start of a transaction on the next SCK cycle.

Bit	Description (Continued)
[4:2] NUMBITS[2:0]	**Number of Data Bits Per Character to Transfer** This field contains the number of bits to shift for each character transfer. For information about valid bit positions when the character length is less than 8-bits, see the ESPI Data Register section on page 193. 000 = 8 bits 001 = 1 bit 010 = 2 bits 011 = 3 bits 100 = 4 bits 101 = 5 bits 110 = 6 bits 111 = 7 bits
[1] SSIO	**Slave Select I/O** This bit controls the direction of the $\overline{SS}$ pin. In single Master Mode, SSIO is set to 1 unless a separate GPIO pin is being used to provide the $\overline{SS}$ output function. In the SPI Slave or Multi-Master configuration SSIO is set to 0. 0 = $\overline{SS}$ pin configured as an input (SPI Slave and Multi-Master modes) 1 = $\overline{SS}$ pin configured as an output (SPI single Master Mode)
[0] SSPO	**Slave Select Polarity** This bit controls the polarity of the $\overline{SS}$ pin. 0 = $\overline{SS}$ is active Low. (SSV = 1 corresponds to $\overline{SS}$ = 0) 1 = $\overline{SS}$ is active High. (SSV = 1 corresponds to $\overline{SS}$ = 1)

ESPI Status Register

The ESPI Status Register, shown in Table 106, indicates the current state of the ESPI. All bits revert to their Reset state, if the ESPI is disabled.

Table 106. ESPI Status Register (ESPISTAT)

Bits	7	6	5	4	3	2	1	0
Field	TDRE	TUND	COL	ABT	ROVR	RDRF	TFST	SLAS
RESET	0	0	0	0	0	0	0	1
R/W	R	R/W*	R/W*	R/W*	R/W*	R	R	R
Addr	FF_E264h							
R/W* = Read access. Write a 1 to clear the bit to 0.								

Bit	Description
[7] TDRE	**Transmit Data Register Empty** 0 = Transmit Data Register is full or ESPI is disabled. 1 = Transmit Data Register is empty. A write to the ESPI (Transmit) Data register clears this bit.

Bit	Description (Continued)
[6] TUND	**Transmit Underrun** 0 = A Transmit Underrun error has not occurred. 1 = A Transmit Underrun error has occurred.
[5] COL	**Collision** 0 = A Multi-Master collision (mode fault) has not occurred. 1 = A Multi-Master collision (mode fault) has been detected.
[4] ABT	**Slave Mode Transaction Abort** This bit is set if the ESPI is configured in Slave Mode, a transaction is occurring and $\overline{SS}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the ESPIMODE Register. This bit is also be set in Slave Mode by an SCK monitor time-out (MMEN = 0, BRGCTL = 1). 0 = A Slave Mode transaction abort has not occurred. 1 = A Slave Mode transaction abort has been detected.
[3] ROVR	**Receive Overrun** 0 = A Receive Overrun error has not occurred. 1 = A Receive Overrun error has occurred.
[2] RDRF	**Receive Data Register Full** 0 = Receive Data register is empty. 1 = Receive Data register is full. A read from the ESPI (Receive) Data register clears this bit.
[1] TFST	**Transfer Status** 0 = No data transfer is currently in progress. 1 = Data transfer is currently in progress.
[0] SLAS	**Slave Select** Reading this bit returns the current value of the $\overline{SS}$ exclusive-OR'd with the SSPO bit. 0 = $\overline{SS}$ pin is Low, if SSPO = 0, $\overline{SS}$ pin is High if SSPO = 1 ($\overline{SS}$ is asserted). 1 = $\overline{SS}$ pin is High, if SSPO = 0, $\overline{SS}$ pin is Low if SSPO = 1 ($\overline{SS}$ is deasserted).

ESPI State Register

The ESPI State Register, shown in Table 107, provides observability of the ESPI clock, data and internal state. Table 108 describes the ESPI State Machine values.

Table 107. ESPI State Register (ESPISTATE)

Bits	7	6	5	4	3	2	1	0
Field	SCKI	SDI	ESPISTATE					
RESET	0	0	0					
R/W	R	R	R					
Addr	FF_E265h							

Bit	Description
[7] SCKI	**Serial Clock Input** This bit reflects the state of the serial clock pin. 0 = The SCK input pin is Low 1 = The SCK input pin is High
[6] SDI	**Serial Data Input** This bit reflects the state of the serial data input (MOSI or MISO depending on the MMEN bit). 0 = The serial data input pin is Low. 1 = The serial data input pin is High.
[5:0] ESPISTATE	**ESPI State Machine** Indicates the current state of the internal ESPI State Machine. This information is intended for manufacturing test. The state values may change in future hardware revisions and are not intended to be used by a software driver. Table 108 defines the valid states.

Table 108. ESPISTATE Values and Description

ESPISTATE Value	Description
00_0000	Idle
00_0001	Slave Wait For SCK
00_0010	I2S Slave Mode start delay
00_0011	I2S Slave Mode start delay
01_0000	SPI Master Mode start delay
11_0001	I2S Master Mode start delay
11_0010	I2S Master Mode start delay
10_1110	Bit 7 Receive
10_1111	Bit 7 Transmit

Table 108. ESPISTATE Values and Description (Continued)

ESPISTATE Value	Description
10_1100	Bit 6 Receive
10_1101	Bit 6 Transmit
10_1010	Bit 5 Receive
10_1011	Bit 5 Transmit
10_1000	Bit 4 Receive
10_1001	Bit 4 Transmit
10_0110	Bit 3 Receive
10_0111	Bit 3 Transmit
10_0100	Bit 2 Receive
10_0101	Bit 2 Transmit
10_0010	Bit 1 Receive
10_0011	Bit 1 Transmit
10_0000	Bit 0 Receive
10_0001	Bit 0 Transmit

ESPI Baud Rate High and Low Byte Registers

The ESPI Baud Rate High and Low Byte registers, shown in Tables 109 and 110, combine to form a 16-bit reload value, BRG[15:0], for the ESPI Baud Rate Generator. The ESPI baud rate is calculated using the following equation:

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

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

When the ESPI function is disabled, the BRG functions as a basic 16-bit timer with interrupt on time-out.

Observe the following procedure to configure the BRG as a general purpose timer with interrupt on time-out:

1. Disable the ESPI by setting ESPIEN[1:0] = 00 in the SPI Control register.
2. Load the appropriate 16-bit count value into the ESPI Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and associated interrupt by setting the BRGCTL bit in the ESPI Control register to 1.

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

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

Table 109. ESPI Baud Rate High Byte Register (ESPIBRH)

Bits	7	6	5	4	3	2	1	0
Field	BRH							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_E266h							

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

Table 110. ESPI Baud Rate Low Byte Register (ESPIBRL)

Bits	7	6	5	4	3	2	1	0
Field	BRL							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/w
Addr	FF_E267h							

Bit	Description
[7:0] BRL	**ESPI Baud Rate Low Byte** Least significant byte, BRG[7:0], of the ESPI Baud Rate Generator's reload value.

I²C Master/Slave Controller

The I²C Master/Slave Controller makes the ZNEO® Z16F Series bus compatible with the I²C protocol. The I²C bus consists of the serial data signal (SDA) and a serial clock (SCL) signal bidirectional lines.

Features of the I²C Controller include:

- Operates in Master/Slave or Slave Only modes
- Supports arbitration in a Multi-Master environment (Master/Slave Mode)
- Supports data rates up to 400 kbps
- 7-bit or 10-bit Slave address recognition (interrupt only on address match)
- Optional general call address recognition
- Optional digital filter on receive SDA and SCL lines
- Optional interactive receive mode allows software interpretation of each received address and/or data byte before acknowledging
- Unrestricted number of data bytes per transfer
- Baud Rate Generator (BRG) is used as a general purpose timer with interrupt if the I²C controller is disabled

Architecture

Figure 43 displays the architecture of the I²C Controller.

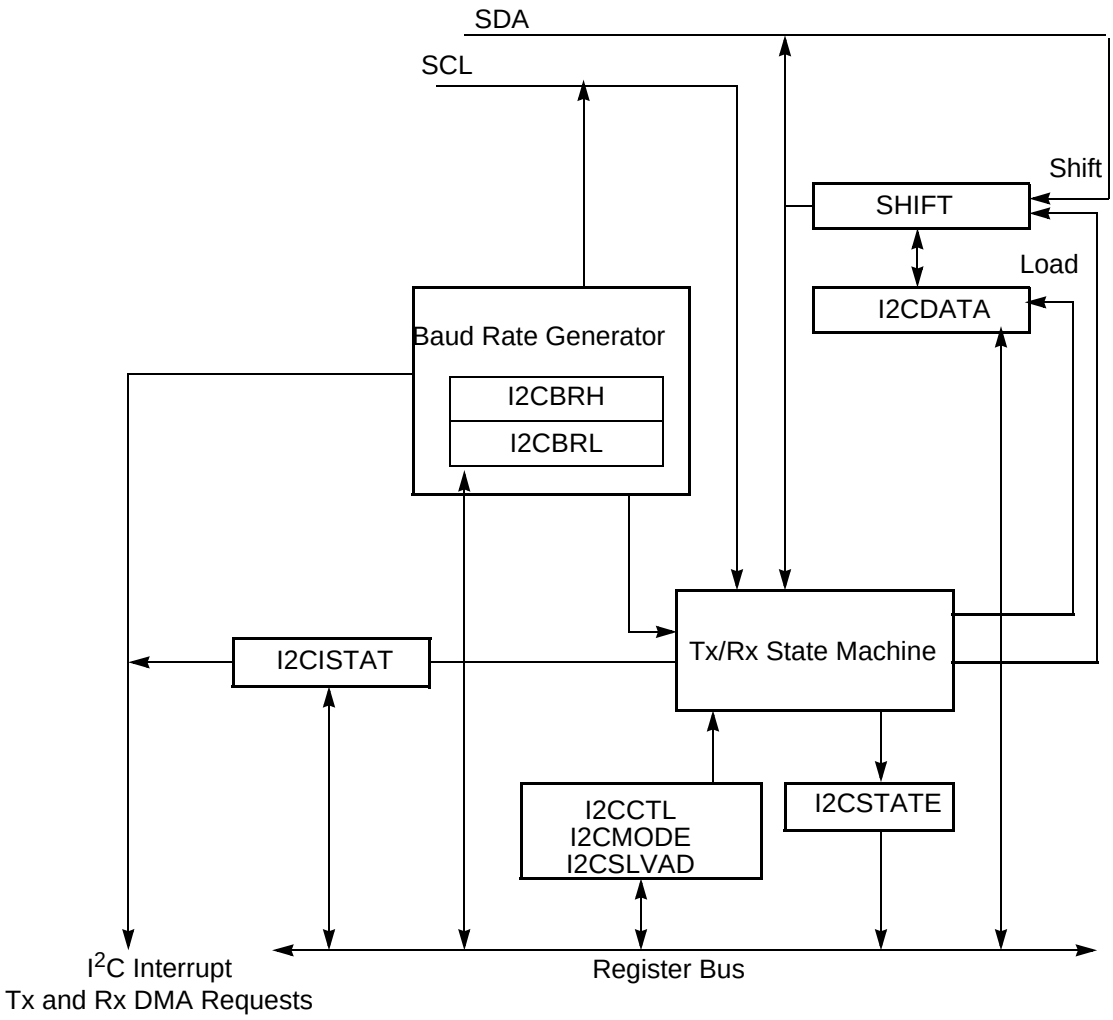


Figure 43. I2C Controller Block Diagram

I²C Master/Slave Controller Registers

Table 111 summarizes the I²C Master/Slave Controller software-accessible registers.

Table 111. I²C Master/Slave Controller Registers

Name	Abbreviation	Description
I²C Data	I2CDATA	Transmit/Receive Data Register.
I²C Interrupt Status	I2CISTAT	Interrupt Status Register.
I²C Control	I2CCTL	Control Register–basic control functions.
I²C Baud Rate High	I2CBRH	High byte of baud rate generator initialization value.
I²C Baud Rate Low	I2CBRL	Low byte of baud rate generator initialization value.
I²C State	I2CSTATE	State Register.
I²C Mode	I2CMODE	Selects MASTER or SLAVE modes, 7-bit or 10-Bit Address. Configure address recognition, Defines Slave Address bits [9:8].
I²C Slave Address	I2CSLVAD	Defines Slave Address bits [7:0]

Comparison with Master Mode only I²C Controller

Porting code written for the Master-only I²C Controller found on other Z8 Encore!® parts to the I²C Master/Slave Controller is straightforward. The I2CDATA, I2CCTL, I2CBRH and I2CBRL Register definitions are not changed.

The differences between the Master-only I²C Controller and I²C Master/Slave Controller designs are:

- The Status Register (I2CSTATE) from the Master-only I²C Controller is split into the Interrupt Status (I2CISTAT) Register and the State (I2CSTATE) Register because there are more interrupt sources. The ACK, 10b, TAS (now called AS) and DSS (now called DS) bits formerly in the status register are moved to the state register.
- The I2CSTATE Register is called as I2CDST (Diagnostic State) Register in the Master Only Mode version. The I2CDST Register provided diagnostic information. The I2CSTATE Register contains status and state information that are useful to software in operational mode.
- The I2CMODE Register is called as I2CDIAG (Diagnostic Control) Register in the Master Only Mode version. The I2CMODE Register provides control for Slave modes of operation as well as the most significant two bits of the 10-bit Slave address.
- The I2CSLVAD Register is added for programming the Slave address.
- The ACKV bit in the I2CSTATE Register enables the Master to verify the acknowledge from the Slave before sending the next byte.

- Support for multi-master environments. If arbitration is lost when operating as a Master, the ARBLST bit in the I2CISTAT Register is set and the mode automatically switches to Slave Mode.

Operation

The I^2C Master/Slave Controller operates in either Slave Only Mode or Master/Slave Mode with Master arbitration. In Master/Slave Mode, it is used as the only Master on the bus or as one of several Masters on the bus with arbitration. In a multi-Master environment, the controller switches from Master to Slave Mode on losing arbitration.

Though slave operation is fully supported in Master/Slave Mode, if a device is intended to operate only as a slave, the Slave Only Mode is selected. In Slave Only Mode, the device does not initiate a transaction even if software inadvertently sets the Start bit.

SDA and SCL Signals

I^2C sends all addresses, data and acknowledge signals over the SDA line, the most-significant bit first. SCL is the clock for the I^2C bus. When the SDA and SCL pin alternate functions are selected for their respective GPIO ports, the pins are automatically configured for open-drain operation.

The Master is responsible for driving the SCL clock signal. During the Low period of the clock, a Slave holds the SCL signal Low to suspend the transaction if it is not ready to proceed. The Master releases the clock at the end of the Low period and notices that the clock remains Low instead of returning to a High level. When the Slave releases the clock, the I^2C Master 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 address, data or acknowledge, the SDA signal changes in the middle of the Low period of SCL. When receiving address, data, or acknowledge, the SDA signal is sampled in the middle of the High period of SCL.

A low-pass digital filter is applied to the SDA and SCL receive signals by setting the filter enable (FILTEN) bit in the I^2C Control Register. When the filter is enabled, any glitch, which is less than a system clock period in width is rejected. This filter must be enabled when running in I^2C Fast Mode (400 kbps) and is also used at lower data rates.

I^2C Interrupts

The I^2C Controller contains multiple interrupt sources that are combined into one interrupt request signal to the interrupt controller. If the I^2C Controller is enabled, the source of the interrupt is determined by bits, which are set in the I2CISTAT Register. If the I^2C Controller is disabled, the BRG Controller is used to generate general-purpose timer interrupts.

Each interrupt source other than the baud rate generator interrupt has an associated bit in the I2CISTAT Register, which clears automatically when software reads the register or performs some other task such as reading or writing the data register.

Transmit Interrupts

Transmit interrupts (TDRE bit = 1 in I2CISTAT) occur under the following conditions:

- The Transmit Data Register is empty and the TXI bit = 1 in the I^2C Control Register
- The I^2C Controller is enabled, with any one of the following operations:
  - The first bit of a 10-bit address is shifted out
  - The first bit of the final byte of an address is shifted out and the RD bit is deasserted
  - The first bit of a data byte is shifted out

Writing to the I^2C Data Register always clears the TRDE bit to 0.

Receive Interrupts

Receive interrupts (RDRF bit = 1 in I2CISTAT) occur when a byte of data has been received by the I^2C Controller. The RDRF bit is cleared by reading from the I^2C Data Register. If the RDRF interrupt is not serviced prior to the completion of the next receive byte, the I^2C Controller holds SCL Low during the last data bit of the next byte until RDRF is cleared to prevent receive overruns. A receive interrupt does not occur when a Slave receives an address byte or for data bytes following a Slave address that did not match. An exception is if the interactive receive mode (IRM) bit is set in the I2CMODE Register in which case receive interrupts occur for all receive address and data bytes in Slave Mode.

Slave Address Match Interrupts

Slave address match interrupts (SAM bit = 1 in I2CISTAT) occur when the I^2C Controller is in Slave Mode and an address is received which matches the unique Slave address. The General Call Address (0000_0000) and STARTBYTE (0000_0001) are recognized if the GCE bit = 1 in the I2CMODE Register. Software verifies the RD bit in the I2CISTAT Register to determine if the transaction is a read or write transaction. The General Call Address and STARTBYTE addresses are also distinguished by the RD bit. The general call address (GCA) bit of the I2CISTAT Register indicates whether the address match occurred on the unique Slave address or the General Call/STARTBYTE address. The SAM bit clears automatically when the I2CISTAT Register is read.

If configured using the MODE[1:0] field of the I^2C Mode Register for 7-bit slave addressing, the most significant 7 bits of the first byte of the transaction are compared against the SLA[6:0] bits of the Slave Address Register. If configured for 10-bit slave addressing, the

first byte of the transaction is compared against {11110,SLA[9:8],R/$\overline{W}$} and the second byte is compared against SLA[7:0].

Arbitration Lost Interrupts

Arbitration Lost interrupts (ARBLST bit = 1 in I2CISTAT) occur when the I^{2}C Controller is in Master Mode and loses arbitration (outputs a 1 on SDA and receives a 0 on SDA). The I^{2}C Controller switches to Slave Mode when this occurs. This bit clears automatically when the I2CISTAT Register is read.

Stop/Restart Interrupts

A Stop/Restart event interrupt (SPRS bit = 1 in I2CISTAT) occurs when the I^{2}C Controller is in Slave Mode and a Stop or Restart condition is received, indicating the end of the transaction. The RSTR bit in the I^{2}C State Register indicates whether the bit was set due to a Stop or Restart condition. When a Restart occurs, a new transaction by the same Master is expected to follow. This bit is cleared automatically when the I2CISTAT Register is read. The Stop/Restart interrupt only occurs on a selected (address match) slave.

Not Acknowledge interrupts

Not Acknowledge interrupts (NCKI bit = 1 in I2CISTAT) occur in Master Mode when a Not Acknowledge is received or sent by the I^{2}C Controller and the Start or Stop bit is not set in the I^{2}C Control Register. In Master Mode the Not Acknowledge interrupt clears by setting the Start or Stop bit. When this interrupt occurs in Master Mode, the I^{2}C Controller waits until it is cleared before performing any action. In Slave Mode, the Not Acknowledge interrupt occurs when a Not Acknowledge is received in response to the data sent. The NCKI bit clears in Slave Mode when software reads the I2CISTAT Register.

General Purpose Timer Interrupt from Baud Rate Generator

If the I^{2}C Controller is disabled (IEN bit in the I2CCTL Register = 0) and the BIRQ bit in the I2CCTL Register = 1, an interrupt is generated when the BRG counts down to 1. The BRG reloads and continues counting, providing a periodic interrupt. None of the bits in the I2CISTAT Register are set, allowing the BRG in the I^{2}C Controller to be used as a general purpose timer when the I^{2}C Controller is disabled.

Start and Stop Conditions

The Master generates the Start and Stop conditions to start or end a transaction. To start a transaction, the I^{2}C Controller generates a Start condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the I^{2}C 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 events occur when the Start and Stop bits in the I^{2}C Control Register are written by software to begin or end a transaction. Any byte transfer currently under way finishes, including the acknowledge phase before the Start or Stop condition occurs.

Software Control of I2C Transactions

The I2C Controller is configured using the I2C Control and I2C Mode registers. The MODE[1:0] field of the I2C Mode Register allows configuring the I2C Controller for Master/Slave or Slave Only Mode and configures the slave for 7-bit or 10-Bit Address recognition. The baud rate High and Low Byte Registers must be programmed for the I2C baud rate in Slave Mode as well as in Master Mode. In Slave Mode, the baud rate value programmed must match the master's baud rate within ±25% for proper operation.

Master/Slave Mode is used for:

- Master Only operation in a single master, one or more slave I2C system
- Master/Slave in a multi-master, multi-slave I2C system
- Slave Only operation in an I2C system

In Slave Only Mode, the Start bit of the I2C Control Register is ignored (software cannot initiate a master transaction by accident). This restricts the operation to Slave Only Mode and prevents accidental operation in Master Mode.

Software controls I2C transactions by enabling the I2C Controller interrupt in the interrupt controller or by polling the I2C Status Register.

To use interrupts, the I2C interrupt must be enabled in the Interrupt Controller and followed by executing an EI instruction. The TXI bit in the I2C Control Register must be set to enable transmit interrupts. An I2C interrupt service routine then verifies the I2C Status Register to determine the cause of the interrupt.

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

Master Transactions

The following sections describe the Master read and write transactions to both 7- and 10-bit slaves.

Master Arbitration

If a Master loses arbitration during the address byte, it releases the SDA line, switches to Slave Mode and monitors the address to determine if it is selected as a Slave. If a Master loses arbitration during a transmit data byte, it releases the SDA line and waits for the next Stop or Start condition.

The Master detects a loss of arbitration when a 1 is transmitted but a 0 is received from the bus in the same bit time. This loss occurs if more than one Master is simultaneously accessing the bus. Loss of arbitration occurs during the address phase (two or more Mas-

ters accessing different Slaves) or during the data phase when the Masters are attempting to write different data to the same Slave.

When a Master loses arbitration, software is informed by means of the Arbitration Lost interrupt. Software repeats the same transaction again at a later time.

A special case occurs when a slave transaction starts just before software attempts to start a new master transaction by setting the Start bit. In this case the state machine enters the slave states before the Start bit is set and the I^2C Controller does not arbitrate. If a slave address match occurs and the I^2C Controller receives or transmits data, the Start bit is cleared and an Arbitration Lost interrupt is asserted. Software minimizes the chance of this occurring by checking the BUSY bit in the I2CSTATE Register before initiating a master transaction. If a slave address match does not occur, the Arbitration Lost interrupt does not occur and the Start bit is not cleared. The I^2C Controller initiates the master transaction after the I^2C bus is no longer busy.

Master Address Only Transactions

It is sometimes appropriate to perform an address-only transaction to determine if a particular Slave device is able to respond. This transaction is performed by monitoring the ACKV bit in the I2CSTATE Register after the address has been written to the I2CDATA Register and the Start bit has been set. After ACKV is set, the ACK bit in the I2CSTATE Register determines if the Slave is able to communicate. The Stop bit must be set in the I2CCTL Register to terminate the transaction without transferring data. For a 10-bit slave address, if the first address byte is acknowledged, the second address byte must also be sent to determine if the appropriate slave is responding.

Another approach is to set both the Stop and Start bits (for sending a 7-bit address). After both bits are cleared (7-bit address has been sent and transaction is complete), the ACK bit is read to determine if the slave is acknowledged. For a 10-bit slave, set the Stop bit after the second TDRE interrupt (second address byte is being sent).

Master Transaction Diagrams

In the following transaction diagrams, shaded regions indicate data transferred from the Master to the Slave and unshaded regions indicate data transferred from the Slave to the Master. The transaction field labels are defined as follows:

S: Start

W: Write

A: Acknowledge

$\overline{A}$: Not Acknowledge

P: Stop

Master Write Transaction with a 7-Bit Address

Figure 44 displays the data transfer format from a Master to a 7-bit addressed Slave.

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

Figure 44. Data Transfer Format, Master Write Transaction with 7-Bit Addressing

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

1. Software initializes the MODE field in the I^2C Mode Register for Master/Slave Mode with either 7-bit or 10-bit slave address. The MODE field selects the address width for this node when addressed as a Slave, not for the remote Slave. 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 sets 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. When one bit of address is shifted out by the SDA signal, the Transmit interrupt asserts.
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 the SDA signal.
11. The I^2C Slave sends an acknowledge (by pulling the SDA signal Low) during the next High period of SCL. The I^2C Controller sets the ACK bit in the I^2C State Register.

    If the slave does not acknowledge the address byte, the I^2C Controller sets the NCKI bit in the I^2C Interrupt Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. Software responds to the Not Acknowledge interrupt by setting the Stop bit and clearing the TXI bit. The I^2C Controller flushes the Transmit Data Register, 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 I2C Controller shifts the data out of through the SDA signal. When the first bit is sent, the Transmit interrupt asserts.
14. If more bytes remain to be sent, return to step 9.
15. When there is no more data to be sent, software responds by setting the Stop bit of the I2C Control Register (or Start bit to initiate a new transaction).
16. If no additional transaction is queued by the Master, software clears the TXI bit of the I2C Control Register.
17. The I2C Controller completes transmission of the data on the SDA signal.
18. The I2C Controller sends the Stop condition to the I2C bus.

Note: If the Slave terminates the transaction early by responding with a Not Acknowledge during the transfer, the I2C Controller asserts the NCKI interrupt and halts. Software must terminate the transaction by setting either the Stop bit (end transaction) or the Start bit (end this transaction, start a new one). In this case, it is not necessary for software to set the FLUSH bit of the I2CCTL Register to flush the data that was previously written but not transmitted. The I2C Controller hardware automatically flushes transmit data in this Not Acknowledge case.

Master Write Transaction with a 10-Bit Address

Figure 45 displays the data transfer format from a Master to a 10-bit addressed Slave.

S	Slave Address 1st Byte	W=0	A	Slave Address 2nd Byte	A	Data	A	Data	A/$\overline{A}$	F/S

Figure 45. Data Transfer Format, Master Write Transaction with 10-Bit Addressing

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 read/write control bit (=0). The transmit operation is carried out in the same manner as 7-Bit Address.

Observe the following procedure for a Master transmit operation to a 10-bit addressed Slave:

1. Software initializes the MODE field in the I2C Mode Register for Master/Slave Mode with 7- or 10-Bit Address (I2C bus protocol allows mixing Slave address types). The MODE field selects the address width for this node when addressed as a Slave, not for the remote Slave. 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 (11110xx0). 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. When one bit of address is shifted out by the SDA signal, the Transmit interrupt asserts.
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. The I2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL. The I2C Controller sets the ACK bit in the I2C Status Register.

    If the slave does not acknowledge the first address byte, the I2C Controller sets the NCKI bit in the I2C Status Register, sets the ACKV bit and clears the ACK bit in the I2C State Register. Software responds to the Not Acknowledge interrupt by setting the Stop bit and clearing the TXI bit. The I2C Controller flushes the second address byte from the data register, 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 (2nd address byte).
13. The I2C Controller shifts the second address byte out the SDA signal. When the first bit is sent, the Transmit interrupt asserts.
14. Software responds by writing the data to be written out to the I2C Control Register.
15. The I2C Controller shifts out the rest of the second byte of Slave address (or ensuing data bytes if looping) by the SDA signal.
16. The I2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL. The I2C Controller sets the ACK bit in the I2C Status Register.

    If the slave does not acknowledge, return to the second paragraph of Step 11.
17. The I2C Controller shifts the data out by the SDA signal. After the first bit is sent, the Transmit interrupt asserts.
18. If more bytes remain to be sent, return to step 14.

19. Software responds by asserting the Stop bit of the I2C Control Register.
20. The I2C Controller completes transmission of the data on the SDA signal.
21. The I2C Controller sends the Stop condition to the I2C bus.

Note: If the Slave responds with a Not Acknowledge during the transfer, the I2C Controller asserts the NCKI bit, sets the ACKV bit and clears the ACK bit in the I2C State Register and halts. Software terminates the transaction by setting either the Stop bit (end transaction) or the Start bit (end this transaction, start a new one). The Transmit Data Register is flushed automatically.

Master Read Transaction with a 7-Bit Address

Figure 46 displays the data transfer format for a read operation to a 7-bit addressed Slave.

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

Figure 46. Data Transfer Format, Master Read Transaction with 7-Bit Addressing

The procedure for a Master read operation to a 7-bit addressed Slave is as follows:

1. Software initializes the MODE field in the I2C Mode Register for Master/Slave Mode with 7- or 10-Bit Address (I2C bus protocol allows mixing Slave address types). The MODE field selects the address width for this node when addressed as a Slave, not for the remote Slave. Software asserts the IEN bit in the I2C Control Register.
2. Software writes the I2C Data Register with a 7-bit Slave address plus the read bit (=1).
3. Software asserts the Start bit of the I2C Control Register.
4. For 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 instruction is sent to the I2C Slave.
5. The I2C Controller sends the Start condition.
6. The I2C Controller sends the address and read bit out the SDA signal.
7. The I2C Slave acknowledges the address by pulling the SDA signal Low during the next High period of SCL.

   If the slave does not acknowledge the address byte, the I2C Controller sets the NCKI bit in the I2C Status Register, sets the ACKV bit and clears the ACK bit in the I2C State Register. Software responds to the Not Acknowledge interrupt by setting the

Stop bit and clearing the TXI bit. The I2C Controller flushes the Transmit Data Register, sends the Stop condition on the bus and clears the Stop and NCKI bits. The transaction is complete (ignore the following steps).

8. The I2C Controller shifts in the first byte of data from the I2C Slave on the SDA signal.
9. The I2C Controller asserts the Receive interrupt.
10. Software responds by reading the I2C Data Register. If the next data byte is to be the last, software must set the NAK bit of the I2C Control Register.
11. The I2C Controller sends a Not Acknowledge to the I2C Slave if it has received the last byte; otherwise it sends an Acknowledge.
12. If there are more bytes to transfer, the I2C Controller returns to step 7.
13. A NAK interrupt (NCKI bit in I2CISTAT) is generated by the I2C Controller.
14. Software responds by setting the Stop bit of the I2C Control Register.
15. A Stop condition is sent to the I2C Slave.

Master Read Transaction with a 10-Bit Address

Figure 47 displays the read transaction format for a 10-bit addressed Slave.

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

Figure 47. Data Transfer Format, Master Read Transaction with 10-Bit Addressing

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

The data transfer procedure for a read operation to a 10-bit addressed Slave is as follows:

1. Software initializes the MODE field in the I2C Mode Register for Master/Slave Mode with 7-bit or 10-Bit Address (I2C bus protocol allows mixing Slave address types). The MODE field selects the address width for this node when addressed as a Slave, not for the remote Slave. Software asserts the IEN bit in the I2C Control Register.
2. Software writes `11110b` followed by the two most significant address bits and a 0 (write) to the I2C Data Register.
3. Software asserts the Start bit of the I2C Control Register.
4. The I2C Controller sends the Start condition.
5. The I2C Controller loads the I2C Shift Register with the contents of the I2C Data Register.

6. When the first bit is shifted out, a Transmit interrupt asserts.
7. Software responds by writing the least significant eight bits of address to the I^2C Data Register.
8. The I^2C Controller completes shifting of the first address byte.
9. The I^2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL.

   If the slave does not acknowledge the address byte, the I^2C Controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. Software responds to the Not Acknowledge interrupt by setting the Stop bit and clearing the TXI bit. The I^2C Controller flushes the Transmit Data Register, sends the Stop condition on the bus and clears the Stop and NCKI bits. The transaction is complete (ignore the following steps).
10. The I^2C Controller loads the I^2C Shift Register with the contents of the I^2C Data Register (lower byte of 10 bit address).
11. The I^2C Controller shifts out the next eight bits of address. After the first bit shifts, the I^2C Controller generates a Transmit interrupt.
12. Software responds by setting the Start bit of the I^2C Control Register to generate a repeated Start.
13. Software responds by writing `11110b` followed by the 2-bit Slave address and a 1 (read) to the I^2C Data Register.
14. If you want to read only one byte, software responds by setting the NAK bit of the I^2C Control Register.
15. After the I^2C Controller shifts out the address bits mentioned in step 9 (second address transfer), the I^2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL.

    If the slave does not acknowledge the address byte, the I^2C Controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. Software responds to the Not Acknowledge interrupt by setting the Stop bit and clearing the TXI bit. The I^2C Controller flushes the Transmit Data Register, sends the Stop condition on the bus and clears the Stop and NCKI bits. The transaction is complete (ignore the following steps).
16. The I^2C Controller sends the repeated Start condition.
17. The I^2C Controller loads the I^2C Shift Register with the contents of the I^2C Data Register (third address transfer).
18. The I^2C Controller sends `11110b` followed by the two most significant bits of the Slave read address and a 1 (read).

19. The I2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL.
20. The I2C Controller shifts in a byte of data from the Slave.
21. The I2C Controller asserts the Receive interrupt.
22. Software responds by reading the I2C Data Register. If the next data byte is to be the last, software must set the NAK bit of the I2C Control Register.
23. The I2C Controller sends an Acknowledge or Not Acknowledge to the I2C Slave based on the NAK bit.
24. If there are more bytes to transfer, the I2C Controller returns to step 18.
25. The I2C Controller generates a NAK interrupt (NCKI bit in I2CISTAT).
26. Software responds by setting the Stop bit of the I2C Control Register.
27. A Stop condition is sent to the I2C Slave.

Slave Transactions

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

Slave Address Recognition

The following Slave address recognition options are supported:

Slave 7-Bit Address Recognition Mode. If IRM = 0 during the address phase and the controller is configured for Master/Slave or Slave 7-Bit Address Mode, the hardware detects a match to the 7-bit Slave address defined in the I2CSLVAD Register and generates the Slave Address Match interrupt (SAM bit = 1 in I2CISTAT Register). The I2C Controller automatically responds during the acknowledge phase with the value in the NAK bit of the I2CCTL Register.

Slave 10-Bit Address Recognition Mode. If IRM = 0 during the address phase and the controller is configured for Master/Slave or Slave 10-Bit Address Mode, the hardware detects a match to the 10-bit Slave address defined in the I2CMODE and I2CSLVAD registers and generates the Slave Address Match interrupt (SAM bit = 1 in I2CISTAT Register). The I2C Controller automatically responds during the acknowledge phase with the value in the NAK bit of the I2CCTL Register.

General Call and STARTBYTE Address Recognition Mode. If GCE = 1 and IRM = 0 during the address phase and the controller is configured for Master/Slave or Slave in either 7- or 10-Bit Address Mode, the hardware detects a match to the General Call Address or Start byte and generates the Slave Address Match interrupt. A General Call Address is a 7-bit address of all 0's with the R/$\overline{W}$ bit = 0. A Start byte is a 7-bit address of all 0's with the R/$\overline{W}$ bit = 1. The SAM and GCA bits are set in the I2CISTAT Register. The RD bit in the I2CISTAT Register distinguishes a General Call Address from a Start byte (=

0 for General Call Address). For a General Call Address, the I2C Controller automatically responds during the address acknowledge phase with the value in the NAK bit of the I2CCTL Register. If software processes the data bytes associated with the GCA bit, the IRM bit is optionally set following the SAM interrupt to allow software to examine each received data byte before deciding to set or clear the NAK bit. A Start byte will not be acknowledged (requirement the I2C specification).

Software Address Recognition Mode. To disable the hardware address recognition, the IRM bit must be set = 1 prior to the reception of the address byte(s). When IRM = 1 each received byte generates a receive interrupt (RDRF = 1 in the I2CISTAT Register). Software must examine each byte and determine whether to set or clear the NAK bit. The Slave holds SCL Low during the acknowledge phase until software responds by writing to the I2CCTL Register. The value written to the NAK bit is used by the controller to drive the I2C Bus, then releasing the SCL. The SAM and GCA bits are not set when IRM = 1 during the address phase, but the RD bit is updated based on the first address byte.

Slave Transaction Diagrams

In the following transaction diagrams, shaded regions indicate data transferred from the Master to the Slave and unshaded regions indicate data transferred from the Slave to the Master. The transaction field labels are defined as follows:

S: Start

W: Write

A: Acknowledge

$\overline{A}$: Not Acknowledge

P: Stop

Slave Receive Transaction with 7-Bit Address

The data transfer format for writing data from Master to Slave in 7-bit address Mode is shown in Figure 47. The following procedure describes the I2C Master/Slave Controller operating as a Slave in 7-bit address Mode, receiving data from the bus Master.

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

Figure 48. Data Transfer Format, Slave Receive Transaction with 7-Bit Addressing

1. Software configures the controller for operation as a Slave in 7-Bit Address Mode as follows.
   a. Initialize the MODE field in the I2C Mode Register for either Slave Only Mode or Master/Slave Mode with 7-Bit Addressing.

b. Optionally set the GCE bit.

c. Initialize the SLA[6:0] bits in the I²C Slave Address Register.

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

e. Program the Baud Rate High and Low Byte registers for the I²C baud rate.

2. The bus Master initiates a transfer, sending the address byte. The Slave Mode I²C Controller recognizes its own address and detects the R/$\overline{W}$ bit = 0 (write from Master to Slave). The I²C Controller acknowledges, indicating it is available to accept the transaction.The SAM bit in the I2CISTAT Register is set = 1, causing an interrupt. The RD bit in the I2CISTAT Register is set = 0, indicating a write to the Slave. The I²C Controller holds the SCL signal Low, waiting for software to load the first data byte.

3. Software responds to the interrupt by reading the I2CISTAT Register (which clears the SAM bit). After verifying that the SAM bit = 1, software checks the RD bit. When RD = 0, no immediate action is required until the first byte of data is received. If software is only able to accept a single byte it sets the NAK bit in the I2CCTL Register at this time.

4. The Master detects the acknowledge and sends the byte of data.

5. The I²C controller receives the data byte and responds with Acknowledge or Not Acknowledge depending on the state of the NAK bit in the I2CCTL Register. The I²C controller generates the receive data interrupt by setting the RDRF bit in the I2CISTAT Register.

6. Software responds by reading the I2CISTAT Register, finding the RDRF bit=1 and reading the I2CDATA Register clearing the RDRF bit. If software accepts only one more data byte, it sets the NAK bit in the I2CCTL Register.

7. The Master and Slave loop on steps 4–6 until the Master detects a Not Acknowledge instruction or runs out of data to send.

8. The Master sends the Stop or Restart signal on the bus. Either of these signals cause the I²C Controller to assert the Stop interrupt (Stop bit = 1 in the I2CISTAT Register). When the Slave receive data from the Master, software takes no action in response to the Stop interrupt other than reading the I2CISTAT Register, clearing the Stop bit in the I2CISTAT Register.

Slave Receive Transaction with 10-Bit Address

The data transfer format for writing data from Master to Slave with 10-Bit Addressing is shown in Figure 48. The following procedure describes the I²C Master/Slave Controller operating as a Slave in 10-Bit Addressing Mode, receiving data from the bus Master.

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

Figure 49. Data Transfer Format, Slave Receive Transaction with 10-Bit Addressing

1. Software configures the controller for operation as a Slave in 10-Bit Addressing Mode as follows.
   a. Initialize the MODE field in the I2CMODE Register for either Slave Only Mode or Master/Slave Mode with 10-Bit Addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[7:0] bits in the I2CSLVAD Register and the SLA[9:8] bits in the I2CMODE Register.
   d. Set IEN = 1 in the I2CCTL Register. Set NAK = 0 in the I^2C Control Register.
   e. Program the Baud Rate High and Low Byte registers for the I^2C baud rate.
2. The Master initiates a transfer by sending the first address byte. The I^2C Controller recognizes the start of a 10-bit address with a match to SLA[9:8] and detects the R/$\overline{W}$ bit = 0 (write from Master to Slave). The I^2C Controller acknowledges, indicating that it is available to accept the transaction.
3. The Master sends the second address byte. The Slave Mode I^2C Controller detects an address match between the second address byte and SLA[7:0]. The SAM bit in the I2CISTAT Register is set = 1, causing an interrupt. The RD bit is set = 0, indicating a write to the Slave. The I^2C Controller Acknowledges, indicating it is available to accept the data.
4. Software responds to the interrupt by reading the I2CISTAT Register, which clears the SAM bit. When RD = 0, no immediate action is taken by software until the first byte of data is received. If software is only able to accept a single byte it sets the NAK bit in the I2CCTL Register.
5. The Master detects the Acknowledge and sends the first byte of data.
6. The I^2C controller receives the first byte and responds with Acknowledge or Not Acknowledge, depending on the state of the NAK bit in the I2CCTL Register. The I^2C controller generates the receive data interrupt by setting the RDRF bit in the I2CISTAT Register.
7. Software responds by reading the I2CISTAT Register, finding the RDRF bit = 1 and then reading the I2CDATA Register, which clears the RDRF bit. If software accepts only one more data byte, it sets the NAK bit in the I2CCTL Register.
8. The Master and Slave loops on steps 5–7 until the Master detects a Not Acknowledge instruction or runs out of data to send.

9. The Master sends the Stop or Restart signal on the bus. Either of these signals cause the I2C Controller to assert the Stop interrupt (Stop bit = 1 in the I2CISTAT Register). When the Slave receive data from the Master, software takes no action in response to the Stop interrupt other than reading the I2CISTAT Register, clearing the Stop bit.

Slave Transmit Transaction with 7-Bit Address

The data transfer format for a Master reading data from a Slave in 7-bit address Mode is shown in Figure 49. The following procedure describes the I2C Master/Slave Controller operating as a Slave in 7-Bit Addressing Mode, transmitting data to the bus Master.

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

Figure 50. Data Transfer Format, Slave Transmit Transaction with 7-Bit Addressing

1. Software configures the controller for operation as a Slave in 7-Bit Addressing Mode as follows.
   a. Initialize the MODE field in the I2C Mode Register for either Slave Only Mode or Master/Slave Mode with 7-Bit Addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[6:0] bits in the I2C Slave Address Register.
   d. Set IEN = 1 in the I2C Control Register. Set NAK = 0 in the I2C Control Register.
   e. Program the Baud Rate High and Low Byte registers for the I2C baud rate.

2. The Master initiates a transfer, sending the address byte. The Slave Mode I2C Controller finds an address match and detects the R/$\overline{W}$ bit = 1 (read by Master from Slave). The I2C Controller acknowledges, indicating that it is ready to accept the transaction.The SAM bit in the I2CISTAT Register is set = 1, causing an interrupt. The RD bit is set = 1, indicating a read from the Slave.
3. Software responds to the interrupt by reading the I2CISTAT Register, clearing the SAM bit. When RD = 1, software responds by loading the first data byte into the I2CDATA Register. Software sets the TXI bit in the I2CCTL Register to enable transmit interrupts. When the Master initiates the data transfer, the I2C Controller holds SCL Low until software has written the first data byte to the I2CDATA Register.
4. SCL is released and the first data byte is shifted out.
5. When the first bit of the first data byte is transferred, the I2C controller sets the TDRE bit, which asserts the transmit data interrupt.
6. Software responds to the transmit data interrupt (TDRE = 1) by loading the next data byte into the I2CDATA Register, which clears TDRE.

7. When the Master receives the data byte, the Master transmits an Acknowledge instruction (or Not Acknowledge instruction for the last data byte).
8. The bus cycles through steps 5–7 until the last byte has been transferred. If software has not yet loaded the next data byte when the Master brings SCL Low to transfer the most significant data bit, the Slave I^2C Controller holds SCL Low until the data register is written. When the Slave receives a Not Acknowledge instruction, the I^2C Controller sets the NCKI bit in the I2CISTAT Register and generates the Not Acknowledge interrupt.
9. Software responds to the Not Acknowledge interrupt by clearing the TXI bit in the I2CCTL Register and by asserting the FLUSH bit of the I2CCTL Register to empty the data register.
10. When the Master completes the last acknowledge cycle, it asserts the Stop or Restart condition on the bus.
11. The Slave I^2C Controller asserts the Stop/Restart interrupt (set the SPRS bit in I2CISTAT Register).
12. Software responds to the Stop/Restart interrupt by reading the I2CISTAT Register which clears the SPRS bit.

Slave Transmit (Master Read) Transaction with 10-Bit Address

Figure 51 displays the data transfer format for a Master reading data from a Slave with 10-Bit Addressing.

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

Figure 51. Data Transfer Format, Slave Transmit Transaction with 10-Bit Addressing

The following procedure describes the I^2C Master/Slave Controller operating as a Slave in 10-Bit Addressing Mode, transmitting data to the bus Master:

1. Software configures the controller for operation as a Slave in 10-Bit Addressing Mode.
   - Initialize the MODE field in the I^2C Mode Register for either Slave Only Mode or Master/Slave Mode with 10-Bit Addressing.
   - Optionally set the GCE bit.
   - Initialize the SLA[7:0] bits in the I2CSLVAD Register and SLA[9:8] in the I2CMODE Register.
   - Set IEN = 1, NAK = 0 in the I^2C Control Register.
   - Program the Baud Rate High and Low Byte registers for the I^2C baud rate.

2. The Master initiates a transfer, sending the first address byte. The Slave Mode I^2C Controller recognizes the start of a 10-bit address with a match to SLA[9:8] and detects the R/$\overline{W}$ bit = 0 (write from Master to Slave). The I^2C Controller acknowledges, indicating that it is available to accept the transaction.
3. The Master sends the second address byte. The Slave Mode I^2C Controller compares the second address byte with the value in SLA[7:0]. If there is a match, the SAM bit in the I2CISTAT Register is set = 1, causing a Slave Address Match interrupt. The RD bit is set = 0, indicating a write to the Slave. If a match occurs, the I^2C Controller acknowledges on the I^2C bus, indicating that it is available to accept the data.
4. Software responds to the Slave Address Match interrupt by reading the I2CISTAT Register which clears the SAM bit. When the RD bit = 0, no further action is required.
5. The Master notifies the Acknowledge and sends a Restart instruction, followed by the first address byte with the R/$\overline{W}$ = 1. The Slave Mode I^2C Controller recognizes the Restart followed by the first address byte with a match to SLA[9:8] and detects the R/$\overline{W}$ = 1 (Master reads from Slave). The Slave I^2C Controller sets the SAM bit in the I2CISTAT Register, which causes the Slave Address Match interrupt. The RD bit is set = 1. The Slave Mode I^2C Controller acknowledges on the bus.
6. Software responds to the interrupt by reading the I2CISTAT Register, clearing the SAM bit. Software loads the initial data byte into the I2CDATA Register and sets the TXI bit in the I2CCTL Register.
7. The Master starts the data transfer by asserting SCL Low. After the I^2C Controller has data available to transmit the SCL is released and the Master proceeds to shift the first data byte.
8. When the first bit of the first data byte is transferred, the I^2C controller sets the TDRE bit, which asserts the transmit data interrupt.
9. Software responds to the transmit data interrupt by loading the next data byte into the I2CDATA Register.
10. The I^2C Master shifts in the remainder of the data byte. The Master transmits the Acknowledge (or Not Acknowledge for the last data byte).
11. The bus cycles through steps 7–10 until the last byte has been transferred. If software has not yet loaded the next data byte when the Master brings SCL Low to transfer the most significant data bit, the Slave I^2C Controller holds SCL Low until the data register is written. When the Slave receives a Not Acknowledge, the I^2C Controller sets the NCKI bit in the I2CISTAT Register and generates the NAK interrupt.
12. Software responds to the NAK interrupt by clearing the TXI bit in the I2CCTL Register and by asserting the FLUSH bit of the I2CCTL Register.
13. When the Master has completed the acknowledge cycle of the last transfer it asserts the Stop or Restart condition on the bus.

14. The Slave I2C Controller asserts the Stop/Restart interrupt (set the SPRS bit in the I2CISTAT Register).
15. Software responds to the Stop interrupt by reading the I2CISTAT Register, clearing the SPRS bit.

DMA Control of I2C Transactions

The DMA engine is configured to support transmit and receive DMA requests from the I2C Controller. The I2C data interrupt requests must be disabled by setting the DMAIF bit in the I2C Mode Register and clearing the TXI bit in the I2C Control Register. This allows error condition interrupts to be handled by software while data movement is handled by the DMA engine.

The DMA interface on the I2C Controller is intended to support data transfer but not Master Mode address byte transfer. The Start, Stop and NAK bits must be controlled by software.

A summary of the sequence of I2C data transfer using the DMA follows.

Master Write Transaction with Data DMA

1. Configure the selected DMA Channel for I2C transmit. The IEOB bit must be set in the DMACTL Register for the last buffer to be transferred.
2. The I2C interrupt must be enabled in the interrupt controller to alert software of any I2C error conditions. A Not Acknowledge interrupt occurs on the last byte transferred.
3. The I2C Master/Slave must be configured as defined in the sections above describing Master Mode transactions. The TXI bit in the I2CCTL Register must be cleared.
4. Initiate the I2C transaction as described in the Master Address Only Transactions section on page 210, using the ACKV and ACK bits in the I2CSTATE Register to determine if the slave acknowledges.
5. Set the DMAIF bit in the I2CMODE Register.
6. The DMA transfers the data, which is to be transmitted to the slave.
7. When the DMA interrupt occurs, poll the I2CSTAT Register until the TDRE bit = 1 to ensure that the I2C Master/Slave hardware has commenced transmitting the most recent byte written by the DMA.
8. Set the Stop bit in the I2CCTL Register. The Stop bit is polled by software to determine when the transaction is actually completed.
9. Clear the DMAIF bit in the I2CMODE Register.

The following section describes the I2C Master/Slave Controller operating as a Slave in 10-Bit Addressing Mode, transmitting data to the bus master.

Note: If the slave sends a Not Acknowledge prior to the last byte, a Not Acknowledge interrupt occurs. Software must respond to this interrupt by clearing the DMAIF bit and setting the Stop bit to end the transaction.

Master Read Transaction with Data DMA

In master read transactions, the Master is responsible for the Acknowledge for each data byte transferred. The Master software must set the NAK bit after the next to the last data byte has been received or while the last byte is being received. The DMA supports this by setting the DMA watermark to 0x01, which results in a DMA interrupt when the next to the last byte has been received. A DMA interrupt also occurs when the last byte is received. Otherwise, the sequence is similar to that described above for the Master write transaction.

1. Configure the selected DMA Channel for I^2C receive. The IEOB bit must be set in the DMACTL Register for the last buffer to be transferred. Typically one buffer is defined with a transfer length of N where N bytes are expected to be read from the slave. The watermark is set to 1 by writing a 0x01 to DMAxLAR[23:16].
2. The I^2C interrupt must be enabled in the interrupt controller to alert software of any I^2C error conditions. A Not Acknowledge interrupt occurs on the last byte transferred.
3. The I^2C Master/Slave must be configured as defined in the sections above describing Master Mode transactions. The TXI bit in the I2CCTL Register must be cleared.
4. Initiate the I^2C transaction as described in the Master Address Only Transactions section on page 210, using the ACKV and ACK bits in the I2CSTATE Register to determine if the slave acknowledges. Do not set the Stop bit unless ACKV=1 and ACK=0 (slave did not acknowledge).
5. Set the DMAIF bit in the I2CMODE Register.
6. The DMA transfers the data to memory as it is received from the slave.
7. When the first DMA interrupt occurs indicating the (N-1)st byte has been received, the NAK bit must be set in the I2CCTL Register.
8. When the second DMA interrupt occurs, it indicates that the Nth byte has been received. Set the Stop bit in the I2CCTL Register. The Stop bit is polled by software to determine when the transaction is actually completed.
9. Clear the DMAIF bit in the I2CMODE Register.

Slave Write Transaction with Data DMA

In a transaction where the I^2C Master/Slave operates as a slave, receiving data written by a master, the software must set the NAK bit after the N-1st byte has been received or during

the reception of the last byte. As in the Master Read transaction described above, the watermark DMA interrupt is used to notify software when the N-1st byte has been received.

1. Configure the selected DMA Channel for I^2C receive. The IEOB bit must be set in the DMACTL Register for the last buffer to be transferred. Typically one buffer will be defined with a transfer length of N where N bytes are expected to be received from the master. The watermark is set to 1 by writing a 0x01 to DMAxLAR[23:16].
2. The I^2C interrupt must be enabled in the interrupt controller to alert software of any I^2C error conditions.
3. The I^2C Master/Slave must be configured as defined in the sections above describing Slave Mode transactions. The TXI bit in the I2CCTL Register must be cleared.
4. When the SAM interrupt occurs, set the DMAIF bit in the I2CMODE Register.
5. The DMA transfers the data to memory as it is received from the master.
6. When the first DMA interrupt occurs indicating that the (N-1)st byte is received, the NAK bit must be set in the I2CCTL Register.
7. When the second DMA interrupt occurs, it indicates that the Nth byte is received. A Stop I^2C interrupt occurs (SPRS bit set in the I2CSTAT Register) when the master issues the Stop (or Restart) condition.
8. Clear the DMAIF bit in the I2CMODE Register.

Slave Read Transaction with Data DMA

In this transaction the I^2C Master/Slave operates as a slave, sending data to the master.

1. Configure the selected DMA Channel for I^2C transmit. The IEOB bit must be set in the DMACTL Register for the last buffer to be transferred. Typically a single buffer with a transfer length of N is defined.
2. The I^2C interrupt must be enabled in the interrupt controller to alert software of any I^2C error conditions. A Not Acknowledge interrupt occurs on the last byte transferred.
3. The I^2C Master/Slave must be configured as defined in the sections above describing Slave Mode transactions. The TXI bit in the I2CCTL Register must be cleared.
4. When the SAM interrupt occurs, set the DMAIF bit in the I2CMODE Register.
5. The DMA transfers the data to be transmitted to the master.
6. When the DMA interrupt occurs, the last byte is being transferred to the master. The master must send a Not Acknowledge for this last byte, setting the NCKI bit in the I2CSTAT Register and generating the I^2C interrupt. A Stop or Restart interrupt (SPRS bit set in I2CSTAT Register) follows.
7. Clear the DMAIF bit in the I2CMODE Register.

Note: If the master sends a Not Acknowledge prior to the last byte, software responds to the Not Acknowledge interrupt by clearing the DMAIF bit.

I²C Control Register Definitions

The following section describes the I²C control registers.

I²C Data Register

The I²C Data Register, shown in Table 112, holds the data that is to be loaded into the Shift Register to transmit onto the I²C bus. This register also holds data that is loaded from the Shift Register after it is received from the I²C bus. The I²C Shift Register is not accessible in the Register File address space, but is used only to buffer incoming and outgoing data.

Writes by software to the I2CDATA Register are blocked if a slave write transaction is underway (I²C Controller in Slave Mode, data being received).

Table 112. I²C Data Register (I2CDATA)

Bits	7	6	5	4	3	2	1	0
Field	DATA							
RESET	0							
R/W	R/W							
Addr	FF–E240h							

I²C Interrupt Status Register

The Read-only I²C Interrupt Status Register, shown in Table 113, indicates the cause of any current I²C interrupt and provides the status of the I²C Controller. When an interrupt occurs, one or more of the TDRE, RDRF, SAM, ARBLST, SPRS or NCKI bits is set. The GCA and RD bits do not generate an interrupt but rather provide status associated with the SAM bit interrupt.

Table 113. I²C Interrupt Status Register (I2CISTAT)

Bits	7	6	5	4	3	2	1	0
Field	TDRE	RDRF	SAM	GCA	RD	ARBLST	SPRS	NCKI
RESET	1	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Addr	FF–E241h							

Bit	Description
[7] TDRE	**Transmit Data Register Empty** When the I²C Controller is enabled, this bit is 1 if the I²C Data Register is empty. When set, the I²C Controller generates an interrupt, except when the I²C Controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit clears by writing to the I2CDATA Register.
[6] RDRF	**Receive Data Register Full** This bit is set = 1 when the I²C Controller is enabled and the I²C Controller has received a byte of data. When asserted, this bit causes the I²C Controller to generate an interrupt. This bit clears by reading the I2CDATA Register.
[5] SAM	**Slave Address Match** This bit is set = 1 if the I²C Controller is enabled in Slave Mode and an address is received which matches the unique Slave address or General Call Address (if enabled by the GCE bit in the I²C Mode Register). In 10-Bit Addressing Mode, this bit is not set until a match is achieved on both address bytes. When this bit is set, the RD and GCA bits are also valid. This bit clears by reading the I2CISTAT Register.
[4] GCA	**General Call Address** This bit is set in Slave Mode when the General Call Address or Start byte is recognized (in either 7- or 10-bit Slave Mode). The GCE bit in the I²C Mode Register must be set to enable recognition of the General Call Address and Start byte. This bit clears when IEN = 0 and is updated following the first address byte of each Slave Mode transaction. A General Call Address is distinguished from a Start byte by the value of the RD bit (RD = 0 for General Call Address, 1 for Start byte).
[3] RD	**Read** This bit indicates the direction of transfer of the data. It is set when the Master is reading data from the Slave. This bit matches the least-significant bit of the address byte after the Start condition occurs (for both Master and Slave modes). This bit clears when IEN = 0 and is updated following the first address byte of each transaction.
[2] ARBLST	**Arbitration Lost** This bit is set when the I²C Controller is enabled in Master Mode and loses arbitration (outputs a 1 on SDA and receives a 0 on SDA). The ARBLST bit clears when the I2CISTAT Register is read.

Bit	Description (Continued)
[1] SPRS	**Stop/Restart Condition Interrupt** This bit is set when the I2C Controller is enabled in Slave Mode and detects a Stop or Restart condition during a transaction directed to this slave. This bit clears when the I2CISTAT Register is read. Read the RSTR bit of the I2CSTATE Register to determine whether the interrupt was caused by a Stop or Restart condition.
[0] NCKI	**NAK Interrupt** In Master Mode, this bit is set when a Not Acknowledge condition is received or sent and neither the Start nor the Stop bit is active. In Master Mode, this bit is cleared only by setting the Start or Stop bits. In Slave Mode, this bit is set when a Not Acknowledge condition is received (Master reading data from Slave), indicating the Master is finished reading. A Stop or Restart condition follows. In Slave Mode this bit clears when the I2CISTAT Register is read.

I2C Control Register

The I2C Control Register, shown in Table 114, enables and configures the I2C operation.

Table 114. I2C Control Register (I2CCTL)

Bits	7	6	5	4	3	2	1	0
Field	IEN	START	STOP	BIRQ	TXI	NAK	FLUSH	FILTEN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W1	R/W1	R/W	R/W	R/W1	R/W	R/W
Addr	FF–E242h							

Note: R/W1 = The bit is set (write 1) but not cleared.

Bit	Description
[7] IEN	**I2C Enable** This bit enables the I2C Controller.
[6] START	**Send Start Condition** When set, this bit causes the I2C Controller (when configured as the Master) to send the Start condition. After it is asserted, this bit is cleared by the I2C Controller after it sends the Start condition or by deasserting the IEN bit. If this bit is 1, it cannot be cleared by writing to the bit. After this bit is set, the Start condition is sent if there is data in the I2CDATA or I2CSHIFT Register. If there is no data in one of these registers, the I2C Controller waits until data is loaded. If this bit is set while the I2C Controller is shifting out data, it generates a Restart condition after the byte shifts and the acknowledge phase completes. If the Stop bit is also set, it also waits until the Stop condition is sent before the Start condition. If Start is set while a Slave Mode transaction is underway to this device, the Start bit is cleared and ARBLST bit in the Interrupt Status Register will be set.

Bit	Description (Continued)
[5] STOP	**Send Stop Condition** When set, this bit causes the I²C Controller (when configured as the Master) to send the Stop condition after the byte in the I²C Shift Register has completed transmission or after a byte has been received in a receive operation. When set, this bit is reset by the I²C Controller after a Stop condition has been sent or by deasserting the IEN bit. If this bit is 1, it cannot be cleared to 0 by writing to the register. If Stop is set while a Slave Mode transaction is underway, the Stop bit will be cleared by hardware.
[4] BIRQ	**Baud Rate Generator Interrupt Request** This bit is ignored when the I²C Controller is enabled. If this bit is set = 1 when the I²C Controller is disabled (IEN = 0) the baud rate generator is used as an additional timer causing an interrupt to occur every time the baud rate generator counts down to 1. The baud rate generator runs continuously in this Mode, generating periodic interrupts.
[3] TXI	**Enable TDRE Interrupts** This bit enables interrupts when the I²C Data Register is empty.
[2] NAK	**Send NAK** Setting this bit sends a Not Acknowledge condition after the next byte of data has been received. It is automatically deasserted after the Not Acknowledge is sent or the IEN bit is cleared. If this bit is 1, it cannot be cleared to 0 by writing to the register.
[1] FLUSH	**Flush Data** Setting this bit clears the I²C Data Register and sets the TDRE bit to 1. This bit allows flushing of the I²C Data Register when an NAK condition is received after the next data byte has been written to the I²C Data Register. Reading this bit always returns 0.
[0] FILTEN	**I²C Signal Filter Enable** Setting this bit enables low-pass digital filters on the SDA and SCL input signals. This function provides the spike suppression filter required in I²C Fast Mode. These filters reject any input pulse with periods less than a full system clock cycle. The filters introduce a 3-system clock cycle latency on the inputs.

I²C Baud Rate High and Low Byte Registers

The I²C Baud Rate High and Low Byte registers, shown in Tables 115 and 116, combine to form a 16-bit reload value, BRG[15:0], for the I²C Baud Rate Generator. The baud rate High and Low Byte Registers must be programmed for the I²C baud rate in Slave Mode as well as in Master Mode. In Slave Mode, the baud rate value programmed must match the master's baud rate within ±25% for proper operation.

The I²C baud rate is calculated using the following equation.

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

Note: If BRG = 0000h, use 10000h in the equation.

Table 115. I²C Baud Rate High Byte Register (I2CBRH)

Bits	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFh							
R/W	R/W							
Addr	FF–E243h							

Bit	Description
[7:0] BRH	**I²C Baud Rate High Byte** Most significant byte, BRG[15:8], of the I²C Baud Rate Generator's reload value.

Note: If the DIAG bit in the I²C Mode Register is set to 1, a read of the I2CBRH Register returns the current value of the I²C Baud Rate Counter[15:8].

Table 116. I²C Baud Rate Low Byte Register (I2CBRL)

Bits	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFh							
R/W	R/W							
Addr	FF–E244h							

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

Note: If the DIAG bit in the I²C Mode Register is set to 1, aread of the I2CBRL Register returns the current value of the I²C Baud Rate Counter[7:0].

I²C State Register

The read-only I²C State Register provides information about the state of the I²C bus and the I²C Bus Controller.

When the DIAG bit of the I²C Mode Register is cleared, this register provides information about the internal state of the I²C Controller and I²C Bus as shown in Table 117. When the DIAG bit of the I²C Mode Register is set, this register returns the value of the I²C Controller state machine, as shown in Table 118. Tables 119 and 120 describe the state encoding of the I²C State registers.

Table 117. I²C State Register (I2CSTATE), Description when DIAG = 0

Bits	7	6	5	4	3	2	1	0
Field	ACKV	ACK	AS	DS	10B	RSTR	SCLOUT	BUSY
RESET	0	0	0	0	0	0	X	X
R/W	R	R	R	R	R	R	R	R
Addr	FF–E245h							

Bit	Description
[7] ACKV	**ACK Valid** This bit is set if sending data (Master or Slave) and the ACK bit in this register is valid for the byte just transmitted. This bit is monitored if it is appropriate for software to verify the ACK value before writing the next byte to be sent. To operate in this Mode, the data register must not be written when TDRE asserts; instead, software waits for ACKV to assert. This bit clears when transmission of the next byte begins or the transaction is ended by a Stop or Restart condition.
[6] ACK	**Acknowledge** This bit indicates the status of the Acknowledge for the last byte transmitted or received. This bit is set for an Acknowledge and cleared for a Not Acknowledge condition.
[5] AS	**Address State** This bit is active High while the address is being transferred on the I²C bus.
[4] DS	**Data State** This bit is active High while the data is being transferred on the I²C bus.
[3] 10B	**10B** This bit indicates whether a 10 or 7-bit address is being transmitted when operating as a Master. After the Start bit is set, if the five most-significant bits of the address are 11110b, this bit is set. When set, it is reset after the address has been sent.
[2] RSTR	**Restart** This bit is updated each time a Stop or Restart interrupt occurs (SPRS bit set in I2CISTAT Register). 0 = Stop condition 1 = Restart condition

Bit	Description (Continued)
[1] SCLOUT	**Serial Clock Output** Current value of Serial Clock being output onto the bus. The actual values of the SCL and SDA signals on the I^{2}C bus is observed via the GPIO Input Register.
[0] BUSY	**I^{2}C Bus Busy** 0 = No activity on the I^{2}C Bus. 1 = A transaction is underway on the I^{2}C bus.

Table 118. I^{2}C State Register (I2CSTATE), Description when DIAG = 1

Bits	7	6	5	4	3	2	1	0
Field	I2CSTATE_H				I2CSTATE_L			
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Addr	FF–E245h							

Bit	Description
[7:4] I2CSTATE_H	**I^{2}C State High** This field defines the current state of the I^{2}C Controller. It is the most significant nibble of the internal state machine. Table 119 defines the states for this field.
[3:0] I2CSTATE_L	**I^{2}C State Low** Least significant nibble of the I^{2}C state machine. This field defines the substates for the states defined by I2CSTATE_H. Table 120 defines the values for this field.

Table 119. I2CSTATE_H

State Encoding	State Name	State Description
0000	Idle	I^{2}C bus is idle or I^{2}C Controller is disabled.
0001	Slave Start	I^{2}C Controller has received a start condition.
0010	Slave Bystander	Address did not match–ignore remainder of transaction.
0011	Slave Wait	Waiting for Stop or Restart condition after sending a Not Acknowledge instruction.
0100	Master Stop2	Master completing Stop condition (SCL = 1, SDA = 1).
0101	Master Start/Restart	Master Mode sending Start condition (SCL = 1, SDA = 0).
0110	Master Stop1	Master initiating Stop condition (SCL = 1, SDA = 0).

Table 119. I2CSTATE_H (Continued)

State Encoding	State Name	State Description
0111	Master Wait	Master received a Not Acknowledge instruction, waiting for software to assert Stop or Start control bits.
1000	Slave Transmit Data	Nine substates, one for each data bit and one for the acknowledge.
1001	Slave Receive Data	Nine substates, one for each data bit and one for the acknowledge.
1010	Slave Receive Addr1	Slave Receiving first address byte (7 and 10 bit addressing) Nine substates, one for each address bit and one for the acknowledge.
1011	Slave Receive Addr2	Slave Receiving second address byte (10 bit addressing) Nine substates, one for each address bit and one for the acknowledge.
1100	Master Transmit Data	Nine substates, one for each data bit and one for the acknowledge.
1101	Master Receive Data	Nine substates, one for each data bit and one for the acknowledge.
1110	Master Transmit Addr1	Master sending first address byte (7- and 10-Bit Addressing) Nine substates, one for each address bit and one for the acknowledge.
1111	Master Transmit Addr2	Master sending second address byte (10-Bit Addressing) Nine substates, one for each address bit and one for the acknowledge.

Table 120. I2CSTATE_L

State I2CSTATE_H	Sub-State I2CSTATE_L	Sub-State Name	State Description
0000–0100	0000	—	There are no substates for these I2CSTATE_H values.
0110–0111	0000	—	There are no substates for these I2CSTATE_H values.

Table 120. I2CSTATE_L (Continued)

State I2CSTATE_H	Sub-State I2CSTATE_L	Sub-State Name	State Description
0101	0000	Master Start	Initiating a new transaction.
	0001	Master Restart	Master is ending one transaction and starting a new one without letting the bus go nonactive.
1000–1111	0111	send/receive bit 7	Sending/Receiving most significant bit.
	0110	send/receive bit 6	
	0101	send/receive bit 5	
	0100	send/receive bit 4	
	0011	send/receive bit 3	
	0010	send/receive bit 2	
	0001	send/receive bit 1	
	0000	send/receive bit 0	Sending/Receiving least significant bit
	1000	send/receive Acknowl-edge	Sending/Receiving Acknowledge

I²C Mode Register

The I²C Mode Register, shown in Table 121, provides control over Master versus Slave operating mode, Slave address and diagnostic modes.

Table 121. I²C Mode Register (I2CMODE)

Bits	7	6	5	4	3	2	1	0
Field	DMAIF	MODE[1:0]		IRM	GCE	SLA[9:8]		DIAG
RESET	0	0		0	0	0		0
R/W	R/W	R/W		R/W	R/W	R/W		R/W
Addr	FF–E246h							

Bit	Description
[7] DMAIF	**DMA Interface Mode** 0 = Used when software polling or interrupts are used to move data. 1 = Used when the DMA is used to move data. The TDRE and RDRF bits in the status register are not affected but the I²C Interrupt is not asserted when TDRE or RDRF are set. The I²C interrupt reflects only the error conditions. The assertion of TDRE causes a transmit DMA request. The assertion of RDRF causes a receive DMA request.
[6:5] MODE[1:0]	**I²C Controller Operational Mode Select** 00 = Master/Slave capable (supports multi-Master arbitration) with 7-bit Slave address 01 = Master/Slave capable (supports multi-Master arbitration) with 10-bit Slave address 10 = Slave Only capable with 7-bit address 11 = Slave Only capable with 10-bit address
[4] IRM	**Interactive Receive Mode** Valid in Slave Mode when software must interpret each received byte before acknowledging. This bit is useful for processing the data bytes following a General Call Address or if software wants to disable hardware address recognition. 0 = Acknowledge occurs automatically and is determined by the value of the NAK bit of the I2CCTL Register. 1 = A receive interrupt is generated for each byte received (address or data). The SCL is held Low during the acknowledge cycle until software writes to the I2CCTL Register. The value written to the NAK bit of the I2CCTL Register is output on SDA. This value allows software to Acknowledge or Not Acknowledge after interpreting the associated address/data byte.
[3] GCE	**General Call Address Enable** Enables reception of messages beginning with the General Call Address or Start byte. 0 = Do not accept a message with the General Call Address or Start byte. 1 = Do accept a message with the General Call Address or Start byte. When an address match occurs, the GCA and RD bits in the I²C Status Register indicates whether the address matched the General Call Address/Start byte or not. Following the General Call Address byte, software sets the IRM bit that allows software to examine the following data byte(s) before acknowledging.

Bit	Description (Continued)
[2:1] SLA[9:8]	**Slave Address Bits 9 and 8** Initialize with the appropriate Slave address value when using 10-bit Slave addressing. These bits are ignored when using 7-bit Slave addressing.
[0] DIAG	**Diagnostic Mode** Selects read back value of the Baud Rate Reload and State registers. 0 = Reading the Baud Rate registers returns the Baud Rate register values. Reading the State Register returns I2C Controller state information. 1 = Reading the Baud Rate registers returns the current value of the baud rate counter. Reading the State Register returns additional state information.

I2C Slave Address Register

The I2C Slave Address Register, shown in Table 122, provides control over the lower order address bits used in 7-bit and 10-bit Slave address recognition.

Table 122. I2C Slave Address Register (I2CSLVAD)

Bits	7	6	5	4	3	2	1	0
Field	SLA[7:0]							
RESET	00h							
R/W	R/W							
Addr	FF–E247h							

Bit	Description
[7:0] SLA[7:0]	**Slave Address Bits 7–0** Initialize with the appropriate Slave address value. When using 7 bit Slave addressing, SLA[9:7] are ignored.

Watchdog Timer

The Watchdog Timer (WDT) helps protect against corrupt or unreliable software, power faults and other system-level problems which places the ZNEO® Z16F Series device into unsuitable operating states.

The WDT includes the following features:

- On-chip RC oscillator
- A selectable time-out response: short reset or system exception
- 16-bit programmable time-out value

Operation

The WDT is a retriggerable one-shot timer that resets or interrupts the ZNEO Z16F Series device, when the WDT reaches its terminal count. The WDT uses its own dedicated on-chip RC oscillator as its clock source. The WDT has only two modes of operation—on and off. After enabled, it always counts and must be refreshed to prevent a time-out. An enable is performed by executing the WDT instruction or by setting the WDT_AO option bit. The WDT_AO bit enables the WDT to operate all of the time, even if a WDT instruction has not been executed.

To minimize power consumption, the RC oscillator is disabled. The RC oscillator is disabled by clearing the WDTEN bit in the Oscillator Control Register. If the RC oscillator is disabled, the WDT will not operate.

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

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

In the equation above, the WDT reload value is the decimal value of the 16-bit value yielded by {WDTH[7:0], WDTL[7:0]} and the typical Watchdog Timer RC oscillator frequency is 10kHz. Table 123 provides approximate time-out delays for the minimum, default and maximum WDT reload values.

Table 123. Watchdog Timer Approximate Time-Out Delays

WDT Reload Value	WDT Reload Value	Approximate Time-Out Delay (with 10 kHz typical WDT oscillator frequency)	
(Hex)	(Decimal)	Typical	Description
0400	1024	102.4 ms	Reset value time-out delay
FFFF	65,536	6.55 s	Maximum time-out delay

Watchdog Timer Refresh

When enabled first, the WDT is loaded with the value in the Watchdog Timer Reload registers. The WDT then counts down to 0000h unless a WDT instruction is executed by the ZNEO 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 ZNEO Z16F Series device is operating in DEBUG Mode (through the OCD), the WDT is continuously refreshed to prevent spurious WDT time-outs.

Watchdog Timer Time-Out Response

The WDT times out when the counter reaches 0000h. A time-out of the WDT generates either a system exception or a short reset. The WDT_RES option bit determines the time-out response of the WDT. For information about programming of the WDT_RES option bit, see the Option Bits chapter on page 292.

WDT System Exception in Normal Operation

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

WDT System Exception in Stop Mode

If configured to generate a system exception when a time-out occurs and the ZNEO Z16F Series device is in Stop Mode, the WDT automatically initiates a Stop Mode Recovery and generates a system exception request. Both the WDT status bit and the Stop bit in the Reset Status and Control Register are set to 1 following WDT time-out in Stop Mode. For detailed information, see the Reset and Stop Mode Recovery chapter on page 56.

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

WDT Reset in Normal Operation

If configured to generate a Reset when a time-out occurs, the WDT forces the device into the Reset state. The WDT status bit in the Reset Status and Control Register is set to 1. For more information about Reset and the WDT status bit, see the Reset and Stop Mode Recovery chapter on page 56. Following a Reset sequence, the WDT Counter is initialized with its reset value.

WDT Reset in Stop Mode

If enabled in Stop Mode and configured to generate a Reset when a time-out occurs and the device is in Stop Mode, the WDT initiates a Stop Mode Recovery. Both the WDT status bit and the Stop bit in the Reset Status and Control Register register are set to 1 following WDT time-out in Stop Mode. For detailed information, see the Reset and Stop Mode Recovery chapter on page 56.

Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watchdog Timer Reload High (WDTH) register address unlocks the two Watchdog Timer Reload registers (WDTH and WDTL) to allow changes to the time-out period. These write operations to the WDTH Register address produce no effect on the bits in the WDTH Register. The locking mechanism prevents spurious writes to the reload registers.

The following sequence is required to unlock the Watchdog Timer Reload registers (WDTH and WDTL) for write access:

1. Write 55h to the Watchdog Timer Reload High register (WDTH).
2. Write AAh to the Watchdog Timer reload high register (WDTH).
3. Write the appropriate value to the Watchdog Timer reload high register (WDTH).
4. Write the appropriate value to the Watchdog Timer reload low register (WDTL).

All steps of the WDT reload unlock sequence must be written in the order presented above. The values in the Watchdog Timer Reload registers are loaded into the counter every time a WDT instruction is executed.

Watchdog Timer Register Definitions

Watchdog Timer Reload High and Low Byte Registers

The Watchdog Timer Reload High and Low Byte (WDTH, WDTL) registers, shown in Table 124 through Table 125) form the 16-bit reload value that is loaded into the WDT when a WDT instruction executes. The 16-bit reload value is {WDTH[7:0], WDTL[7:0]}. Writing to these registers following the unlock sequence sets the appropriate reload value. Reading from these registers returns the current WDT count value.

Caution: The 16-bit WDT Reload Value must not be set to a value less than 0004h.

Table 124. Watchdog Timer Reload High Byte Register (WDTH)

Bits	7	6	5	4	3	2	1	0
Field	WDTH							
RESET	0	0	0	0	0	1	0	0
R/W	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*
Addr	FF_E042h							

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** Most significant byte (MSB), Bits[15:8], of the 16-bit WDT reload value.

Table 125. Watchdog Timer Reload Low Byte Register (WDTL)

Bits	7	6	5	4	3	2	1	0
Field	WDTL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*
Addr	FF_E043h							

Note: R/W* = Read returns the current WDT count value. Write sets the appropriate Reload Value.

Bit	Description
[7:0] WDTL	**WDT Reload Low Byte** Least significant byte (LSB), Bits[7:0], of the 16-bit WDT reload value.

Analog Functions

The ZNEO® Z16F Series devices include a 12-channel Analog-to-Digital Converter (ADC), an operational amplifier and a comparator.

The features of the analog functions include:

- ADC with 12 analog input sources multiplexed with General-Purpose Input/Output (GPIO) ports
- Operational amplifier with output internally connect to the ADC
- Comparator with separate inputs or shared with the operational amplifier

Figure 52 displays the block diagram for analog functions.

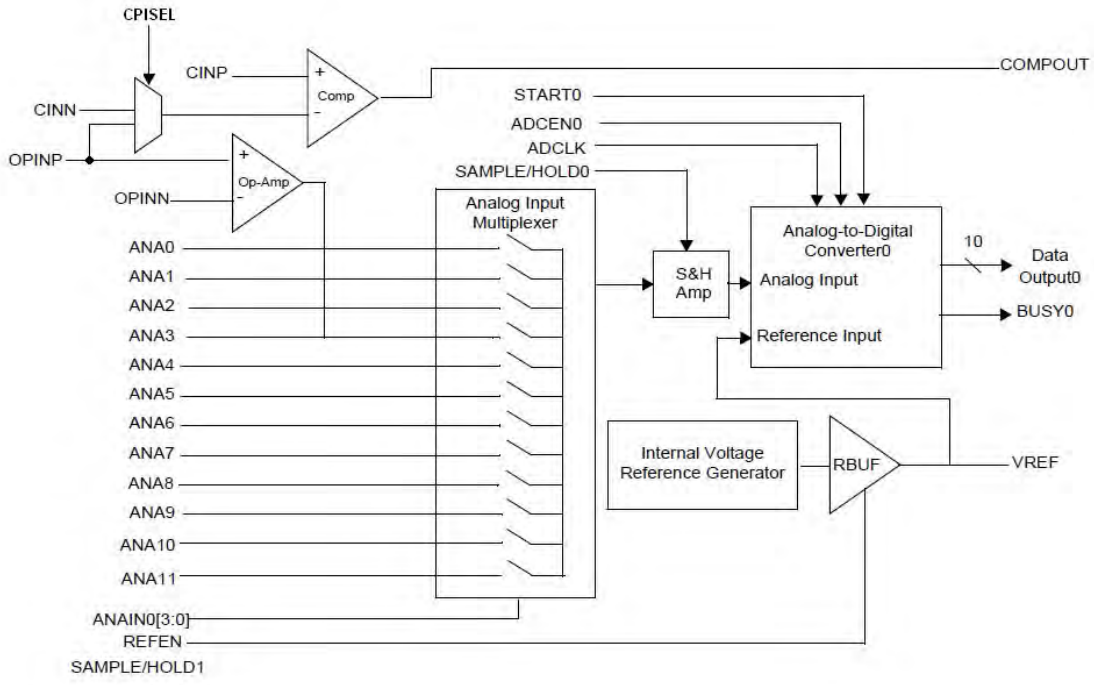


Figure 52. Analog Functions Block Diagram

ADC Overview

The ZNEO Z16F Series devices include a 12-channel ADC. The ADC converts an analog input signal to a 10-bit binary number. The features of the successive approximation ADC include:

- 12 analog input sources multiplexed with GPIO ports
- Fast conversion time (2.5 µs)
- Programmable timing controls
- Interrupt on conversion complete
- Internal voltage reference generator
- Internal reference voltage available externally
- Ability to supply external reference voltage
- Ability to perform simultaneous or independent conversions

Architecture

The architecture as illustrated in Figure 52 consists of a 12-input multiplexer, sample-and-hold amplifier and 10-bit successive approximation ADC. The ADC digitizes the signal on selected channel and stores the digitized data in the ADC data registers. In environment with high electrical noise, an external RC filter must be added at the input pins to reduce high frequency noise.

Operation

The ADC converts the analog input, ANA*x*, to a 10-bit digital representation. The equation for calculating the digital value is:

ADC Output = 1024*(ANAx/VREF)

Assuming zero gain and offset errors, any voltage outside the ADC input limits of AVSS and VREF returns all 0s or 1s, respectively.

A new conversion is initiated by either software write to the ADC Control register's Start bit or by PWM trigger. For detailed information about the PWM trigger, see the Synchronization of PWM and ADC section on page 120. Initiating a new conversion stops any conversion currently in progress and begins a new conversion. To avoid disrupting a con-

version already in progress, the Start bit is read to indicate ADC operation status (busy or available).

ADC Timing

Each ADC measurement consists of three phases:

1. Input sampling (programmable, minimum of 1.0 µs).
2. Sample-and-hold amplifier settling (programmable, minimum of 0.5 µs).
3. Conversion is 12 ADCLK cycles.

Figure 53 displays the timing of an ADC conversion.

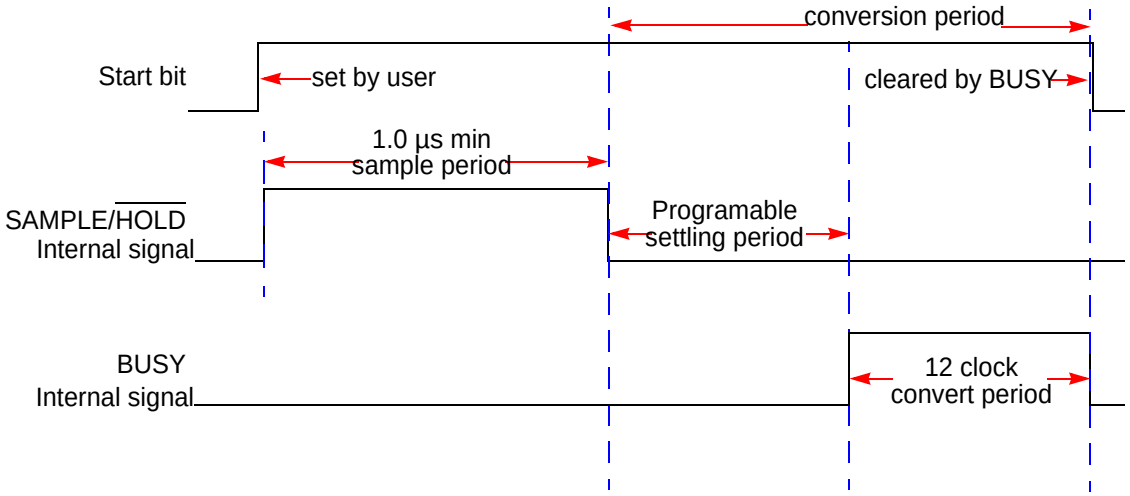


Figure 53. ADC Timing Diagram

Figure 54 displays the timing of convert period showing the 10 bit progression of the output.

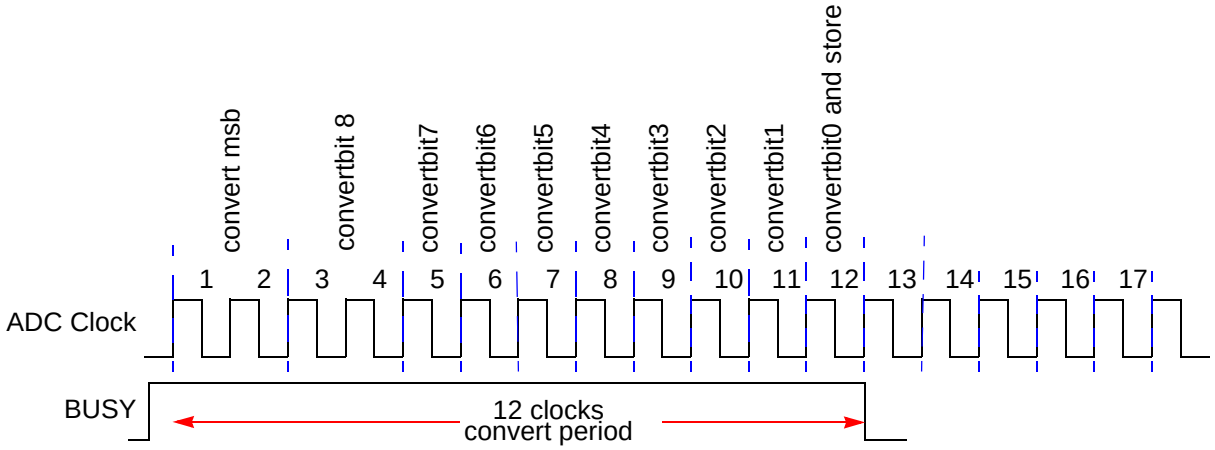


Figure 54. ADC Convert Timing

ADC Interrupts

The ADC generates an interrupt request when a conversion has been completed. An interrupt request pending when the ADC is disabled is not automatically cleared.

ADC0 Timer 0 Capture

The Timer 0 count is captured for every ADC0 conversion. The information is used to determine the zero crossing of back EMF in motor control applications. The capture of the Timer 0 count occurs when the programmed sample time is complete for every conversion and stored in the ADC timer capture register (ADCTCAP).

ADC Convert on Read

The ADC is set up to automatically convert the next channel input after reading the results of the current conversion. The conversions continue up to the channel listed in the ADC0MAX Register and then start over at the initial channel. The initial channel to convert is written to the control register, ADC0CTL, prior to starting the convert on Read process. Once started, the conversions continue to loop from the initial channel to Max channel until the convert on Read bit, CVTRD0, is cleared or the data is not read from the data registers.

Reference Buffer, RBUF

The reference buffer, RBUF, supplies the reference voltage for the ADC. When enabled, the internal voltage reference generator supplies the ADC and the voltage is available on the VREF pin. When RBUF is disabled, the reference voltage must be supplied externally through the VREF pin. RBUF is controlled by the REFEN bit in the ADC0 Control Register.

Internal Voltage Reference Generator

The internal voltage reference generator provides the voltage to RBUF. The internal reference voltage is 2 V.

ADC Control Register Definitions

The following sections describe the control registers for the ADC.

ADC0 Control Register 0

The ADC0 Control Register initiates the A/D conversion and provides ADC0 status information.

Table 126. ADC0 Control Register 0 (ADC0CTL)

Bits	7	6	5	4	3	2	1	0
Field	START0	CVTRD0	REFEN	ADC0EN	ANAIN0[3:0]			
RESET	0	0	0	0	0	0	0	0
R/W	R/W1	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF–E500h							

Bit	Description
[7] START0	**ADC0 Start/Busy** 0 = Writing to 0 has no effect. Reading a 0 indicates the ADC0 is available to begin a conversion. 1 = Writing to 1 starts a conversion on ADC0. Reading a 1 indicates a conversion is currently in progress.
[6] CVTRD0	**Convert On Read** 0 = The ADC0 operates normally. 1 = If this bit is set to 1, whenever the ADC0D Register is read it increments the ANAIN field by one and start a new conversion. The ANAIN field increments until it reaches the value set in the ADC0MAX Register. After doing the conversion on the channel specified by the ADC0MAX Register, the next read resets the ANAIN field to 0. This function is used with the DMA to perform continuous conversions.

Bit	Description (Continued)
[5] REFEN	**Reference Enable** 0 = Internal reference voltage is disabled allowing an external reference voltage to be used by the ADC0. 1 = Internal reference voltage for the ADC0 is enabled. The internal reference voltage is measured on the VREF pin.
[4] ADC0EN	**ADC0 Enable** 0 = ADC0 is disabled for low power operation. 1 = ADC0 is enabled for normal use.
[3:0] ANAIN0	**Analog Input Select** 0000 = ANA0 input is selected for analog-to-digital conversion. 0001 = ANA1 input is selected for analog-to-digital conversion. 0010 = ANA2 input is selected for analog-to-digital conversion. 0011 = ANA3 input is selected for analog-to-digital conversion. 0100 = ANA4 input is selected for analog-to-digital conversion. 0101 = ANA5 input is selected for analog-to-digital conversion. 0110 = ANA6 input is selected for analog-to-digital conversion. 0111 = ANA7 input is selected for analog-to-digital conversion. 1000 = ANA8 input is selected for analog-to-digital conversion. 1001 = ANA9 input is selected for analog-to-digital conversion. 1010 = ANA10 input is selected for analog-to-digital conversion. 1011 = ANA11 input is selected for analog-to-digital conversion. 1100 = This bit is reserved and must be programmed to 0. 1111 = This bit is reserved and must be programmed to 0.

ADC0 Data High Byte Register

The ADC0 Data High Byte Register contains the upper eight bits of the ADC0 output. Access to the ADC0 Data High Byte Register is read-only.

Table 127. ADC0 Data High Byte Register (ADC0D_H)

Bits	7	6	5	4	3	2	1	0
Field	ADC0D_H							
RESET	X							
R/W	R							
Addr	FF–E502h							

Bit	Description
[7:0] ADC0D_H	**ADC0 High Byte** 00h–FFh = The last conversion output is held in the data registers until the next ADC conversion is completed.

ADC0 Data Low Bits Register

The ADC0 Data Low Bits Register contains the lower bits of the ADC0 output. Access to the ADC0 Data Low Bits Register is Read-Only.

Table 128. ADC0 Data Low Bits Register (ADC0D_L)

Bits	7	6	5	4	3	2	1	0
Field	ADC0D_L		Reserved					
RESET	X		X					
R/W	R		R					
Addr	FF–E503h							

Bit	Description
[7:6] ADC0D_L	**ADC0 Low Bits** 00–11b = These bits are the 2 least significant bits of the 10-bit ADC0 output. These bits are undefined after a Reset.
[5:0]	**Reserved** These bits are reserved and must be programmed to 0.

Sample Settling Time Register

The Sample Settling Time Register is used to program the length of time from the SAMPLE/$\overline{\text{HOLD}}$ signal to the Start signal, when the conversion begins. The number of clock cycles required for settling varies from system to system depending on the system clock period used. This register must be programmed to contain the number of clocks required to meet a 0.5µs minimum settling time.

Table 129. Sample and Settling Time (ADCSST)

Bits	7	6	5	4	3	2	1	0
Field	Reserved			SST				
RESET	0	0	0	1	1	1	1	1
R/W	R			R/W				
Addr	FF–E504h							

Bit	Description
[7:5]	**Reserved** These bits are reserved and must be programmed to 0.
[4:0] SST	**Sample Settling Time** 00h–1Fh = Sample settling time in number of system clock periods to meet 0.5µs minimum.

Sample Time Register

The Sample Time Register is used to program the length of active time for the sample after a conversion has begun by setting the Start bit in the ADC Control Register or initiated by the PWM. The number of system clock cycles required for sample time varies from system to system depending on the clock period used. This register must be programmed to contain the number of system clocks required to meet a 1 µs minimum sample time.

Table 130. Sample Time (ADCST)

Bits	7	6	5	4	3	2	1	0
Field	Reserved		ST					
RESET	0		1	1	1	1	1	1
R/W	R		R/W					
Addr	FF–E505h							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:0] SHT	**Sample Hold Time** 00h–3Fh = Sample Hold time in number of system clock periods to meet 1 µs minimum.

ADC Clock Prescale Register

The ADC Clock Prescale Register is used to provide a divided system clock to the ADC. When this register is programmed with 0h, the system clock is used for the ADC Clock.

Table 131. ADC Clock Prescale Register (ADCCP)

Bits	7	6	5	4	3	2	1	0
Field	Reserved				DIV16	DIV8	DIV4	DIV2
RESET	0				0	0	0	0
R/W	R				R/W			
Addr	FF–E506h							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] DIV16	**DIV16** 0 = Clock is not divided. 1 = System Clock is divided by 16 for ADC Clock.

Bit	Description (Continued)
[2] DIV8	**DIV8** 0 = Clock is not divided. 1 = System Clock is divided by 8 for ADC Clock.
[1] DIV4	DIV4 0 = Clock is not divided. 1 = System Clock is divided by 4 for ADC Clock.
[0] DIV2	DIV2 0 = Clock is not divided. 1 = System Clock is divided by 2 for ADC Clock.

ADC0 Max Register

The ADC0 Max Register determines the highest channel that the Convert on Read increments to.

Table 132. ADC0 MAX Register (ADC0MAX)

Bits	7	6	5	4	3	2	1	0
Field	Reserved				LASTCHAN0			
RESET	0				0h			
R/W	R/W							
Addr	FF–E507h							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3:0] LASTCHAN0	**Last Channel 0** 0 = These bits determine the last channel number to increment to when the Convert On Read is set.

ADC Timer 0 Capture Register

The ADC Timer 0 Capture Register contains the sixteen bits of the ADC Timer 0 count. The access to the ADC Timer 0 Capture Register is read-only. It reads 8 bits at a time or as a 16-bit word.

Table 133. ADC Timer 0 Capture Register, High Byte (ADCTCAP_H)

Bits	7	6	5	4	3	2	1	0
Field	ADCTCAPH							
RESET	X							
R/W	R							
Addr	FF–E512h							

Bit	Description
[7:0] ADCTCAPH	**ADC Timer 0 Count High Byte** 00h–FFh = The Timer 0 count is held in the data registers until the next ADC conversion is started.

Table 134. ADC Timer 0 Capture Register, Low Byte (ADCTCAP_L)

Bits	7	6	5	4	3	2	1	0
Field	ADCTCAPL							
RESET	X							
R/W	R							
Addr	FF–E513h							

Bit	Description
[7:0] ADCTCAPL	**ADC Timer 0 Count Low Byte** 00h–FFh = The Timer 0 count is held in the data registers until the next ADC conversion is started.

Comparator and Operational Amplifier Overview

ZNeo devices feature a general-purpose comparator and an operational amplifier. The comparator is a moderate speed (200 ns propagation delay) device which is designed for a maximum input offset of 5 mV. The comparator is used to compare two analog input signals. General-purpose input pins (CINP and CINN) provides the comparator inputs. The output is available as an interrupt source.

The operational amplifier is a two-input, one-output operational amplifier with a typical open loop gain of 10,000 (80 dB). The general-purpose input pin (OPINP) provides the noninverting amplifier input, while general-purpose input pin (OPINN) provides the inverting amplifier input. The output is available at the output pin (OPOUT).

The key operating characteristics of the operational amplifier are:

- Frequency compensated for unity gain stability
- Input common-mode-range from GND (0.0 V) to V_{DD} – 1 V
- Input offset voltage less than 15 mV
- Output voltage swing from GND + 0.1 V to V_{DD} – 0.1 V
- Input bias current less than 1 µA
- Operating the operational amplifier open loop (no feedback) effectively provides another on-chip comparator

Comparator Operation

The comparator output reflects the relationship between the noninverting input and the inverting (reference) input. If the voltage on the noninverting input is higher than the voltage on the inverting input, the comparator output is at a high state. If the voltage on the noninverting input is lower than the voltage on the inverting input, the comparator output is at a low state.

To operate, the comparator must be enabled by setting the CMPEN bit in the comparator and op-amp register to 1. In addition the CINP and CINN comparator input alternate functions must be enabled on their respective GPIO pins. For more information, see the GPIO Alternate Functions section on page 67.

The comparator does not automatically power-down. To reduce operating current when not in use, the comparator is disabled by clearing the CMPEN bit to 0.

Operational Amplifier Operation

To operate, the operational amplifier must be enabled by setting the OPEN bit in the comparator and op-amp register to 1. In addition, the OPINP, OPINN and OPOUT alternate functions must be enabled on their respective general-purpose I/O pins. For more information, see GPIO Alternate Functions.

The logical value of the operational amplifier output (OPOUT) is read from the Port 3 data input register if both the operational amplifier and input pin Schmitt trigger are enabled. For more information, see GPIO Alternate Functions. The operational amplifier generates an interrupt via the GPIO Port B3 input interrupt, if enabled.

The output of the operational amplifier is also connected to an analog input (ANA3) of the ADC multiplexer.

The operational amplifier does not automatically power-down. To reduce operating current when not in use, the operational amplifier is disabled by clearing the OPEN bit in the comparator and op-amp register to 0.

When the operational amplifier is disabled, the output is high impedance.

Interrupts

The comparator generates an interrupt on any change in the logic output value (from 0 to 1 and from 1 to 0). For information about enabling and prioritization of the comparator interrupt, see the Interrupt Controller chapter on page 80.

Comparator Control Register Definitions

The following sections describe the comparator control registers.

Comparator and Operational Amplifier Control Register

The Comparator and Operational Amplifier Control Register (CMPOPC), shown in Table 135, enables the comparator and operational amplifier and provides access to the comparator output.

Table 135. Comparator and Op Amp Control Register (CMPOPC)

Bits	7	6	5	4	3	2	1	0
Field	OPEN	Reserved		CPISEL	CMPIRQ	CMPIV	CMPOUT	CMPEN
RESET	0	00		0	0	0	X	0
R/W	R/W	R		R/W	R/W	R/W	R	R/W
Addr	FF_E510h							

Bit	Description
[7] OPEN	**Operational Amplifier Disable** 0 = Operational amplifier is disabled. 1 = Operational amplifier is enabled.
[6:5]	**Reserved** These bits are reserved and must be programmed to 00.

Bit	Description (Continued)
[4] CPISEL	**Comparator Input Select** 0 = PortB6 provides the comparator - input. 1 = PortC0 provides the comparator - input.
[3] CMPIRQ	**Comparator Interrupt Edge Select** 0 = Interrupt Request on Comparator Rising Edge. 1 = Interrupt Request on Comparator Falling Edge.
[2] CMPIV	**PWM Fault Comparator Polarity** 0 = PWM Fault is active when cp+ > cp- 1 = PWM Fault is active when cp- > cp+
[1] CMPOUT	**Comparator Output Value** 0 = Comparator output is logical 0. 1 = Comparator output is logical 1.
[0] CMPEN	**Comparator Enable** 0 = Comparator is disabled. 1 = Comparator is enabled.

Flash Memory

The products in the ZNEO® Z16F Series feature up to 128 KB of nonvolatile Flash memory with read/write/erase capability. The Flash memory is programmed and erased in-circuit by either user code or through the OCD.

The Flash memory array is arranged in 2 KB pages. The 2 KB page is the minimum Flash block size that is erased. The Flash memory is also divided into eight sectors, which is protected from programming and erase operations on a per sector basis.

Table 136 describes the Flash memory configuration for each device in the ZNEO Z16F Series. Table 137 lists the sector address ranges. Figure 55 displays the Flash memory arrangement.

Table 136. Flash Memory Configurations

Part Number	Internal Flash Size	Number of Pages	Program Memory Addresses	Sector Size	Number of Sectors	Pages per Sector
Z16F2811	128 KB	64	000000h–01FFFFh	16 KB	8	8
Z16F2810	128 KB	64	000000h–01FFFFh	16 KB	8	8
Z16F6411	64 KB	32	0000h–FFFFh	8 KB	8	4
Z16F3211	32 KB	16	0000h–7FFFh	4 KB	8	2

Table 137. Flash Memory Sector Addresses

Sector Number	Flash Sector Address Ranges		
	Z16F2811/Z16F2810	Z16F6411	Z16F3211
0	000000h–003FFFh	000000h–001FFFh	000000h–000FFFh
1	004000h–007FFFh	002000h–003FFFh	001000h–001FFFh
2	008000h–00BFFFh	004000h–005FFFh	002000h–002FFFh
3	00C000h–00FFFFh	006000h–007FFFh	003000h–003FFFh
4	010000h–013FFFh	008000h–009FFFh	004000h–004FFFh
5	014000h–017FFFh	00A000h–00BFFFh	005000h–005FFFh
6	018000h–01BFFFh	00C000h–00DFFFh	006000h–006FFFh
7	01C000h–01FFFFh	00E000h–00FFFFh	007000h–007FFFh

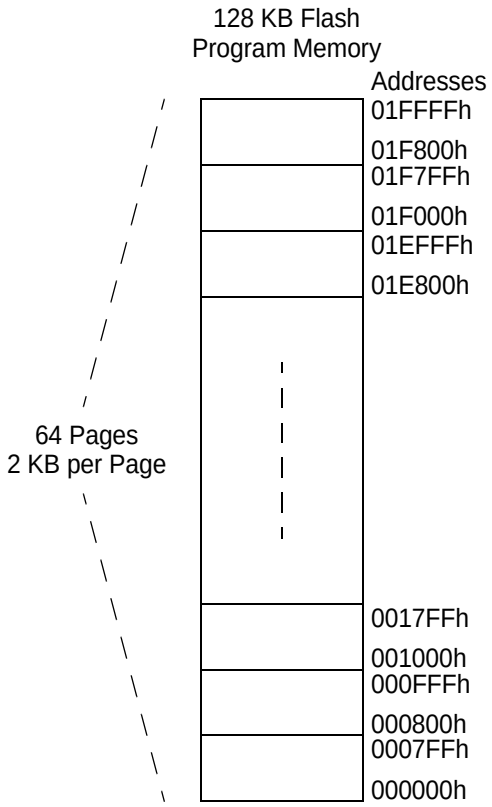


Figure 55. Flash Memory Arrangement

Information Area

Table 138 describes the ZNEO Z16F Series Information Area. This 128-byte Information Area is accessed by setting bit 7 of the Flash Control register to 1. When access is enabled, the Information Area is mapped into program memory and overlays the 128 bytes at addresses 000000h to 00007Fh. When the Information Area access is enabled, instructions access data from the Information Area. The CPU instruction fetches always come from Main Memory regardless of the Information Area access bit. Access to the Information Area is read-only.

Table 138. ZNEO Z16F Series Information Area Map

Program Memory Address (Hex)	Function
000000h–00003Fh	Reserved.
000040h–000053h	Part Number: 20-character ASCII alphanumeric code, left-justified and padded with zeros.
000054h–00007Fh	Reserved.

Operation

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

Timing Using the Flash Frequency Register

Before performing a program or erase operation on the Flash memory, you must first configure the Flash Frequency Register. The Flash Frequency Register allows programming and erasure of the Flash with system clock frequencies ranging from 32 kHz through 20 MHz (the valid range is limited to the device operating frequencies).

The 16-bit Flash Frequency Register must be written with the system clock frequency in kHz before a program or erase operation is initiated. This value is calculated using the following equation:

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

Caution: Flash programming and erasure is not supported for system clock frequencies below 32 kHz, above 20 MHz, or outside of the device operating frequency range. The Flash Frequency Register must be loaded with the correct value to ensure proper Flash programming and erase operations.

Flash Read Protection

The user code within the Flash memory is protected from external access. Programming the Flash Read Protect option bit prevents reading of user code by the OCD or by using the Flash Controller Bypass Mode. For more information, see the Option Bits chapter on page 292 and the On-Chip Debugger chapter on page 298.

Flash Write/Erase Protection

The ZNEO Z16F Series provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by the Flash Controller unlock mechanism, the Flash Sector Protect register and the Flash Write Protect option bit.

Flash Controller Unlock Mechanism

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

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

1. Write the page to be programmed or erased to the Flash Page Select Register.
2. Write the first unlock command 73h to the Flash Command Register.
3. Write the second unlock command 8Ch to the Flash Command Register.

Flash Sector Protection

The Flash Sector Protect register is 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 will be lost. User code must write this register in their initialization routine if they want to enable sector protection.

When user code writes the Flash Sector Protect register, bits are set to 1 only. Thus, sectors are protected, but not unprotected, using register write operations.

Flash Write Protection Option Bit

The Flash Write Protect option bit is enabled to block all program and erase operations from user code. For more detail, see the Option Bits chapter on page 292.

Programming

When the Flash Controller is unlocked, word writes to Program memory from user code programs a word into the Flash if the address is located in the unlocked page. An erased Flash word contains all ones (FFFFh). The programming operation is used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from 0 to 1 requires a Page Erase or Mass Erase operation.

The Flash must be programmed one word (16-bits) at a time. If a byte (8-bit) write to Flash memory occurs, the Flash controller waits until the other byte within the word is written before beginning the programming operation.

While the Flash Controller programs the Flash memory, Flash reads are held in wait. If the CPU is fetching instruction from Flash, the CPU idles until the programming operation is complete. 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 Command Register.

User code cannot program Flash Memory on a page that lies 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 steps to program the Flash from user code:

1. Write the page of memory to be programmed to the Flash Page Select Register.
2. Write the first unlock command 73h to the Flash Command Register.
3. Write the second unlock command 8Ch to the Flash Command Register.
4. Write a word to Program memory.
5. Repeat step 4 to program additional memory locations on the same page.
6. Write 00h to the Flash Command Register to lock the Flash Controller.

Page Erase

The Flash memory is erased one page (2 KB) at a time. Page Erasing the Flash memory sets all words in that page to the value FFFFh. The Flash Page Select Register identifies the page to be erased. While the Flash Controller executes the Page Erase operation, Flash reads are held in wait. Interrupts that occur when the Page Erase operation is in progress will be serviced after the Page Erase operation is complete. When the Page Erase opera-

tion is complete, the Flash Controller returns to its locked state. Only pages located in unprotected sectors are erased.

The four steps to performing a Page Erase operation are:

1. Write the page to be erased to the Flash Page Select Register.
2. Write the first unlock command 73h to the Flash Command Register.
3. Write the second unlock command 8Ch to the Flash Command Register.
4. Write the Page Erase command 95h to the Flash Command Register.

Mass Erase

The Flash memory cannot be Mass Erased by user code.

Flash Controller Bypass

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

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

Flash Controller Behavior using the On-Chip Debugger

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

- The Flash Controller does not have to be unlocked for program and erase operations.
- 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 Flash Page Select Register.
- Bits in the Flash Sector Protect register is written to 1 or 0.
- The Flash Page Select Register is written when the Flash Controller is unlocked.
- The Mass Erase command is enabled.

Flash Control Register Definitions

Flash Command Register

The Flash Command Register, shown in Table 139, unlocks the Flash Controller for programming and erase operations. The Write-only Flash Command Register shares its address with the Read-only Flash Status Register.

Table 139. Flash Command Register (FCMD)

Bits	7	6	5	4	3	2	1	0
Field	FCMD							
RESET	XXH							
R/W	W							
Addr	FF_E060h							

Bit	Description
[7:0] FCMD	**Flash Command** 73h = First unlock command. 8Ch = Second unlock command. 95h = Page erase command. 63h = Mass erase command.

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

Flash Status Register

The Flash Status Register, shown in Table 140, indicates the current state of the Flash Controller. This register is read at any time. The Read-only Flash Status Register shares its address with the Write-only Flash Command Register.

Table 140. Flash Status Register (FSTAT)

Bits	7	6	5	4	3	2	1	0
Field	UNLOCK	Reserved	FSTAT					
RESET	0	0	00h					
R/W	R	R	R					
Addr	FF_E060h							

Bit	Description
[7] UNLOCK	**Unlocked** This status bit is set when the Flash Controller is unlocked. 0 = Flash Controller locked. 1 = Flash Controller unlocked.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5:0] FSTAT	**Flash Controller Status** 00_0000 = Flash Controller idle. 00_1xxx = Program operation in progress. 01_0xxx = Page erase operation in progress. 10_0xxx = Mass erase operation in progress.

Flash Control Register

The Flash Control Register, shown in Table 141, selects how the Flash memory is accessed.

Table 141. Flash Control Register (FCTL)

Bits	7	6	5	4	3	2	1	0
Field	INFO	Reserved						
RESET	0	00h						
R/W	R/W	R						
Addr	FF_E061h							

Bit	Description
[7] INFO	**Information Area Access** This bit selects access to the information area. 0 = Information Area is not selected. 1 = Information Area is selected. The Information area is mapped into the Program memory address space at addresses 000000h through 00007Fh.
[6:0]	**Reserved** These bits are reserved and must be programmed to 0000000.

Flash Sector Protect Register

The Flash Sector Protect Register, shown in Table 142, protects Flash memory sectors from being programmed or erased from user code. User code can only write bits in this register to 1 (bits cannot be cleared to 0 by user code).

Table 142. Flash Sector Protect Register (FSECT)

Bits	7	6	5	4	3	2	1	0
Field	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
RESET	0	0	0	0	0	0	0	0
R/W	R/W1	R/W1	R/W1	R/W1	R/W1	R/W1	R/W1	R/W1
Addr	FF_E062h							

Note: R/W1 = Register is accessible for read operations. Register is written to 1 only (via user code).

Bit	Description
[7:0] SECT*n*	**Sector Protect** 0 = Sector *n* is programmed or erased from user code. 1 = Sector *n* is protected and cannot be programmed or erased from user code.

Note: User code write bits from 0 to 1 only.

Flash Page Select Register

The Flash Page Select (FPAGE) Register, shown in Table 143, selects one of the 64 available Flash memory pages to be erased or programmed. Each Flash Page contains 2048 words of Flash memory. During a Page Erase operation, all Flash memory locations within the page will be erased to FFFFh.

Table 143. Flash Page Select Register (FPAGE)

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved							PAGE						Reserved		
RESET	00h							00h						0h		
R/W	R							R/W						R		
Addr	FF_E064–FF_E065h															

Bit	Description
[15:9]	**Reserved** These bits are reserved and must be programmed to 00000000.
[8:3] PAGE	**Page Select** This 6-bit field selects the Flash memory page for Programming and Page Erase operations. Program Memory Address[16:11] = FPAGE[8:3] = PAGE[5:0].
[2:0]	**Reserved** These bits are reserved and must be programmed to 000.

Flash Frequency Register

The Flash Frequency Register, shown in Table 144, sets the time for Flash program and erase operations. The 16-bit Flash Frequency Register must be written with the system clock frequency in kiloHertz. The Flash Frequency value is calculated using the following equation:

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

Caution: Flash programming and erasure is not supported for system clock frequencies below 32 kHz, above 20 MHz, or outside of the valid operating frequency range for the device. The Flash Frequency Register must be loaded with the correct value to ensure proper program and erase times.

Table 144. Flash Frequency Register (FFREQ)

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	FFREQ															
RESET	0000h															
R/W	R/W															
Addr	FF_E066-FFE067h															

Bit	Description
[15:0] FFREQ	**Flash Frequency** This value is used to time Flash program and erase operations.

DMA Controller

The ZNEO Z16F Series' four DMA channels are used to transfer data from memory to memory, memory to peripherals, peripherals to memory, or peripherals to peripherals.

DMA Features

The features of the DMA Controller include:

- Four independent DMA channels.
- Memory <=> memory, memory <=> peripheral, peripheral <=> memory, peripheral <=> peripheral transfers
- Direct or Linked List modes of operation
- Byte, word, or quad operation
- DMA and CPU bandwidth sharing control
- Up to 64 K transfers (64 KByte, 64 KWord or 64 KQuad)
- External DMA request and DMA acknowledge signals

DMA Block Diagram

Figure 56 shows the blocks that comprise the DMA Controller.

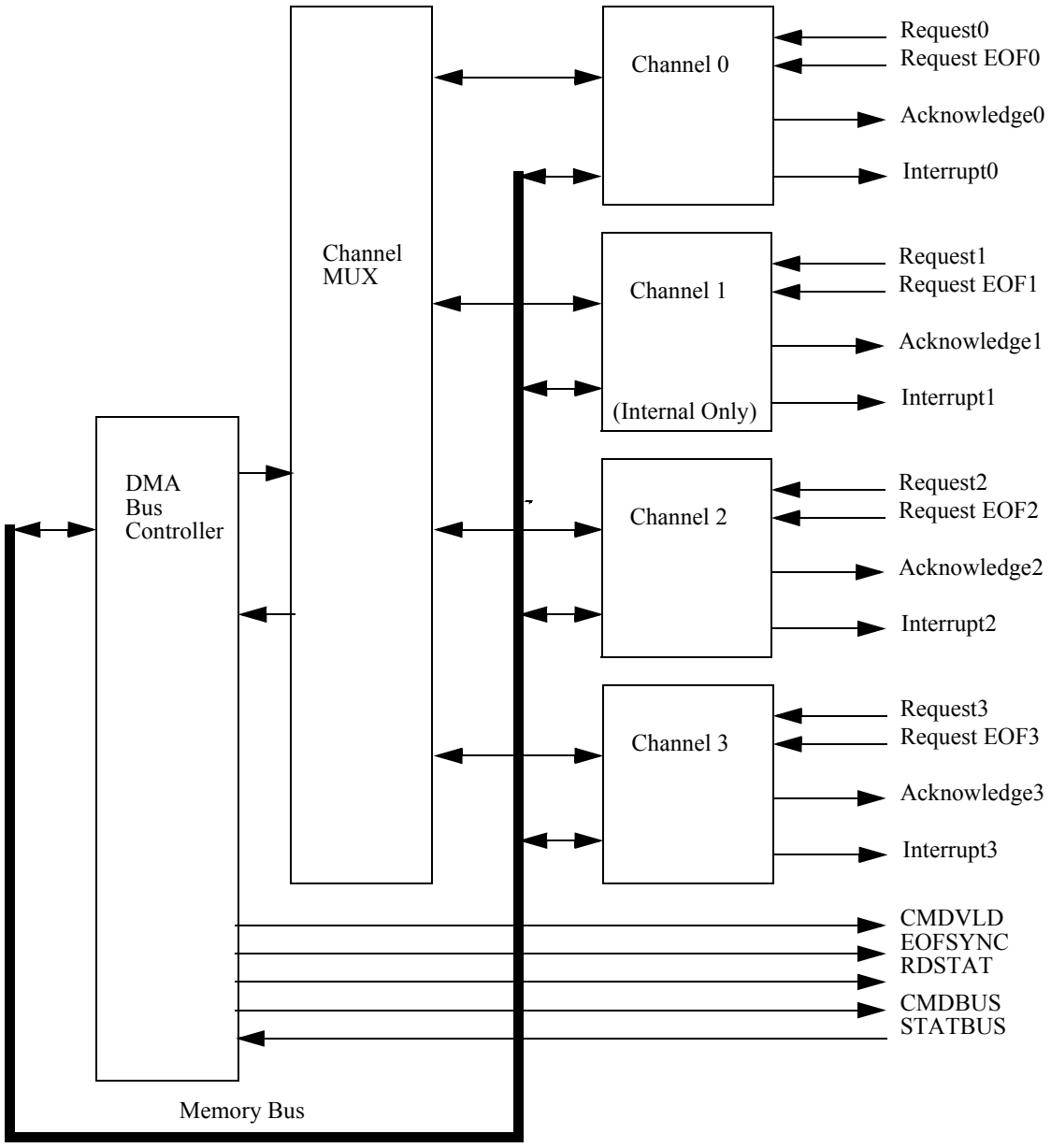


Figure 56. DMA Block Diagram

DMA Description

The DMA is used to off load the processor from doing repetitive tasks. DMA transfers data from one memory address to another memory address. Because all peripherals are mapped in memory, the DMA transfers data to or from peripherals.

The DMA transfers data from the source address to the destination address. This requires a read and/or write cycle that is generated by the DMA Controller. Each DMA transfer requires a minimum of two system clock cycles to execute.

The DMA operates in Direct or Linked List modes. Direct Mode and Linked List Mode are almost the same. In Direct Mode, the software loads the DMA Channel registers directly. In Linked List Mode, the DMA loads its registers from memory.

DMA Register Description

Each DMA channel consists of a 16-bit control register, a 16-bit transfer length register, a 24-bit destination address register, a 24-bit source address register and a 24-bit list address register; see Figure 57.

DMA Control (DMACTL)
Transfer Length (TXLN)
Destination Address (DAR)
Source Address (SAR)
List Address (LAR)

Figure 57. DMA Channel Registers

Buffers

A buffer is an allocation of contiguous memory bytes. Buffers are allocated by software to be used by the DMA. The DMA transfers data to or from buffers. A typical application would be to send data to serial channels such as I^2C, UART and SPI. The data to be sent is placed in a buffer by software.

Frames

A frame is a single buffer or a collection of buffers. Frame boundaries spans multiple buffers.

Source Address Register

The source address register (SAR) points to the data to be transferred. Each time a transfer occurs the SAR is selected to stay fixed or increment/decrement by the size of the transfer (example 1, 2, 4). If we were sending data to a serial channel, the SAR points to the data to be transferred and the SAR would be set to increment or decrement depending on the order of data in the buffer (ascending or desending).

Destination Address Register

The destination address register (DAR) points to the location to store the data transferred from the address pointed to by the SAR. Each time a transfer occurs the DAR is selected to stay fixed or increment/decrement by the size of the transfer (for example, 1, 2 and 4). When sending data to a serial channel, the DAR points to the data register of the serial channel and is set to a fixed address. Each transfer is then sent to the serial channel data register because the DAR would not change.

Transfer Length

The Transfer Length Register (TXLN) is used to specify how many transfers must occur to transfer this buffer. If we were sending bytes to a serial channel, the value of the number of bytes in the buffer pointed to by the SAR would be placed in this register. Each time a transfer takes place this register is decremented by one. When the transfer length decrements to 0, the buffer is complete and the DMA either stops or loads new control information and addresses. For additional detail, see the Linked List Mode section on page 277.

List Address Register

The list address register (LAR) is only used for LINKED LIST Mode. The LAR points to a list of descriptors (described in the Linked List Descriptor, Table 145). This descriptor list contains setup information for each buffer that the DMA will transfer. Linked-list DMAs reduce the amount of overhead on the CPU to service the DMA.

Descriptor

A descriptor is a 16-byte field in the memory space that must be aligned on 16-byte boundaries (i.e., the lower 4 bits of the address are 0). Table 145 provides the descriptor format.

Table 145. Linked List Descriptor

Address	Even
LAR	CONTROL
LAR + 02h	TXLN
LAR + 04h	DAR High
LAR + 08h	SAR High
LAR + 0Ch	LAR High

DMA Control Bit Definitions

The following paragraphs explain the control bits of each DMA Channel.

DMAxEN

This bit if set by the CPU enables the DMA Channel for direct operation. Direct operation uses the addresses and transfer length, which has been directly written to the DMA Channel by software.

If this bit is set by a descriptor read then LINKED LIST Mode is enabled. Linked list operation starts when an address is written to the DMAxLAR. This write causes the DMA to read in the descriptor control value and addresses and place them in the DMA Channel.

LOOP

If the DMA is in LINKED LIST Mode and this bit is set to 1, it prevents the DMA from updating the descriptor when the buffer is closed. This bit is set to allow lists to loop on themselves without software intervention.

TXSIZE

The TXSIZE bits set the width of the transfer, as follows:

00 8-bit bytes are transferred on each DMA transfer. The destination and source addresses increment or decrement by 1 for each transfer if the DSTCTL and/or SRCCTL is selected for increment or decrement. The transfer length is decremented by 1 to allow 64Kbytes to be transferred.

01 A 16-bit word is transferred on each DMA transfer. The destination and source addresses increment or decrement by 2 if the DSTCTL and/or SRCCTL is selected for increment or decrement. In Word Mode, the transfer length is still decremented by 1 to allow 64Kwords to be transferred.

10 A 32-bit quad is transferred on each DMA transfer. The destination and source addresses increment or decrement by 4 if the DSTCTL and/or SRCCTL is selected for increment or decrement. In Quad Mode, the transfer length is still decremented by 1 to allow 64Kquads to be transferred.

DSTCTL and SRCCTL Fields

The DSTCTL and SRCCTL fields control the increment or decrement of the source and destination addresses. The address is set to increment, decrement or not change on each DMA transfer.

00 = Fixed
01 = Increment
10 = Decrement
11 = Reserved

Interrupt on End Of Buffer (IEOB)

The Interrupt On End Of Buffer bit forces the DMA Channel to generate an interrupt when the buffer is closed. If the DMA is operating in Direct Mode and the TXLN decrements to the watermark value (see the DMA Watermark section on page 273) and this bit is set, then an interrupt is also generated.

Transfer List (TXFR)

If the DMA is operating in Linked List Mode and this bit is set, the DMA uses the next LAR address in the descriptor for the next descriptor address instead of incrementing the current DMAxLAR address by 16. As a result, the looping of true linked lists with buffers is allowed following the descriptor or transfers to other loops.

End of Frame (EOF)

If this bit is set, the EOF signal is sent to the peripheral on the last transfer in the buffer (i.e., TXLN == 1). This action signals the peripheral to close the frame (only used for on-chip peripherals). This bit is also set if a peripheral requests an end of frame before the buffer transfer is completed.

Halt After This Buffer (HALT)

If this bit is set, then the DMA stops after this buffer is closed. The DMAxLAR points to the next descriptor, but the descriptor will not be fetched.

Command Status (CMDSTAT)

These four bits are exported to the requesting device on the CMDBUS on the first transfer of a new buffer. These bits are set by a software write or from the DMA reading the descriptor. At the end of a buffer, these four bits will convey the status from the peripheral if the EOF bit is set. See the appropriate Zilog product specification(s) for the definitions of these commands and statuses.

DMA Watermark

When operating in DIRECT Mode, the DMAxLAR[23:16] byte is used as a watermark interrupt. If these bits are set to any value other than 0, they are compared to the low byte of the decremented transfer length during a transfer. If the IEOB bit is set and the upper byte of DMAxTXLN[15:8] is zero and DMAxTXLN[7:0] == DMAxLAR[23:16] then an interrupt is generated. This function allows the DMA Channel to generate an interrupt prior to the buffer becoming empty.

DMA Peripheral Interface signals

The DMA uses two input signals, four output signals and two 4-bit buses to communicate with the peripherals. The input signals are Request (REQ) and Request EOF. The output signals are Acknowledge (ACK), Command Valid (CMDVLD), End of Frame (EOF-SYNC) and Read Status (RDSTAT). The two 4-bit buses are Command Bus (CMDBUS) and Stat Bus (STATBUS).

A DMA transfer is initiated with the Request (REQ). When the DMA is servicing a Request from a peripheral it will assert its acknowledge signal (ACK) to let the peripheral know that a transfer is in progress. When the first byte of the transfer is written the CMD-VLD is asserted and the command bits are placed on the CMDBUS. The peripheral must latch the command from the bus when it sees this combination of signals.

If the EOF bit is set on the current buffer, then when the TXLN decrements to 0, the EOF-SYNC signal is asserted on the last data transfer to the peripheral. As a result, the periph-eral is informed that it has received the last byte in the frame.

After receiving the EOFSYNC signal, the peripheral must assert the Request EOF signal to the DMA to let the DMA know that the descriptor is closed. This could be immediately or at some later time if the data transferred still must be processed. For peripherals, which do not support a Request EOF, the EOFSYNC is tied to Request EOF to terminate the transfer.

After the Request EOF is asserted the DMA closes the descriptor. The DMA asserts the ACK and RDSTAT signal, if the descriptor EOF bit is set. The peripheral, if it has status, places it on the STATBUS. This status is then placed in the descriptor and DMA status bits when it is closed.

If a peripheral must close a descriptor because of an error or the end of a packet is reached then it asserts it is Request EOF. If the transfer length is not zero, then the DMA will set the EOF bit, close the descriptor and generate an interrupt.

Buffer Closure

A DMA buffer closure is requested in two ways. The first is when the transfer length reaches zero. The second is when the DMA receives a request end of frame from the peripheral. When either of these cases occur, the DMA begins closure of the buffer.

Loop Mode Closure

If the LOOP bit is set then the current buffer descriptor is not modified. The DMAxLAR increments or a new LAR value is fetched from the descriptor.

EOF Closure

The DMAxEN bit is reset to 0. If the EOF bit is set, the CMDSTAT field is set with the status data from the peripheral. If the channel is in LINKED LIST Mode then the DMAx-CTL word is written back to the CONTROL word of the descriptor. The DMAxLAR increments or is loaded with new LAR data from the descriptor if the TXFR bit is set.

Normal Closure

The DMAxEN bit is reset to 0. If the channel is in LINKED LIST Mode then the DMAx-CTL word is written back to the CONTROL word of the descriptor. The DMAxLAR increments or is loaded with new LAR data from the descriptor if the TXFR bit is set.

DMA Modes

Each DMA channel operates in two modes, direct and linked list. Both modes use the DMA Channel registers. The only difference is in how they are loaded. In DIRECT Mode, the DMA Channel registers are directly loaded by software and when the transfer is complete, the DMA stops. In LINKED LIST Mode, the DMA will load its own registers from a descriptor list which is pointed to by the DMAxLAR Register. It then loads the next descriptor in the list and continues executing.

The descriptor Control/Status field and address bytes maintain the same format as the control and address registers in the DMA.

Direct Mode

DIRECT Mode only uses the registers in the DMA for operation. The software writes these registers directly to set up and enable the DMA. DIRECT Mode is entered by directly setting the DMAxEN bit in the DMAxCTL0 Register.

Figure 58 displays the DMA registers and how they point to the buffers allocated in memory.

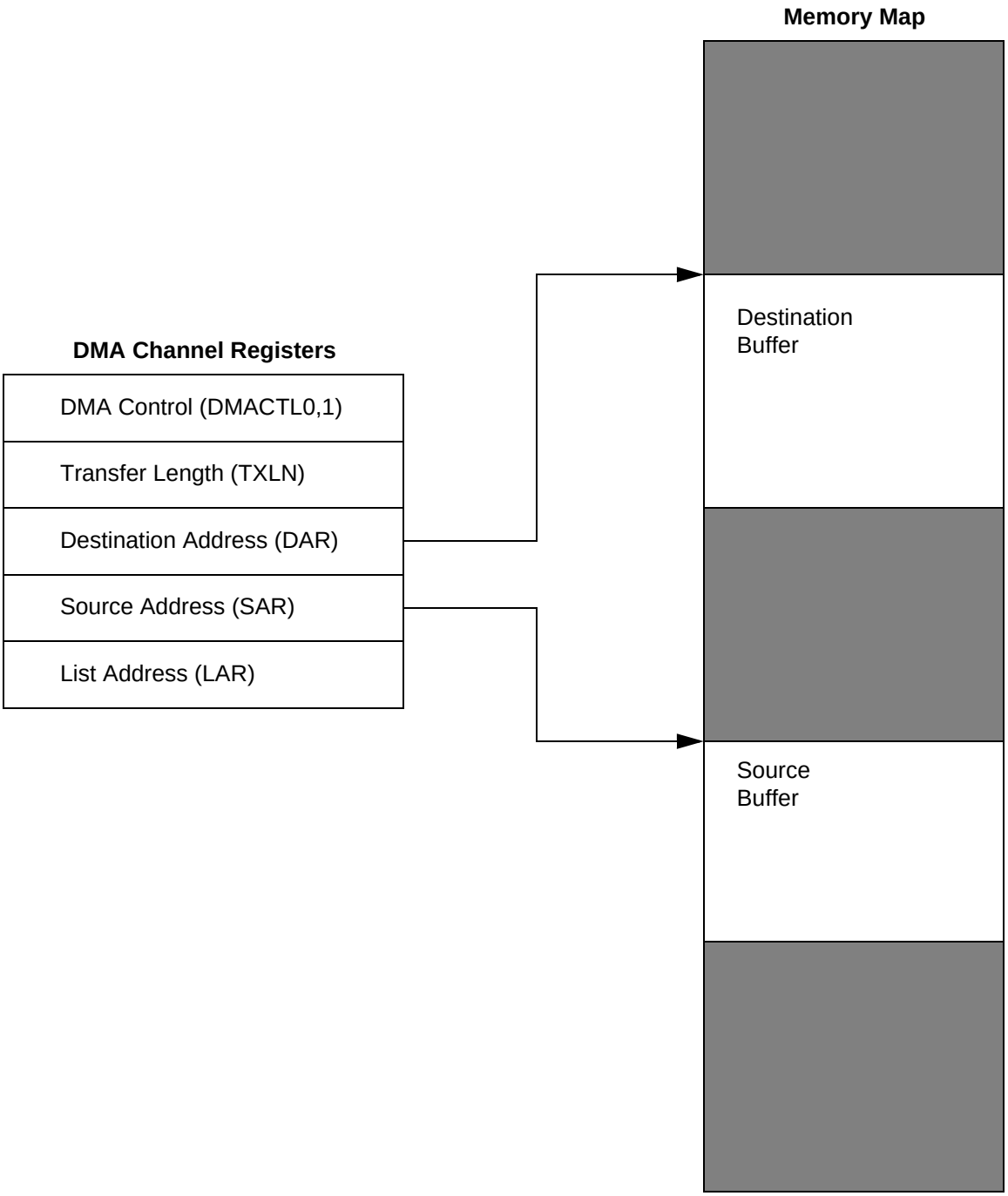


Figure 58. Direct DMA Diagram

Direct DMA Setup and Operation

Observe the following steps to set up the DMA in DIRECT Mode:

1. Write the DAMxREQSEL to select the request source.
2. Write the DMAxDAR Register with the destination address.
3. Write the DMAxSAR Register with the source address.
4. Write the DMAxTXLN with the transfer length.
5. Write DMAxLARU with watermark if required, otherwise write to 0.
6. Write DMAxCTL. Note that the control register and the address are written directly with word and quad operations.
   - DMA*x*EN: set to 1.
   - LOOP: reset to 0; not used in this mode.
   - TXSIZE: set to the transfer size, byte, word or quad.
   - DSTCTL: set to fixed, increment, or decrement.
   - SRCCTL: set to fixed, increment, or decrement.
   - IEOB: set to 1 to generate an interrupt at the end of buffer or watermark.
   - TXFR: reset to 0; not used in this mode.
   - EOF: set this bit to 1 if it is an EOF buffer.
   - HALT: reset to 0; not used in this mode.
   - CMDSTAT: set these bits with the command for the peripheral.
7. The DMA is now set up and begins operating when it receives a request.

After the DMA is set up and a request is received, the DMA performs the following operation:

1. Generates a request to the CPU.
2. Transfers data for each request until the transfer length reaches zero or the DMA receives a Request EOF signal.
3. When the DMA receives the Request EOF signal, or the transfer length reaches 0, it resets the DMAxEN bit, then performs the following operation, based upon the EOF and IEOB bits:
   a. If EOF is set, then the DMA reads the status from the peripheral and places it in the CMDSTAT field of the DMAxCTL Register.
   b. If the IEOB bit is set, or if the buffer ended with a Request EOF, the DMA Channel generates a request to the CPU.

c. If EOF is not set and IEOB is set, then the DMA Channel generates a request to the CPU.

Linked List Mode

Linked List Mode requires the software to allocate buffers and set up a list of descriptors for each buffer. After allocation is complete, the software writes to the DMAxLAR Register with the address of the first descriptor. After the DMAxLAR is written, the DMA reads the first descriptor into the DMA control and address registers, with the exception of the LAR data. It next executes the transfers as specified by the descriptor data in the DMA. When these transfers are complete, the DMA reads in the next descriptor in the list and continues executing transfers. Figure 59 displays two descriptors and two sets of destination and source buffers; it also displays how the descriptors are loaded into the DMA and executed.

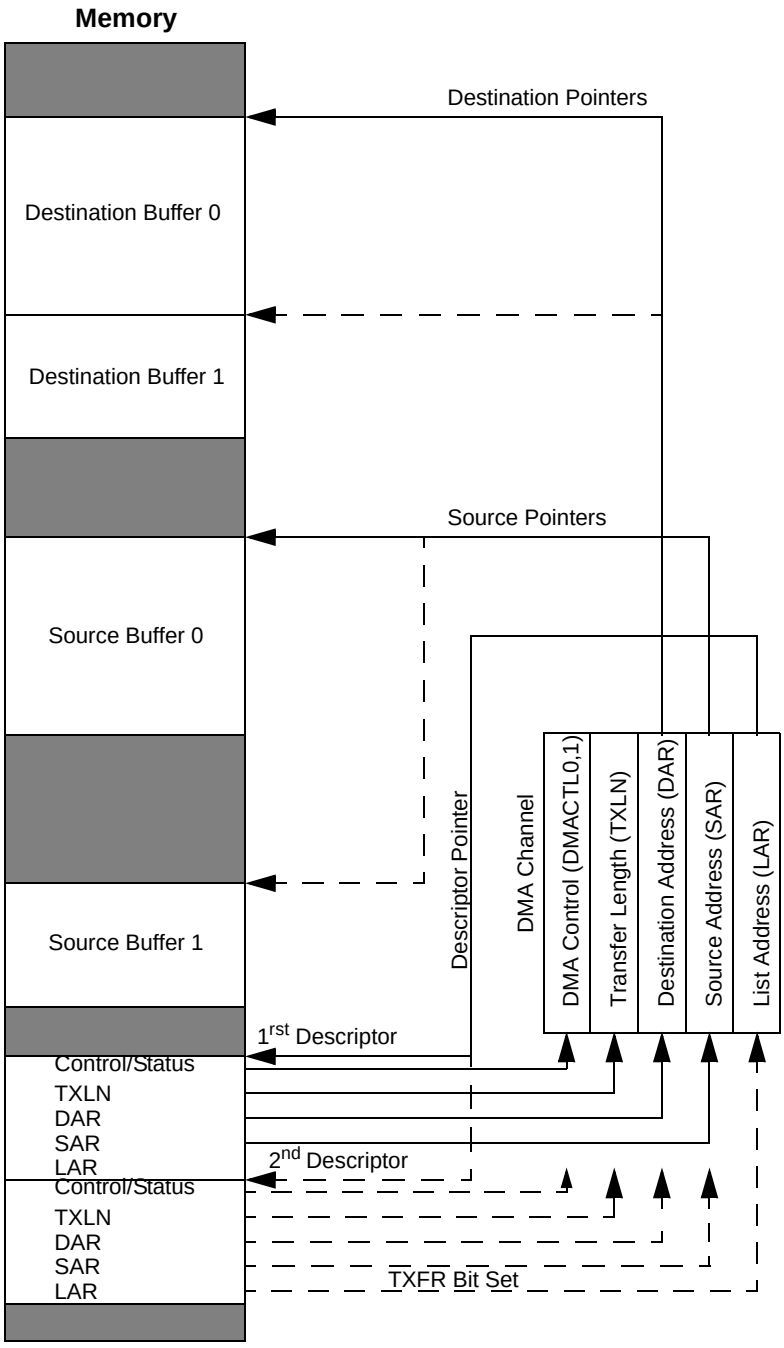


Figure 59. Linked List Diagram

Linked List Setup and Operation

The software initially must create the descriptor lists and allocate the buffers for each list. In addition, software must perform the following operations.

1. Write the DAMxREQSEL to select the appropriate request source.
2. Set the CONTROL field in the descriptor (not the DMA) for the appropriate operation:
   a. DMAxEN: set to 1.
   b. LOOP: set to 1 to not have the descriptor modified.
   c. TXSIZE: set the appropriate size for byte, word or quad.
   d. DSTCTL: set to increment, decrement, or fixed.
   e. SRCCTL: set to increment, decrement, or fixed.
   f. IEOB: set to 1 if an interrupt must be generated when this descriptor is closed.
   g. TXFR: set this bit if the LAR is used to point to the next descriptor.
   h. EOF: if an end-of-frame buffer, then set this bit.
   i. HALT: if the DMA must stop at the end of this buffer, then set this bit to 1.
   j. CMDSTAT: set this field with a command for the selected peripheral.
3. Write the destination address to the destination field.
4. Write the source address to the source field.
5. Write the transfer length for this buffer.
6. If this descriptor has its TXFR bit set then the LAR address to point to the next descriptor.
7. If there are additional descriptors in the list then set them up using the same procedure listed above.

After the descriptor has been set up, the software must write the DMAxLAR in the appropriate DMA with the address of the descriptor. The DMA performs the following operations:

1. Generate a request to the CPU.
2. Place the DMAxLAR address on the bus and fetch the CONTROL word from the descriptor. This word is then placed in the DMAxCTL Register of the DMA Channel.
3. Fetch the Destination address from the descriptor and place it in the DMAxDAR Register in the DMA Channel.
4. Fetch the Source address from the descriptor and place it in the DMAxSAR Register in the DMA Channel.

5. Fetch the TXLN length from the descriptor and place it in the DMAxTXLN Register in the DMA Channel.
6. After the reads have been completed, the DMA starts looking for requests and transfer data until the transfer length reaches zero or the DMA receives a Request EOF signal.
7. When the DMA receives the Request EOF signal, it performs the following operations based upon the LOOP and EOF bit:
   - 00: The DMA writes the descriptor Control/Status word with the DMAxEN bit reset to 0.
   - 01: The DMA requests status from the peripheral. It then writes the descriptor Control/Status word with the DMAxEN bit reset to 0 and the status returned from the peripheral. The DMA then writes the TXLN length to the descriptor.
   - 10, 11: The DMA does not modify the descriptor.
8. If the HALT bit is set the DMA closes the current buffer but does not fetch the next descriptor.
9. After a new DMAxLAR address has been updated, the DMA returns to Step 2 and fetches the control/status byte.

DMA Priority

The DMA priority is based upon the last channel serviced. After a channel is serviced it becomes the lowest priority channel. Table 146 lists the DMA priority; each DMA has equal priority under this scheme.

Table 146. DMA Priority

Last Channel Serviced	DMA Priority
DMA0	DMA1 (Highest) DMA2 DMA3 DMA0 (Lowest)
DMA1	DMA2 (Highest) DMA3 DMA0 DMA1 (Lowest)
DMA2	DMA3 (Highest) DMA0 DMA1 DMA2 (Lowest)
DMA3	DMA 0 (Highest) DMA 1 DMA 2 DMA 3 (Lowest)

DMA Bandwidth Selection

In the CPUCTL Register, the DMABW Mode bits set the maximum bus bandwidth the DMA is allowed. There are four modes (For more details, refer to the ZNEO CPU Core User Manual (UM0188), available for download at www.zilog.com.

Table 147 lists the DMA bandwidth selection.

Table 147. DMA Bandwidth Selection

Bits	Description
00	DMA uses 100% of the bandwidth.
01	DMA is allowed one transfer for each CPU operation.
10	DMA is allowed one transfer for every two CPU operations.
11	DMA is allowed one transfer for every three CPU operations.

DMA Interrupts

Each DMA has its own interrupt vector. Interrupts occur on the following conditions:

- Whenever a buffer is completed which has its IEOB set

- When the upper eight bits of the transfer length equal zero and the lower eight bits of the transfer length is equal to the DMAxLAR[23:16] and the DMA is in DIRECT Mode
- If a buffer has been terminated by a Request EOF

For additional information about interrupts, see the Interrupt Controller chapter on page 80.

DMA Request Select Register

The DMA Request Select Register, shown in Table 148, governs the state of the DMA Channel.

Table 148. DMA Select Register (DAMxREQSEL)

Bits	7	6	5	4	3	2	1	0
Field	CHANSTATE				REQSEL			
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Addr	FFE400h, FFE401h, FFE402h, FFE403h							

Bit	Description
[7:4] CHANSTATE	**Channel State** 0000 = DMA Off 0001 = DIRECT Mode, Waiting for End of Frame signal 0010 = LINKED LIST Mode, Waiting for End of Frame signal 0011 = Reserved 0100 = DIRECT Mode, First byte transfer, send command 0101 = LINKED LIST Mode, First byte transfer, send command 0110 = DIRECT Mode, Transfer of buffer in progress 0111 = LINKED LIST Mode, Transfer of buffer in progress 1000 = DIRECT Mode, Close Descriptor 1001 = LINKED LIST Mode, New List 1010 = LINKED LIST Mode, Close Descriptor 1011-1111 = Reserved

Bit	Description (Continued)
[3:0] REQSEL	**DMA Request Selection by Channel**
DMA0	**DMA 0 Request Select** 0000 = Continuous (i.e., Memory to Memory) 0001 = Timer 0 0010 = Timer 1 0011 = Timer 2 0100 = UART0 RXD 0101 = UART0 TXD 0110 = UART1 RXD 0111 = UART1 TXD 1000 = I2C RX 1001 = I2C TX 1010 = SPI RX 1011 = SPI TX 1100 = ADC0 1101 = Reserved 1110 = Reserved 1111 = DMA0REQ Pin
DMA1	**DMA 1 Request Select** 0000 = Continuous (i.e., Memory to Memory) 0001 = Timer 0 0010 = Timer 1 0011 = Timer 2 0100 = UART0 RXD 0101 = UART0 TXD 0110 = UART1 RXD 0111 = UART1 TXD 1000 = I2C RX 1001 = I2C TX 1010 = SPI RX 1011 = SPI TX 1100 = ADC0 1101 = Reserved 1110 = Reserved 1111 = DMA1REQ Pin

Bit	Description (Continued)
DMA2	**DMA 2 Request Select** 0000 = Continuous (i.e., Memory to Memory) 0001 = Timer 0 0010 = Timer 1 0011 = Timer 2 0100 = UART0 RXD 0101 = UART0 TXD 0110 = UART1 RXD 0111 = UART1 TXD 1000 = I2C RX 1001 = I2C TX 1010 = SPI RX 1011 = SPI TX 1100 = ADC0 1101 = Reserved 1110 = Reserved 1111 = DMA2REQ Pin
DMA3	**DMA 3 Request Select** 0000 = Continuous (i.e., Memory to Memory) 0001 = Timer 0 0010 = Timer 1 0011 = Timer 2 0100 = UART0 RXD 0101 = UART0 TXD 0110 = UART1 RXD 0111 = UART1 TXD 1000 = I2C RX 1001 = I2C TX 1010 = SPI RX 1011 = SPI TX 1100 = ADC0 1101 = Reserved 1110 = Reserved 1111 = Reserved

DMA Control Registers

The following section describes the DMA control registers.

DMA Control Register

The DMA Control Register, shown in Table 149, enables and control the DMA transfer.

Table 149. DMA Control Register A (DMAxCTL)

Bits	15	14	13	12	11	10	9	8
Field	DMAxEN	LOOP	TXSIZE		DSTCTL		SRCCTL	
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE410h, FFE420h, FFE430h, FFE440h							
Bits	7	6	5	4	3	2	1	0
Field	IEOB	TXFR	EOF	HALT	CMDSTAT			
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE411h, FFE421h, FFE431h, FFE441h							

Bit	Description
[15] DMAxEN	**DMA X Enable** If this bit is written directly, then Normal Mode is executed. If this bit is read in from a descriptor, then LINKED LIST Mode is executed. 0 = DMA is disabled. 1 = DMA is enabled.
[14] LOOP	**LOOP Mode** 0 = Descriptor is modified when the buffer is closed. 1 = Descriptor is not modified when buffer is closed.
[13:12] TXSIZE	**Transfer Size** 00 = Byte. 01 = Word. 10 = Quad. 11 = Reserved.
[11:10] DSTCTL	**Destination Control Register** 00 = Destination address does not change. 01 = Destination address increments. 10 = Destination address decrements. 11 = Reserved.

Bit	Description (Continued)
[9:8] SRCCTL	**Source Control Register** 00 = Source address does not change. 01 = Source address increments. 10 = Source address decrements. 11 = Reserved.
[7] IEOB	**Interrupt On End Of Buffer** 0 = Do not generate an interrupt when the DMA completes this buffer. 1 = Generate interrupt at the end of this buffer.
[6] TXFR	**Transfer To New List Address** This bit is used only in LINKED LIST Mode. 0 = Increment DMAxLAR by 16 at the end of this buffer. 1 = Load the DMAxLAR with the new List Address value from the descriptor.
[5] EOF	**End Of Frame** 0 = Not a End Of Frame buffer. 1 = This buffer is the end of the current frame.
[4] HALT	**Halt After This Buffer** This bit is used only in LINKED LIST Mode. 0 = Next descriptor is loaded. 1 = The DMA will halt at the end of this buffer.
[3:0] CMDSTAT	**Command Status Field** On the first transfer of a buffer, this field is placed on the CMDBUS and the CMDVALID is asserted. If the EOF bit is set, the DMA requests a status from the peripheral and places it in this field. In LINKED LIST Mode, this field is written back to the descriptor. The DMA does not use this field; it simply passes it on. The definitions of these bits are specified in each peripheral.

DMA X Transfer Length Register

The DMA X Transfer Length High and Low registers, shown in Tables 150 and 151, form a 16-bit transfer length. Each of these registers is decremented each time a DMA transfer occurs.

Table 150. DMA X Transfer Length High Register (DMAxTXLNH)

Bits	7	6	5	4	3	2	1	0
Field	DMAxTXLNH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE412h, FFE422h, FFE432h, FFE442h							

Table 151. DMA X Transfer Length Low Register (DMAxTXLNL)

Bits	7	6	5	4	3	2	1	0
Field	DMAxTXLNL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE413h, FFE423h, FFE433h, FFE443h							

DMA Destination Address

The DMA X Destination Address Register Upper, High and Low registers form the destination address. This address points to the location in which the data from the transfer will be stored.

Table 152. DMA X Destination Address Register Upper (DMAxDARU)

Bits	7	6	5	4	3	2	1	0
Field	DMAxDARU							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE415h, FFE425h,FFE435h,FFE445							

Table 153. DMA X Destination Address Register High (DMAxDARH)

Bits	7	6	5	4	3	2	1	0
Field	DMAxDARH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE416h, FFE426h, FFE436h, FFE446h							

Table 154. DMA X Destination Address Register Low (DMAxDARL)

Bits	7	6	5	4	3	2	1	0
Field	DMAxDARL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE417h, FFE427h, FFE437h, FFE447h							

DMA Source Address Registers

The DMA X Source Address Register Upper, High and Low registers form a 24-bit source address. This address is used to point to the source data for the transfer.

Table 155. DMA X Source Address Register Upper DMAxSARU

Bits	7	6	5	4	3	2	1	0
Field	DMAxSARU							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE419h, FFE429h, FFE439h, FFE449h							

Table 156. DMA X Source Address Register High (DMAxSARH)

Bits	7	6	5	4	3	2	1	0
Field	DMAxSARH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE41Ah, FFE42Ah, FFE43Ah, FFE44Ah							

Table 157. DMA X Source Address Register Low (DMAxSARL)

Bits	7	6	5	4	3	2	1	0
Field	DMAxSARL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE41Bh, FFE42Bh, FFE43Bh, FFE44Bh							

DMA List Address Register

The DMA List Address Register is written when the list mode for the DMA is used. This register contains the address of the current list the DMA is operating on. Writing the DMAxLARL Register (shown in Table 160) enables the DMA for list operation.

Table 158. DMA X List Address Register Upper DMAxLARU

Bits	7	6	5	4	3	2	1	0
Field	DMAxLARU							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE41Dh, FFE42Dh, FFE43Dh, FFE44Dh							

In DIRECT Mode, this register is used to set a watermark interrupt. This interrupt occurs when the DMATXLN[15:8] equals 0 and DMAxTXLN[7:0] equals DMAxLARU. Note when using the watermark the DMAxLARL must not be written.

Table 159. DMA X List Address Register High (DMAxLARH)

Bits	7	6	5	4	3	2	1	0
Field	DMAxLARH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE41Eh, FFE42Eh, FFE43Eh, FFE44Eh							

Table 160. DMA X List Address Register Low (DMAxLARL)

Bits	7	6	5	4	3	2	1	0
Field	DMAxLARL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFE41Fh, FFE42Fh, FFE43Fh, FFE44Fh							

Writing to the DMAxLARL Register causes the DMA to enter LINKED LIST Mode.

External DMA Signals

Two external pins are associated with each DMA Channel capable of external transfers (Channel 3 does not have external DMA capability). They are active Low DMAxREQ and DMAxACK signals. DMAxACK signals are outputs and DMAxREQ are inputs. DMAxREQ must be asserted for a minimum of one system clock period to generate one DMA transfer. DMAxREQ is left asserted for multiple transactions and deasserted after DMAxACK asserts for the last appropriate transfer.

DMA Timing

External DMA Transfer

Figure 60 shows the read and write timing of external DMA transfers.

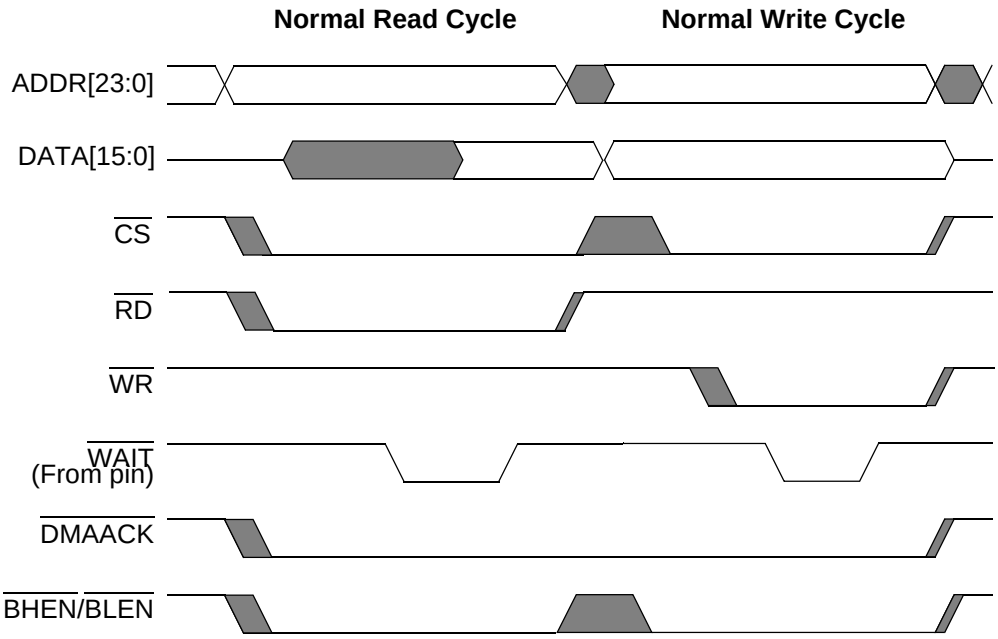


Figure 60. External DMA Transfer

External ISA DMA Transfer

Figure 61 shows the read and write timing of external ISA DMA transfers.

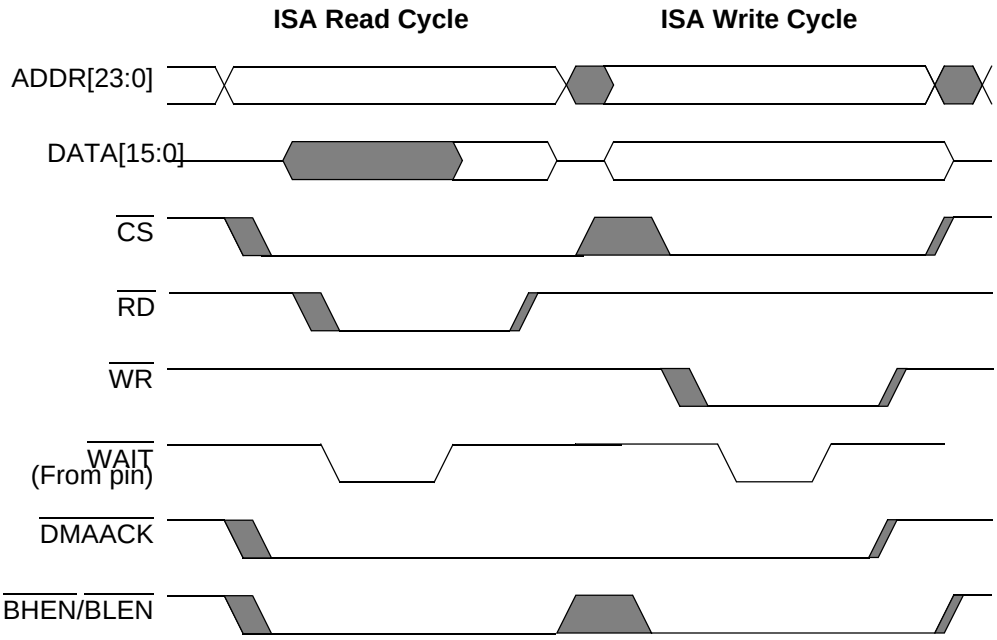


Figure 61. External ISA DMA transfer

Option Bits

Option bits allow user configuration of certain aspects of the ZNEO® Z16F Series operation. The feature configuration data is stored in the Program memory and read during Reset. The features available for control using the option bits are:

- WDT time-out response selection–interrupt or Reset
- WDT enabled at Reset
- The ability to prevent unwanted read access to user code in Program memory
- The ability to prevent accidental programming and erasure of the user code in Program memory
- Voltage Brown-Out (VBO) configuration—always enabled or disabled during Stop Mode to reduce Stop Mode power consumption
- Oscillator mode selection for high, medium and low power crystal oscillators, or external RC oscillator
- PWM pin setup for motor control application

Operation

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, Short Reset, or Stop Mode Recovery), the option bits are automatically read from the Program memory and written to Option Configuration registers. The Option Configuration registers control operation of the device. Option Bit Control Register are loaded before the device exits Reset and the ZNEO 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 four bytes of Program Memory at addresses 0000h through 0003h, shown in Tables 161 and 162, respectively, are reserved for the user option bits. These bytes are used to configure user specific options. You can change the option bits to meet application requirements.

Program Memory Address 0000h

Option bits in this space are altered to change the chip configuration at reset.

Table 161. Option Bits At Program Memory Address 0000h

Bits	7	6	5	4	3	2	1	0
Field	OSC_SEL[1:0]		WDT_RES	WDT_AO	VBO_AO	DBGUART	FWP	RP
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	Program Memory 0000h							

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

Bit	Description
[7:6] OSC_SEL[1:0]	**Oscillator Mode Selection** 00 = On-chip oscillator configured for use with external RC networks (<4MHz). 01 = Minimum power for use with very low frequency crystals (32kHz to 1.0MHz). 10 = Medium power for use with medium frequency crystals or ceramic resonators (0.5MHz to 10.0MHz). 11 = Maximum power for use with high frequency crystals (8.0MHz to 20.0MHz). This setting is the default for unprogrammed (erased) Flash.
[5] WDT_RES	**WDT Reset** 0 = WDT time-out generates an interrupt request. Interrupts must be globally enabled for the ZNEO CPU to acknowledge the interrupt request. 1 = WDT time-out causes a Short Reset. This setting is the default for unprogrammed (erased) Flash.
[4] WDT_AO	**WDT Always On** 0 = WDT is automatically enabled after reset. The WDT oscillator is disabled by clearing the WDTEN bit in the OSCCTL Register. 1 = WDT is enabled upon execution of the WDT instruction. The WDT oscillator is disabled by clearing the WDTEN bit in the OSCCTL Register.
[3] VBO_AO	**Voltage Brown-Out Protection Always On** 0 = Voltage Brown-Out protection is disabled in Stop Mode to reduce total power consumption. 1 = Voltage Brown-Out protection is always enabled, including during Stop Mode. This setting is the default for unprogrammed (erased) Flash.
[2] DBGUART	**Debug UART Enable** 0 = The Debug UART option is enabled. 1 = The Debug UART option is disabled.

Bit	Description (Continued)
[1] FWP	**Flash Write Protect** 0 = Programming, Page Erase and Mass Erase through user code is disabled. Flash operations are allowed through the On-Chip Debugger. 1 = Programming, Page Erase and Mass Erase are enabled for all of Flash Program Memory.
[0] RP	**Read Protect** 0 = User program code is inaccessible. Limited control features are available through the OCD. 1 = User program code is accessible. All OCD commands are enabled. This setting is the default for unprogrammed (erased) Flash.

Program Memory Address 0001h

Option bits in this space are altered to change the chip configuration at reset.

Table 162. Options Bits at Program Memory Address 0001h

Bits	7	6	5	4	3	2	1	0
Field	Reserved					MCEN	PWMHI	PWMLO
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	Program Memory 0001h							

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

Bit	Description
[7:3]	**Reserved** These Option Bits are reserved for future use and must always be 1. This setting is the default for unprogrammed (erased) Flash.
[2] MCEN	**Motor Control Enable** 0 = Motor control pins are enabled on reset. 1 = Normal Pin operation.
[1] PWMHI	**High Side Off Initial Value** 0 = The high side off value is equal to 0. 1 = The high side off value is equal to 1.
[0] PWMLO	**Low Side Off Initial Value** 0 = The low side off value is equal to 0. 1 = The low side off value is equal to 1.

Program Memory Address 0002h

Option Bits in this space are altered to change the chip configuration at reset.

Table 163. Options Bits at Program Memory Address 0002h

Bits	7	6	5	4	3	2	1	0
Field	Reserved							
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	Program Memory 0002h							

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

Bit	Description
[7:0]	**Reserved** These option bits are reserved for future use and must always be 1. This setting is the default for unprogrammed (erased) Flash.

Program Memory Address 0003h

Option bits in this space are altered to change the chip configuration at reset.

Table 164. Options Bits at Program Memory Address 0003h

Bits	7	6	5	4	3	2	1	0
Field	ROMLESS 16	LPOPT	Reserved					
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	Program Memory 0003h							

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

Bit	Description
[7] ROMLESS 16	**ROMLESS 16 Select** 0 = If the device is ROMLESS, the data bus is 8 bits wide and is on Port E[7:0]. 1 = If the device is ROMLESS, the data bus is 16 bits wide and is on {Port J[7:0], Port E[7:0]}

Bit	Description (Continued)
[6] LPOPT	**Low Power Option** 0 = The part will come up in low power mode. The Clock is divided by 8 and Flash memory will only be accessed the last half of the last cycle of the divide. This reduces Flash power consumption. 1 = The part will come up normally.
[5:0]	**Reserved** These option bits are reserved for future use and must always be 1. This setting is the default for unprogrammed (erased) Flash.

Information Area

Data in the information area of memory cannot be altered directly. If you wish to alter the factory settings, it must be done by writing to the Register Address identified. The part defaults to the factory settings after reset and the registers must be rewritten to have the user settings in effect. Read the information area address to determine the factory settings.

IPO Trim Registers (Information Area Address 0021h and 0022h)

Tables 165 and 166 define the IPO Trim settings, which are altered after reset by accessing the IPOTRIM1 and IPOTRIM2 registers.

Table 165. IPO Trim 1 (IPOTRIM1)

Bits	7	6	5	4	3	2	1	0
Field	IPO TEMP TRIM						IPO TRIM	
RESET	L	L	L	L	L	L	L	L
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFFF_FF25h							
Note: L = Loaded at Reset. R/W = Read/Write. This register is loaded from Information area on Reset.								

Table 166. IPO Trim 2 (IPOTRIM2)

Bits	7	6	5	4	3	2	1	0
Field	IPO TRIM							
RESET	L	L	L	L	L	L	L	L
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFFF_FF26							
Note: L = Loaded at Reset. R/W = Read/Write. This register is loaded from Information area on Reset.								

ADC Reference Voltage Trim (Information Area Address 0023h)

Table 167 defines the ADC Reference Voltage Trim settings, which are altered after reset by accessing the ADCTRIM Register.

Table 167. ADC Reference Voltage Trim (ADCTRIM)

Bits	7	6	5	4	3	2	1	0
Field	Reserved			ADCREF[4:0]				
RESET	L	L	L	L	L	L	L	L
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FFFF_FF27							

Note: L = Loaded at Reset. R/W = Read/Write. This register is loaded from Information area on Reset.

Bit	Description
[7:5]	**Reserved** These bits are reserved and must be programmed to 1.
[4:0] ADCREF[4:0]	**ADC Reference Trim** These bits are used to trim the ADC reference voltage generator. If the part is not going to be trimmed, the value of this register must be F0h.

On-Chip Debugger

The ZNEO® Z16F Series products have an integrated On-Chip Debugger (OCD) that provides the following features:

- Reading and writing memory
- Reading and writing CPU registers
- Execution of CPU instructions
- In-circuit programming and erasing of the Flash
- Unlimited number of software breakpoints
- Four hardware breakpoints
- Instruction execution trace
- Single-pin serial communication interface

Architecture

The OCD consists of two main blocks: the transmitter/receiver unit and the debug control logic. Figure 62 displays the architecture of the OCD.

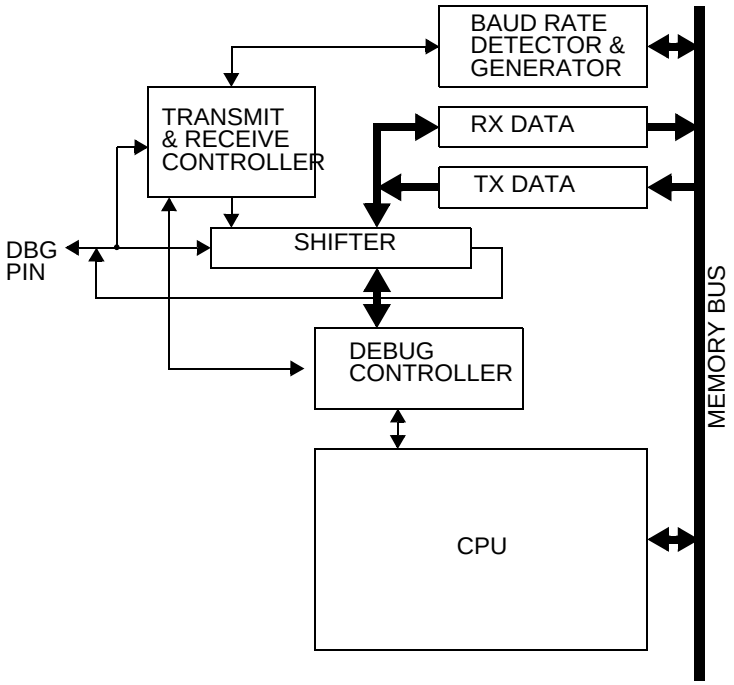


Figure 62. On-Chip Debugger Block Diagram

Operation

Caution: For effective operation of the device, all power pins (V_{DD} and AV_{DD}) must be supplied with power and all ground pins (V_{SS} and AV_{SS}) must be properly grounded. The DBG pin must be connected to V_{DD} through an external pull-up resistor to ensure proper operation.

On-Chip Debug Enable

The DBG pin is mainly used for debugging. The OCD is always enabled by default following reset. Disable the OCD after startup and use the DBG pin as a UART or a GPIO pin if the DBGUART option bit has been cleared.

To use the DBG pin as a UART or GPIO pin, the OCD must be disabled. The OCD is disabled by clearing the OCDEN bit in the Debug Control Register (DBGCTL). The OCD cannot be disabled if the OCDLOCK bit in the DBGCTL Register is set.

Serial Interface

The DBG pin is used for serial communication. This one-pin interface is a bidirectional half-duplex open-drain interface that transmits and receives data. Transmit and receive operations cannot occur simultaneously. The serial data is sent and received using the asynchronous protocol defined in RS-232. The serial pin is connected to the serial port of the PC using minimal external hardware. Two different methods for connecting the serial pin to an RS-232 interface are depicted in Figures 63 and 64.

The serial pin is open-drain and must be connected to V_{DD} through an external pull-up resistor to ensure proper operation.

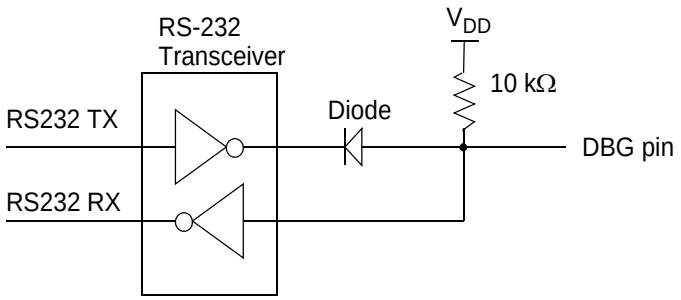


Figure 63. Interfacing a Serial Pin with an RS-232 Interface, #1 of 2

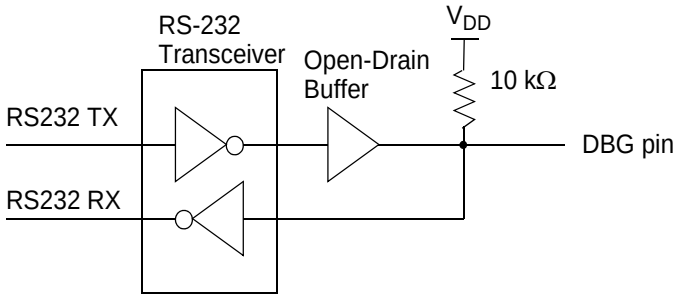
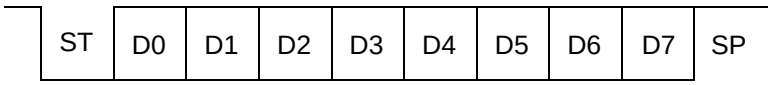


Figure 64. Interfacing a Serial Pin with an RS-232 Interface, #2 of 2

Serial Data Format

The data format of the serial interface uses the asynchronous protocol defined in RS-232. Each character is transmitted as one start bit, eight to nine data bits (least-significant bit first) and one stop bit; see Figure 65.



ST = Start Bit
SP = Stop Bit
D0–D7 = Data Bits

Figure 65. OCD Serial Data Format

Each bit time is of same length. The bit period is set by the baud rate generator.

When the transmitter sends a character, it first sends a Low start bit. The transmitter then waits one bit time. After the start bit is sent, the transmitter sends the next data bit. The transmitter sends each data bit in turn, waiting one full bit time before sending the next data bit. After the last data bit is sent, the transmitter sends a high stop bit for one bit time.

The receiver looks for the falling edge of the start bit. After the receiver sees the start bit is Low, it waits one half bit time and samples the middle of the start bit. If the middle of the start bit is High, the receiver considers this as a false start bit. The receiver ignores a false start bit and searches for another falling edge. If the middle of the start bit is Low, the receiver considers the start bit valid. The receiver will wait a full bit time from the middle of the start bit to sample the next data bit. The next data bit is sampled in the middle of the bit period. The receiver repeats this operation for each data bit, waiting one full bit time to between sampling each data bit.

After the receiver has sampled the last data bit, it waits one full bit time and sample the middle of the stop bit. If the stop bit is Low, the receiver detects a framing error.

If the stop bit is High, the data was correctly framed between a start and stop bit. After the receiver samples the middle of the stop bit, it begins searching for another start bit. To correct for any bit skew due to error between the transmit and receive baud rate clocks, the receiver does not wait for the full stop bit to be received before searching for the next start bit.

Baud Rate Generator

The baud rate generator (BRG) is used to generate a bit clock for transmit and receive operations. The BRG reload register is automatically configured by the auto-baud detector, or it is written by software.

The value in the BRG reload register is calculated as:

$$\text{BAUD RELOAD VALUE} = \frac{\text{SYSTEM CLOCK}}{\text{BAUD RATE}} \times 8$$

This reload value is the number of system clocks used to transmit and receive eight data bits.

The BRG has a 16-bit reload counter and is clocked by the system clock. When the OCD is enabled, this register is limited to 12 bits. The minimum baud rate is calculated using the following equation:

$$\text{BAUD RELOAD VALUE} = \frac{\text{SYSTEM CLOCK}}{\text{BAUD RATE}} \times 8$$

The minimum baud rate when the OCD is enabled is the system clock frequency divided by 512. The minimum baud rate is the system clock frequency divided by 8192 when the OCD is disabled.

For asynchronous operation, the maximum baud rate is roughly the system clock frequency divided by eight (eight clocks per bit). With slow baud rates and clean signals, you will be able to achieve asynchronous baud rates up to 4 clocks per bit. If data is synchronized with the system clock, the maximum baud rate is the system clock frequency (one bit per clock). The maximum baud rates are limited by the rise and fall times due to the cable impedance. Table 168 lists minimum and maximum baud rates for sample crystal frequencies.

Table 168. OCD Baud Rate Limits

System Clock Frequency	Maximum Baud Rate	Minimum Baud Rate (OCDEN=0)	Minimum Baud Rate (OCDEN=1)
20.0MHz	2.5M baud*	2442 baud	39,062 baud
1.0MHz	125K baud	123 baud	1953 baud
32.768kHz	4096 baud	4.0 baud	64 baud

Note: *The maximum baud rate is limited by the rise and fall times due to the cable impedance.

Auto-Baud Detector

To operate using various clock frequencies over a range of baud rates, the serial interface has an auto-baud detector. The auto-baud detector is used to automatically set up the baud rate generator.

The auto-baud detector is set up to measure one of two different auto-baud characters, 80h (default) or 0Dh. The default auto-baud character 80h is compatible with previous Z8 Encore!® debug interfaces. When the OCD is disabled and the DBG pin is being used as a UART, you can switch to an auto-baud character of 0Dh. The 0Dh character is the ASCII carriage return character and is sent using a terminal interface.

When using the auto-baud character 80h, the auto-baud detector measures the period from the falling edge at the beginning of the start bit to the rising edge at the beginning of data

bit 7. For the auto-baud character 0Dh, the auto-baud detector measures the period from the rising edge at the end of the start bit to the rising edge at the beginning of the stop bit. This measured value is automatically written to the BRG reload register after the auto-baud character is received. Once configured, the BRG will generate a bit clock based on this measured character time.

Line Control

When operating at high speeds, it is appropriate to speed up the rise and fall times of the single wire bus. Three control bits are used to control the bus rise and fall times, the high drive strength enable bit, the drive high enable bit and the output enable control bit.

The high drive strength enable bit puts the pin into High Driv eMode. For information about high drive strength, see the Electrical Characteristics chapter on page 337.

If the output enable control bit is set, the line is driven High and Low during transmission. If the drive high control bit is set, it drives the line high for short periods when transmitting a logic one. This rapidly charges the inherent capacitance of the single wire bus.

If both the output enable and drive high control bits are set, the line is driven high for one clock cycle when transmitting a one. If the output enable bit is clear and the drive high bit is set, the line is driven high until the input is detected High or the center of the bit time occurs, whichever is first.

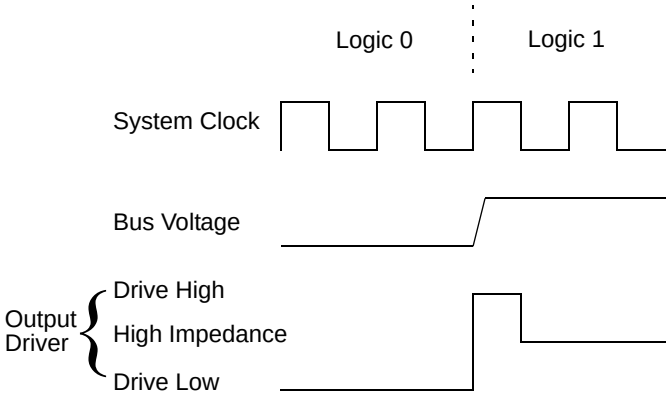


Figure 66. Output Driver when Drive High and Open Drain Enabled

9-Bit Mode

The serial interface is configured to transmit and receive a ninth data bit. This ninth bit is used to transmit or receive a software generated parity bit. It is used as an address/data bit in a multi-node system such as RS-485.

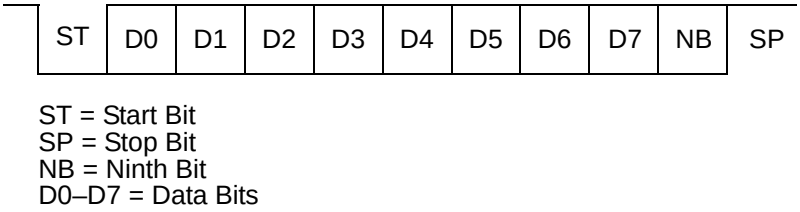


Figure 67. 9-Bit Mode

Start Bit Flow Control

If flow control is required, start bit flow control is used. Start bit flow control requires the receiving device send the start bit. The transmitter waits for the start bit, then transmit its data following the start bit.

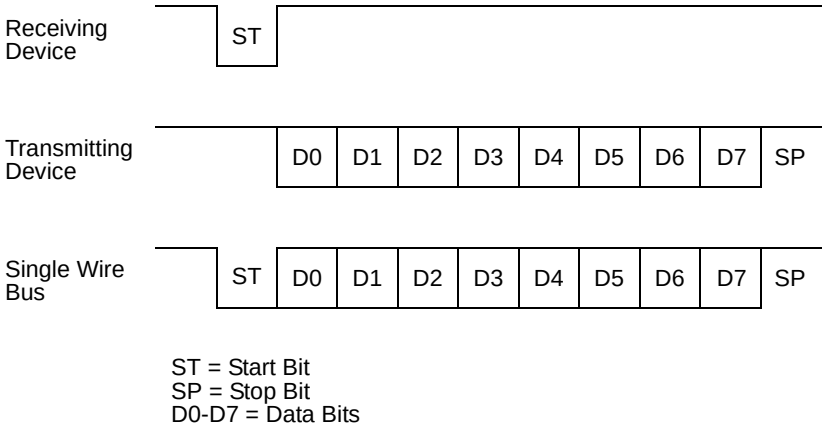


Figure 68. Start Bit Flow Control

If the standard serial port of a PC is used, transmit flow control is enabled on the ZNEO Z16F Series device. The PC sends the start bit when receiving data by transmitting the character FFh. Because the FFh character is also received from a nonresponsive device, space parity (parity bit always zero) must be enabled and used as an acknowledge bit.

Initialization

The OCD ignores any data received until it receives the read revision command 00h. After the read revision command is received, the remaining debug commands are issued. The packet CRC is not sent for the first read revision command issued during initialization.

Initialization during Reset

The OCD is initialized during reset by asserting the reset pin, sending the auto-baud character, and then issuing the read revision command. When the OCD is initialized during reset, the DBGHALT bit in the OCDCTL Register is automatically set.

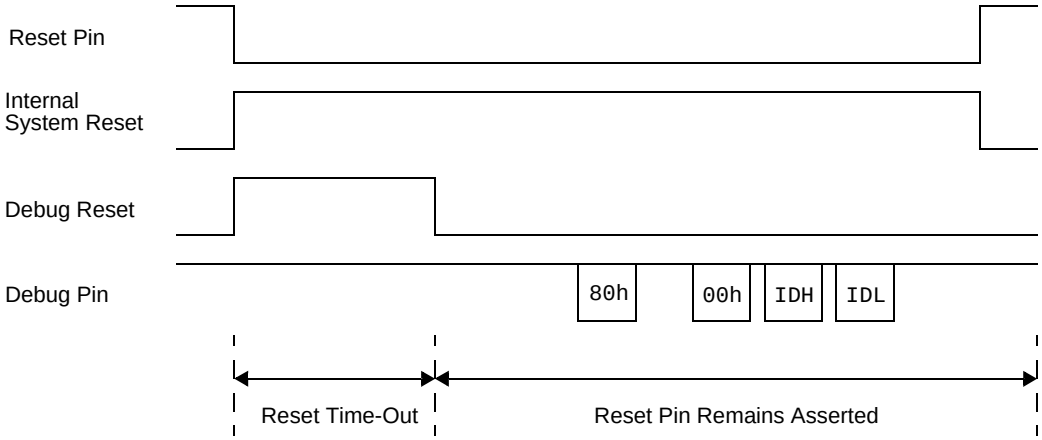


Figure 69. Initialization During Reset

Debug Lock

The interface has a locking mechanism to prevent user code from disabling the OCD and using the DBG pin as a UART or GPIO pin. The DBGLOCK bit in the DBGCTL Register prevents you from disabling the OCD and modifying any register that would inhibit communication with the OCD. The default state of the DBGLOCK bit is set accordingly to the DBGUART option bit.

In order to use the DBG pin as a UART or GPIO pin, you must program the DBGUART option bit to 0 so the OCDLOCK control bit is cleared after reset. After the control register is unlocked, software then clears the OCDEN control bit to use the DBG pin as aUART or GPIO pin.

If the DBGUART option bit is cleared and the OCDLOCK control bit is not set, the OCD is still locked before code has the chance to disable the OCD by initializing the Debugger during reset and writing the OCDLOCK control bit to 1.

Error Reset

The serial interface has an Auto-Reset mechanism that resets the serial interface when a Transmit Collision or Receive Framing Error is detected. When a Transmit Collision or Receive Framing Error is detected when OCDEN is set, the OCD aborts any command

currently in progress, transmits a Serial Break condition for 4096 system clocks and sets the ABSRCH bit in the DBGCTL Register. This break is sent to ensure the host also detects the error.

A clock change invalidates the baud reload value. Communication cannot continue until a new autobaud reload value is set. As a result, the device automatically sends a serial break to reset the communication link whenever a clock change occurs.

DEBUG Halt Mode

During debugging, it is appropriate to stop the CPU from executing instructions by placing the device in DEBUG Halt Mode. The operating characteristics of the ZNEO Z16F Series devices in DEBUG Halt Mode are:

- The ZNEO CPU fetch unit stops, idling the ZNEO CPU
- All enabled on-chip peripherals operate unless in Stop Mode
- Constantly refreshes the WDT, if enabled

Entering DEBUG Halt Mode

The device enters DEBUG Halt Mode by any of the following operations:

- Write the DBGHALT bit in the DBGCTL Register to 1 using the OCD interface
- ZNEO CPU execution of BRK instruction (when enabled)
- Hardware breakpoint match.

Exiting DEBUG Halt Mode

The device exits DEBUG Halt Mode by any of the following operations:

- Clearing the DBGHALT bit in the DBGCTL Register to 0
- Power-On Reset
- Voltage Brown-Out reset
- Asserting the $\overline{\text{RESET}}$ pin Low to initiate a Reset

Reading and Writing Memory

Most debugging functions are accomplished by reading and writing control registers. The OCD hardware has the capability of reading and writing memory when the CPU is running.

When a read or write request from the OCD hardware occurs, the OCD steals the bus for the number of cycles required to complete the read or write operation. This bus stealing

occurs on a per byte basis, not a per command basis. Because the debugger operates serially, it takes several clock cycles to transmit or receive a character.

If the debugger receives a command to read or write a block of memory, it will not steal the bus for the entire read or write command. The debugger will only steal the bus for a short period of time for each data byte. A debug write cycle will occur after a byte has been received during a write operation. A debug read cycle will occur when the transmitter is empty during a read operation.

Data read from or written to the OCD occurs one byte at a time. Therefore, memory read and write operations occur one byte at a time. Operations that occur on multi-byte words does not occur concurrently.

Reading Memory CRC

Because the ZNEO device has such a large memory space and the debug interface is serial, reading massive amounts of data during debugging is time consuming. The OCD hardware has the capability of calculating a cyclic redundancy check (CRC) on memory to allow memory caching mechanisms to be used by the host debugging software. This CRC verifies that the contents of a memory cache has not changed.

When the read CRC command is issued, the OCD hardware steals the CPU bus during the entire read operation. The length of time it takes to generate the CRC is equal to the amount of time it takes to read the memory used in the CRC calculation.

The OCD hardware also features the capability of returning separate CRCs for each 4K block of memory. Software determines these portions of memory, which are modified when the cache for a large block of memory is invalidated.

Breakpoints

Software Breakpoints

Breakpoints are generated when the CPU executes the BRK instruction and breakpoints are enabled. If breakpoints are not enabled, the BRK instruction will vector to the system exception vector and set the illegal instruction status bit.

If a Breakpoint is generated, the OCD is configured to automatically enter DEBUG Halt Mode or to just loop on the instruction. If the OCD is configured to loop on the instruction, the CPU is still able to service DMA and interrupt requests in the background. Software polls the DBGBRK bit of the DBGCTL Register to determine if the OCD has reached a Breakpoint.

Hardware Breakpoint

There are four hardware breakpoints on the ZNEO Device. When enabled, a breakpoint is generated when the program counter matches the value in the breakpoint register, or when

a memory access occurs at the address in the breakpoint register. A data watchpoint watches a range of addresses by selecting how many lower address bits are ignored.

Instruction Trace

Trace Overview

The ZNEO has the ability to trace the instruction flow. If enabled, it uses existing Memory to store the Program Counter data each time a change in execution flow occurs. This requires you to allocate memory space to hold the trace information.

Trace Events

A trace event occurs anytime a CALL, RET, Interrupt, IRET, TRAP, JP, DJNZ, or Exception occurs. Trace takes four cycles each time a trace event occurs (five cycles for IRQ, TRAP and Exceptions).

Trace Buffer

The Trace Buffer is controlled by two registers: Trace Control (TRACECTL) and Trace Address (TRACEADDR) register. The TRACECTL register is used to enable the trace and select the size of the Trace Buffer. TRACEADDR selects the starting address for the trace. The trace address is modulo-n based upon the size of the TRACESEL field in the TRACECTL Register. The modulo-n is zero aligned, which means that the trace buffer always wraps to 0 for the selected size. For example, if the TRACEADDR is set to `FFFFB050h` and the TRACECTL is set to `81h`, the Buffer is located from `FFFFB000h` to `FFFFB0FFh` with the first trace event to be written to `FFFFB050h`. When the address reaches `FFFFB0FFh` it will roll over to `FFFFB000h`.

Trace buffer sizes are 128, 256, 512, 1024, 2048, 4096, 8192 and 16384 bytes. Each trace event requires eight bytes giving a minimum of 16 events to a maximum of 2048 events. Only the Program Counter values are stored. Other information has to be inferred from the source code by the trace debugger.

Trace Operation

On each trace event the current program counter is placed in memory pointed to by the TRACEADDR. TRACEADDR increments by 4 and the next state of the program counter is written to the TRACEADDR. TRACEADDR increments by 4 again. TRACEADDR always points to the next data to be written. The lower two bits of the TRACEADDR are always zero.

Extracting Trace Information

The trace information is extracted by reading the data from the selected trace memory area. The data is then interpreted by the Trace Debugger software.

On-Chip Debugger Commands

The hardware OCD supports several commands for controlling the device. In the following list of commands, data sent from the host to the OCD is identified by:

```
DBG <-- Data
```

Data sent from the OCD back to the host is identified by:

```
DBG --> Data
```

Multiple bytes transmitted are represented with double arrows, either <<or >>.

Read Revision. The Read Revision command returns the revision identifier.

```
DBG <-- 0000_0000
DBG --> RevID[15:8]
DBG --> RevID[7:0]
DBG --> CRC[0:7]
```

Read Status Register. The Read Status Register command returns the contents of the OCDSTAT Register.

```
DBG <-- 0000_0001
DBG --> status[7:0]
DBG --> CRC[0:7]
```

Read Control Register. The Read Control register command returns the contents of the OCDCTL Register.

```
DBG <-- 0000_0010
DBG --> OCDCTL[7:0]
DBG --> CRC[0:7]
```

Write Control Register. The Write Control register command writes data to the OCDCTL Register.

```
DBG <-- 0000_0011
DBG <-- OCDCTL[7:0]
DBG --> CRC[0:7]
```

Read Registers. The Read registers command returns the contents of CPU registers R15 through R0.

```
DBG <-- 0000_0100
DBG ->> regdata[31:24]
DBG ->> regdata[23:16]
DBG ->> regdata[15:8]
DBG ->> regdata[7:0]
DBG --> CRC[0:7]
```

Write Registers. The Write registers command writes data to CPU registers R15 through R0.

```
DBG <-- 0000_0101
DBG <<- regdata[31:24]
```

```
DBG <<- regdata[23:16]
DBG <<- regdata[15:8]
DBG <<- regdata[7:0]
DBG --> CRC[0:7]
```

Read PC. The Read Program Counter command returns the contents of the program counter.

```
DBG <-- 0000_0110
DBG --> 00h
DBG --> PC[23:16]
DBG --> PC[15:8]
DBG --> PC[7:0]
DBG --> CRC[0:7]
```

Write PC. The Write Program Counter command writes data to the program counter.

```
DBG <-- 0000_0111
DBG <-- 00h
DBG <-- PC[23:16]
DBG <-- PC[15:8]
DBG <-- PC[7:0]
DBG --> CRC[0:7]
```

Read Flags. The Read Flags command returns the contents of the CPU flags.

```
DBG <-- 0000_1000
DBG --> 00h
DBG --> flags[7:0]
DBG --> CRC[0:7]
```

Write Instruction. The Write Instruction command writes one word of Op Code to the CPU.

```
DBG <-- 0000_1001
DBG <-- opcode[15:8]
DBG <-- opcode[7:0]
DBG --> CRC[0:7]
```

Read Register. The Read Register command returns the contents of a single CPU register.

```
DBG <-- {0100,regno[3:0]}
DBG --> regdata[31:24]
DBG --> regdata[23:16]
DBG --> regdata[15:8]
DBG --> regdata[7:0]
DBG --> CRC[0:7]
```

Write Register. The Write Register command writes data to a single CPU register.

```
DBG <-- {0101,regno[3:0]}
DBG <-- regdata[31:24]
DBG <-- regdata[23:16]
DBG <-- regdata[15:8]
DBG <-- regdata[7:0]
DBG --> CRC[0:7]
```

Read Memory. The Read memory command reads data from memory. The memory address is sign extended.

```
DBG <-- {1000,Size[3:0]}
DBG <-- addr[15:8]
DBG <-- addr[7:0]
DBG ->> 1 to 16 bytes of data
DBG --> CRC[0:7]
```

Write Memory. The Write memory command writes data to memory. The memory address is sign extended.

```
DBG <-- {1001,size[3:0}
DBG <-- addr[15:8]
DBG <-- addr[7:0]
DBG <<- 1 to 16 bytes of data
DBG --> CRC[0:7]
```

Read Memory. The Read memory command reads data from memory**.**

```
DBG <-- {1010,size[3:0}
DBG <-- size[11:4]
DBG <-- 00h
DBG <-- addr[23:16]
DBG <-- addr[15:8]
DBG <-- addr[7:0]
DBG ->> 1 to 4096 bytes of data
DBG --> CRC[0:7]
```

Write Memory. The Write memory command writes data to memory**.**

```
DBG <-- {1011,size[3:0}
DBG <-- size[11:4]
DBG <-- 00h
DBG <-- addr[23:16]
DBG <-- addr[15:8]
DBG <-- addr[7:0]
DBG <<- 1 to 4096 bytes of data
DBG --> CRC[0:7]
```

Read Memory CRC. The Read memory CRC command computes and return the CRC of a block of memory.

```
DBG <-- {1110,BlockCount[3:0]}
DBG <-- BlockCount[11:4]
DBG <-- 00h
DBG <-- addr[23:16]
DBG <-- {addr[15:12],xxxx}
DBG --> MemoryCRC[0:7]
DBG --> MemoryCRC[8:15]
DBG --> CRC[0:7]
```

MemoryCRC is computed on memory in increments of 4K blocks. The BlockCount field determines how many blocks of memory to compute the MemoryCRC on.

Read Each Memory CRC. The Read memory CRC command computes and return the CRC of a block each 4K memory block.

```
DBG <-- {1111,BlockCount[3:0]}
DBG <-- BlockCount[11:4]
DBG <-- 00h
DBG <-- addr[23:16]
DBG <-- {addr[15:12],xxxx}
DBG ->> MemoryCRC[0:7]
DBG ->> MemoryCRC[8:15]
DBG --> CRC[0:7]
```

The MemoryCRC is computed on memory in increments of 4K blocks. The CRC is returned for each 4K block and is reset at the start of each block. The BlockCount field determines how many blocks of memory to compute the MemoryCRC on.

The On-Chip Debugger commands are summarized in Table 169.

Table 169. On-Chip Debugger Commands

Debug Command	Command Byte	Disabled by Read Protect Option Bit
Read Revision	0000–0000	—
Read OCD Status Register	0000–0001	—
Read OCD Control Register	0000–0010	—
Write OCD Control Register	0000–0011	Cannot single step (bit0 has no effect).
Read Registers (CPU registers R15-R0)	0000–0100	Yes.
Write Registers (CPU registers R15-R0)	0000–0101	Yes.
Read Program Counter	0000–0110	Yes.
Write Program Counter	0000–0111	Yes.
Read Flags	0000–1000	Yes.
Write Instruction	0000–1001	Yes.
Read Register (single CPU register)	0100–(regno[3:0])	Yes.
Write Register (single CPU register)	0100–(regno[3:0])	Yes.
Read Memory (short -address is sign extended)	0100–(regno[3:0])	Read only unprotected memory locations.

Note: Unlisted command byte values are reserved.

Table 169. On-Chip Debugger Commands (Continued)

Debug Command	Command Byte	Disabled by Read Protect Option Bit
Write Memory (short -address is sign extended)	0100–(regno[3:0])	Write only unprotected memory locations.
Read Memory (long)	1010–size[3:0]	Read only unprotected memory locations.
Write Memory (long)	1011–size[3:0]	Write only unprotected memory locations.
Read Memory CRC	1110–BlockCount[3:0]	—
Read Each Memory CRC	1111–BlockCount[3:0]	—

Note: Unlisted command byte values are reserved.

Cyclic Redundancy Check

To ensure transmitted and received data is free of errors, the OCD transmits an 8-bit cyclic redundancy check (CRC) at the end of each command. The CRC is enabled after the OCD is initialized, it is not sent with the first read revision command. This CRC is disabled by clearing the CRCEN bit of the DBGCTL Register.

The CRC is reset at the beginning of each command and is computed on the data received from and sent to the host. The CRC is calculated using the ATM-8 HEC polynomial $x^8+x^2+x^1+x^0$. The CRC is preset to all ones. Data is shifted through the polynomial LSB first. The resulting CRC is reversed and inverted. The check value is CFh.

Memory Cyclic Redundancy Check

The read memory CRC command computes the CRC on memory in 4K blocks, up to 4K blocks at a time (16M of data). The Memory CRC is computed using the 16-bit CCITT polynomial $x^{16}+x^{12}+x^5+x^0$. The CRC is preset to all ones. Data is shifted through the polynomial LSB first. The resulting CRC is reversed and inverted. The check value is F0B8h.

UART Mode

When the OCD is disabled, the DBG pin is used as a single pin half-duplex UART. When the serial interface is in UART Mode, data received on the single wire bus is written to the Receive Data register. Data written to the Transmit Data register is transmitted on the single wire bus. In UART Mode, the auto-baud hardware is used to configure the BRG, or the baud rate registers are written to set a specific baud rate.

The UARTEN control bit must be set to 1 to use the serial interface as a UART. Clearing the UARTEN control bit to 0 will prevent data received on the DBG pin from being written to the Receive Data register. Clearing the UARTEN control bit to 0 also prevents data written to the Transmit Data register from being transmitted on the single pin interface.

If the UART is disabled, data is still written to the Receive Data register and read from the Transmit Data register. These actions still generates UART interrupts. The UARTEN control bit only prevents data from being transmitted to or received from the DBG pin.

Serial Errors

The serial interface detects the following error conditions:

- Receive framing error (received Stop bit is Low)
- Transmit collision (OCD releases the bus high to send a logic 1 and detects it is Low)
- Receive overrun (received data before previously received data read)
- Receive break detect (10 or more bits Low)

Transmission of data is prevented if the transmit collision, receive framing error, receive break detect, receive overrun, or Receive Data Register full status bits are set.

Interrupts

The Debug UART generates interrupts during the following conditions:

- Receive Data register is Full (includes Rx Framing Error and Rx Overrun Error)
- Transmit Data register is empty
- Auto-Baud Detector loads the BRG (auto-baud character received)
- Receive Break detected

DBG Pin as a GPIO Pin

The DBG pin is used as a GPIO pin. The serial interface cannot be used for debugging when the DBG pin is configured as a GPIO pin. To set up the DBG pin as a GPIO pin, software must clear the DBGUART option bit and OCDEN control bit.

Software uses the pin as an input by clearing the output enable control bit. The PIN status bit in Line Control Register (DBGLCR) reflects the state of the DBG pin.

The DBG pin is configured as an output pin by setting the output enable control bit. The logic state of the IDLE bit in Line Control Register is driven onto the DBG pin.

Control Register Definitions

Receive Data Register

The Receive Data Register (DBGRXD), shown in Table 170, holds data received by the serial UART.

Table 170. Receive Data Register (DBGRXD)

Bits	7	6	5	4	3	2	1	0
Field	RXDATA							
RESET	XX							
R/W	R/W							
Addr	FF_E080							

Bit	Description
[7:0] RXDATA	**Receive Data** In UART Mode, data received on the serial interface is transferred from the shift register into this register. This register is written to simulate data received if the DBG pin is being used by the OCD.

Transmit Data Register

The Transmit Data Register (DBGTXD), shown in Table 171, holds data to be transmitted by the serial UART.

Table 171. Transmit Data Register (DBGTXD)

Bits	7	6	5	4	3	2	1	0
Field	TXDATA							
RESET	XX							
R/W	R/W							
Addr	FF_E081							

Bit	Description
[7:0] TXDATA	**Transmit Data** In UART Mode, data written to this register is transmitted on the serial interface. This register is read to simulate data transmitted if the DBG pin is being used by the OCD.

Baud Rate Reload Register

The Baud Rate Reload Register (DBGBR), shown in Table 172, is used to configure the baud rate of the serial communication stream. This register is automatically set by the Auto-Baud Detector. This register cannot be written by the CPU when OCDLOCK is set.

Table 172. Baud Rate Reload Register (DBGBR)

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	RELOAD															
RESET	0000h															
R/W	R/W															
Addr	FF_E082-FF_E083															

Bit	Description
[7:0] RELOAD	**Baud Rate Reload Value** This value is the baud rate reload value used to generate a bit clock. It is calculated as: $\text{RELOAD} = \frac{\text{SYSTEM CLOCK}}{\text{BAUD RATE}} \times 8$

Line Control Register

The Line Control Register (DBGLCR), shown in Table 173, controls the state of the UART. This register cannot be written by the CPU when OCDLOCK is set.

Table 173. Line Control Register (DBGLCR)

Bits	7	6	5	4	3	2	1	0
Field	OE	TDH	HDS	TXFC	NBEN	NB	OUT	PIN
RESET	0	0	0	0	0	0	1	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Addr	FF_E084							

Bit	Description
[7] OE	**Output Enable** This bit controls the output driver. If the UART is enabled, this bit controls the output driver during transmission only. 0 = Pin is open-drain during UART transmit. Pin behaves as an input if UART is disabled. 1 = Pin is driven during transmission if UART is enabled. Pin is an output if UART is disabled.

Bit	Description (Continued)
[6] TDH	**Transmit Drive High** This control bit causes the interface to drive the line high when a logic 1 is being transmitted. If OE is zero, the line stops being driven when the input is high or at the center of the bit, whichever is first. If OE is one, the line is driven high for one clock cycle. This bit is ignored if Debug Mode is zero and the UART is disabled. 0 = Transmit Drive High disabled. 1 = Transmit Drive High enabled.
[5] HDS	**High Drive Strength** This control bit enabled high drive strength for the output driver. 0 = Low Drive Strength 1 = High Drive Strength
[4] TXFC	**Transmitter Start Bit Flow Control** This control bit enables start bit flow control on the transmitter. The transmitter waits until a remote device sends a start bit before transmitting its data. 0 = Transmitter start bit flow control disabled. 1 = Transmitter start bit flow control enabled.
[3] NBEN	**9-Bit Mode Enable** This control bit enables transmission and reception of a ninth data bit. 0 = Nine bit mode disabled. 1 = Nine bit mode enabled.
[2] NB	**Value Of Ninth Bit** This bit is the value of the ninth data bit. When written, this reflects the ninth data bit that will be transmitted if nine bit mode is enabled. When read, this bit reflects the value of the ninth bit of the last nine bit character received. 0 = Ninth bit is zero. 1 = Ninth bit is one.
[1] OUT	**Output State** This control bit sets the state of the output transceiver. If the UART is enabled, this bit must be set to 1 to idle high. Clearing this bit to 0 when the UART is enabled will transmit a break condition. If the UART is disabled, this logic value will be driven onto the pin if OE is set. This bit is ignored in Debug Mode. 0 = Transmit Break if UART enabled. Drive Low if UART disabled and output enabled. 1 = Idle High if UART enabled. Drive high if UART disabled and output enabled.
[0] PIN	**Debug Pin** This bit reflects the state of the DBG pin. 0 = DBG pin is Low. 1 = DBG pin is High.

Status Register

The Status Register (DBGSTAT), shown in Table 174, contains status information about the state of the UART.

Table 174. Status Register (DBGSTAT)

Bits	7	6	5	4	3	2	1	0
Field	RDRF	RXOV	RXFE	RXBRK	TDRE	TXCOL	RXBUSY	TXBUSY
RESET	0	0	0	0	1	0	0	0
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1S	R/W1C	R	R
Addr	FF_E085							

Bit	Description
[7] RDRF	**Receive Data Register Full** This bit reflects the status of the Receive Data register. When data is written to the Receive Data register, or data is transferred from the shift register to the Receive Data register, this bit is set to 1. When the Receive Data register is read, this bit is cleared to 0. This bit is also cleared to 0 by writing a one to this bit. 0 = Receive Data register is empty. 1 = Receive Data register is full.
[6] RXOV	**Receive Overrun** This bit is set when a Receive Overrun occurs. A Receive Overrun occurs when there is data in the Receive Data register and another byte is written to this register. 0 = Receive Overrun has not occurred 1 = Receive Overrun has occurred.
[5] RXFE	**Receive Framing Error** This bit is set when a Receive Framing error has been detected. This bit is cleared by writing a one to this bit. 0 = No Framing Error detected. 1 = Receive Framing Error detected.
[4] RXBRK	**Receive Break Detect** This bit is set when a Break condition has been detected. This occurs when 10 or more bits received are Low. This bit is cleared by writing a one to this bit. 0 = No Break detected. 1 = Break detected.
[3] TDRE	**Transmit Data Register Empty** This bit reflects the status of the Transmit Data register. When the Transmit Data register is written, this bit is cleared to 0. When data from the Transmit Data Register is read or transferred to the transmit shift register, this bit is set to 1. This bit is written to 1 to abort the transmission of data being held in the Transmit Data Register. 0 = Transmit Data register is full. 1 = Transmit Data register is empty.

Bit	Description (Continued)
[2] TXCOL	**Transmit Collision** This bit is set when a Transmit Collision occurs. This bit is cleared by writing a one to this bit. 0 = No collision has been detected. 1 = Transmit Collision has been detected.
[1] RXBUSY	**Receiver Busy** This bit is set when the receiver is receiving the data. Multi-master systems uses this bit to ensure the line is idle before sending the data. 0 = Receiver is idle. 1 = Receiver is receiving data.
[0] TXBUSY	**Transmitter Busy** This bit is set when the transmitter is sending the data. This bit is used to determine when to turn off a transceiver for RS-485 applications. 0 = Transmitter is idle. 1 = Transmitter is sending the data.

Control Register

The Control Register (DBGCTL), shown in Table 175, sets the mode of the serial interface.

Table 175. Control Register (DBGCTL)

Bits	7	6	5	4	3	2	1	0
Field	OCDLOCK	OCDEN	Reserved		CRCEN	UARTEN	ABCHAR	ABSRCH
RESET	1	1	00		1	0	0	1
R/W	R/W	R/W	R		R/W	R/W	R/W	R/W
Addr	FF_E086							

Bit	Description
[7] OCDLOCK	**On-Chip Debug Lock** This bit locks the Debug Control register so it cannot be written by the CPU. This bit is automatically set if the DBGUART option bit is in its default erased state (one). 0 = Debug Control register unlocked. 1 = Debug Control register locked.
[6] OCDEN	**On-Chip Debug Enable** This bit is set when the OCD is enabled. When this bit is set, received data is interpreted as debug command. To use the DBG pin as a UART or GPIO pin, this bit must be cleared to 0 by software. This bit cannot be written by the CPU if OCDLOCK is set. 0 = OCD is disabled. 1 = OCD is enabled.

Bit	Description (Continued)
[5:4]	**Reserved** These bits are reserved and must be programmed to 00.
[3] CRCEN	**CRC Enable** If this bit is set, a CRC is appended to the end of each debug command. Clearing this bit will disable transmission of the CRC. 0 = CRC disabled 1 = CRC enabled
[2] UARTEN	**UART Enable** This bit is used to enable or disable the UART. This bit is ignored when OCDEN is set. 0 = UART Disabled. 1 = UART Enabled.
[1] ABCHAR	**Auto-Baud Character** This bit selects the character used during auto-baud detection. This bit cannot be written by the CPU if OCDEN is set. 0 = Auto-baud character to be measured is 80h. 1 = Auto-baud character to be measured is 0Dh.
[0] ABSRCH	**Auto-Baud Search Mode** This bit enables auto-baud search mode. When this bit is set, the next character received is measured to set the Baud Rate Reload register. This bit clears itself to 0 after the reload register has been written. This bit is automatically set when OCDEN is set if a serial communication error occurs. This bit cannot be written by the CPU if the OCDEN bit is set. 0 = Auto-baud search disabled. 1 = Auto-baud search enabled.

OCD Control Register

The OCD Control Register (OCDCTL), shown in Table 176, controls the state of the CPU. This register puts the CPU in DEBUG Halt Mode, enables breakpoints, or single-steps an instruction.

Table 176. OCD Control Register (OCDCTL)

Bits	7	6	5	4	3	2	1	0
Field	DBGHALT	BRKHALT	BRKEN	DBGSTOP	Reserved			STEP
RESET	0	0	0	0	000			0
R/W	R/W	R/W	R/W	R/W	R			R/W

Bit	Description
[7] DBGHALT	**Debug Halt** Setting this bit to 1 causes the device to enter DEBUG HALT Mode. When in DEBUG HALT Mode, the CPU stops fetching instructions. Clearing this bit causes the CPU to start running again. This bit is automatically set to 1 when a breakpoint occurs if the BRKHALT bit is set. 0 = The device is running. 1 = The device is in DEBUG HALT Mode.
[6] BRKHALT	**Breakpoint Halt** This bit determines what action the OCD takes when a Breakpoint occurs. If this bit is set to 1, then the DBGHALT bit is automatically set to 1 when a breakpoint occurs. If BRKHALT is zero, then the CPU will loop on the breakpoint. 0 = CPU loops on current instruction when breakpoint occurs. 1 = A Breakpoint sets DBGHALT to 1.
[5] BRKEN	**Enable Breakpoints** This bit controls the behavior of the BRK instruction and the hardware breakpoint. By default, these generate an illegal instruction system trap. If this bit is set to 1, these events generate a Breakpoint instead of a system trap. The resulting action depends upon the BRKHALT bit. 0 = BRK instruction and hardware breakpoint generates system trap. 1 = BRK instruction and hardware breakpoint generates a breakpoint.
[4] DBGSTOP	**Debug Stop Mode** This bit controls the system clock behavior in Stop Mode. When set to 1, the system clock will continue to operate in Stop Mode. 0 = Stop Mode debug disabled. system clock stops in Stop Mode. 1 = Stop Mode debug enabled. system clock runs in Stop Mode.

Bit	Description (Continued)
[3:1]	**Reserved** These bits are reserved and must be programmed to 000.
[0] STEP	**Single-Step An Instruction** This bit is used to single step an instruction when in DEBUG Halt Mode. This bit is automatically cleared after an instruction is executed. 0 = Idle 1 = Single Step an Instruction.

OCD Status Register

The OCD Status Register (OCDSTAT), shown in Table 177, reports status information about the current state of the system.

Table 177. OCD Status Register (OCDSTAT)

Bits	7	6	5	4	3	2	1	0
Field	DBGHALT	DBGBRK	HALT	STOP	RPEN	Reserved	TDRF	RDRE
RESET	0	0	0	0	0	0	0	1
R/W	R	R	R	R	R	R	R	R

Bit	Description
[7] DBGHALT	**DEBUG Halt Mode** This status bit indicates if the CPU is stopped and in DEBUG HALT Mode. 0 = Device is running. 1 = CPU is in DEBUG HALT Mode.
[6] DBGBRK	**Debug Break** This bit indicates if the CPU has reached a BRK instruction. This bit is set when a BRK instruction is executed. It is cleared when the DBGHALT control bit is written to 0.
[5] HALT	**Halt Mode** 0 = The device is not in Halt Mode. 1 = The device is in Halt Mode.
[4] STOP	**Stop Mode** 0 = The device is not in Stop Mode. 1 = The device is in Stop Mode.
[3] RPEN	**Read Protect Enabled** 0 = Memory Read Protect is disabled. 1 = Memory Read Protect is enabled.
[2]	**Reserved** This bit is reserved and must be programmed to 0.
[1] TDRF	**Transmit Data Register Full** This bit is set when the Transmit Data Register is full. 0 = Transmit Data register is empty 1 = Transmit Data register is full
[0] RDRE	**Receive Data Register Empty** This bit indicates when the Receive Data Register is empty. 0 = Receive Data register is full. 1 = Receive Data register is empty.

Hardware Breakpoint Registers

The Hardware Breakpoint Register (HWBPn), shown in Table 178, is used to set hardware breakpoints.

Table 178. Hardware Breakpoint Register (HWBPn)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	PC	ST	RD	WR	MASK				ADDR[23:16]							
RESET	0	0	0	0	0000				00h							
R/W	R/W	R/W	R/W	R/W	R/W				R/W							
Addr	FF_E090-FF_E091,FF_E094-FF_E095,FF_E098-FF_E099,FF_E09C-FF_E09D															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	ADDR[15:0]															
RESET	0000h															
R/W	R/W															
Addr	FF_E092-FF_E093,FF_E096-FF_E097,FF_E09A-FF_E09B,FF_E09E-FF_E09F															

Bit	Description
[31] PC	**Break on Program Counter Match** This bit will enable the hardware breakpoint. 0 = Break on program counter match disabled. 1 = Break on program counter match enabled.
[30] ST	**Status** This bit is set when a hardware breakpoint occurs. 0 = No breakpoint occurred because this bit was last written to 0. 1 = Breakpoint has occurred or this bit written to 1.
[29] RD	**Break On Data Read** This bit will enable the hardware watchpoint for data reads. 0 = Hardware watchpoint on read disabled. 1 = Hardware Watchpoint on read enabled.
[28] RW	**Break On Data Write** This bit will enable the hardware watchpoint for data writes. 0 = Hardware watchpoint on data write disabled. 1 = Hardware watchpoint on data write enabled.
[27:24] MASK	**Watchpoint Address Mask** The MASK field specifies the number of bits in ADDR to ignore when comparing against addresses for read and write watchpoints. The mask is set to ignore 0 to 15 of the lower address bits. This allows the watchpoint to monitor a memory block up to 32K in size.

Bit	Description (Continued)
[23:16] ADDR[23:16]	**Breakpoint Address** The address to match when generating a breakpoint.
[15:0] ADDR[15:0]	

Trace Control Register

The Trace Control Register (TRACECTL), shown in Table 179, is used to enable the Trace operation. It also selects the size of the trace buffer.

Table 179. Trace Control Register (TRACECTL)

Bits	7	6	5	4	3	2	1	0
Field	TRACEEN	Reserved				TRACESEL		
RESET	0	0	0	0	0	000		
R/W	R/W	R	R	R	R	R/W		
Addr	FF_E013							

Bit	Description
[7] TRACEEN	**Trace Enable** 0 = Trace is disabled. 1 = Traces is enabled
[6:3]	**Reserved** This bit is reserved and must be programmed to 0000.
[2:0] TRACESEL	**Trace Size Select** 000 – 128 Bytes (16 Events) 001 – 256 Bytes (32 Events) 010 – 512 Bytes (64 Events) 011 – 1024 Bytes (128 Events) 100 – 2048 Bytes (256 Events) 101 – 4096 Bytes (512 Events) 110 – 8192 Bytes (1024 Events) 111 – 16384 Bytes (2048 Events)

Trace Address Register

The Trace Address Register (TRACEADDR), shown in Table 180, points to the next data trace location.

Table 180. Trace Address (TRACEADDR)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved								TRACEADDR[23:16]							
RESET	00h								XXH							
R/W	R								R/W							
Addr	FF_E014															
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TRACEADDR[15:2]														00	
RESET	XXXXH														00	
R/W	R/W														R	
Addr	FF_E016															

Bit	Description
[31:24]	**Reserved** This bit is reserved and must be programmed to 00000000.
[23:16] TRACEADDR[23:16]	**Trace Address** These bits form a 24-bit address used by the trace logic to store the next PC value to memory.
[15:2] TRACEADDR[15:2]	
[1:0] 00	These read-only bits are unused.

On-Chip Oscillator

The products in the ZNEO® Z16F Series feature an on-chip oscillator for use with external crystals with frequencies from 32 kHz to 20 MHz. In addition, the oscillator supports external RC networks with oscillation frequencies up to 4 MHz or ceramic resonators with oscillation frequencies up to 20 MHz. This oscillator generates the primary system clock for the internal ZNEO CPU and the majority of the on-chip peripherals. Alternatively, the X_{IN} input pin also accept a CMOS-level clock input signal (32 kHz to 20 MHz). If an external clock generator is used, the X_{OUT} pin must be left unconnected.

When configured for use with crystal oscillators or external clock drivers, the frequency of the signal on the X_{IN} input pin determines the frequency of the system clock (that is, no internal clock divider). In RC operation, the system clock is driven by a clock divider (divide by 2) to ensure 50% duty cycle.

Operating Modes

The ZNEO Z16F 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 292.

Crystal Oscillator Operation

Figure 70 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 181. 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, it reduce the values of capacitors C1 and C2 to decrease loading.

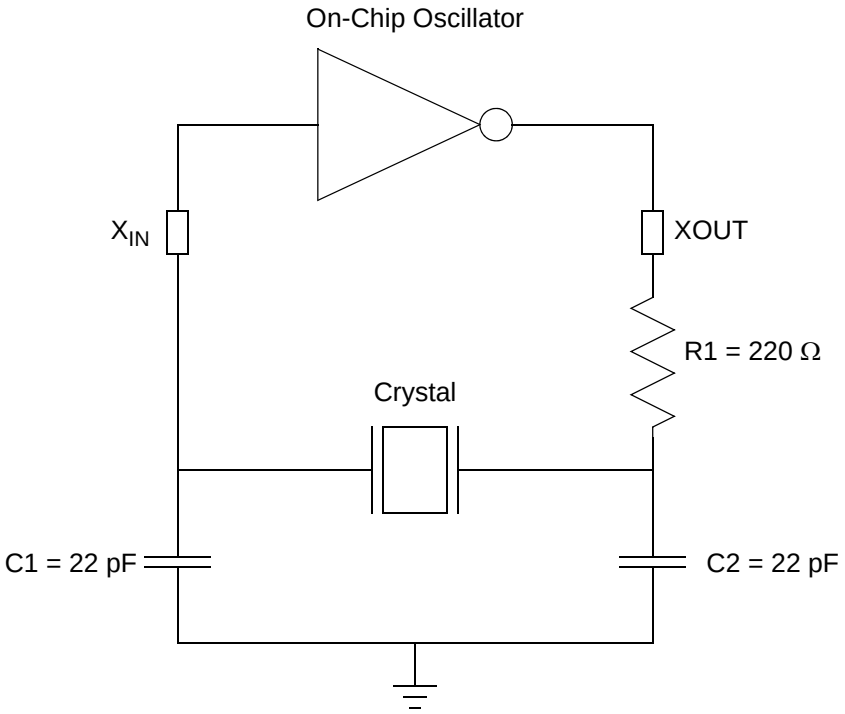


Figure 70. Recommended 20 MHz Crystal Oscillator Configuration

Table 181. Recommended Crystal Oscillator Specifications (20 MHz Operation)

Parameter	Value	Units	Comments
Frequency	20	MHz	
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	25	Ω	Maximum
Load Capacitance (C_L)	20	pF	Maximum
Shunt Capacitance (C_0)	7	pF	Maximum
Drive Level	1	mW	Maximum

Oscillator Operation with an External RC Network

Figure 71 displays a recommended configuration for connection with an external resistor-capacitor (RC) network.

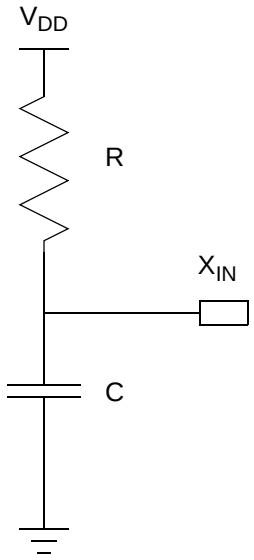


Figure 71. Connecting the On-Chip Oscillator to an External RC Network

An external resistance value of 15 kΩ is recommended for oscillator operation with an external RC network. The minimum resistance value to ensure operation is 10 kΩ. The typical oscillator frequency is estimated from the values of the resistor (*R* in kΩ) and capacitor (*C* in pF) elements using the following equation:

$$\text{Oscillator Frequency (kHz)} = \frac{1 \times 10^6}{(1.5 \times R \times C)}$$

Figure 3 displays the typical (3.3 V and 25°C) oscillator frequency as a function of the capacitor (*C* in pF) employed in the RC network assuming a 15 kΩ external resistor. For very small values of C, the parasitic capacitance of the oscillator X_{IN} pin and the printed circuit board must be included in the estimation of the oscillator frequency.

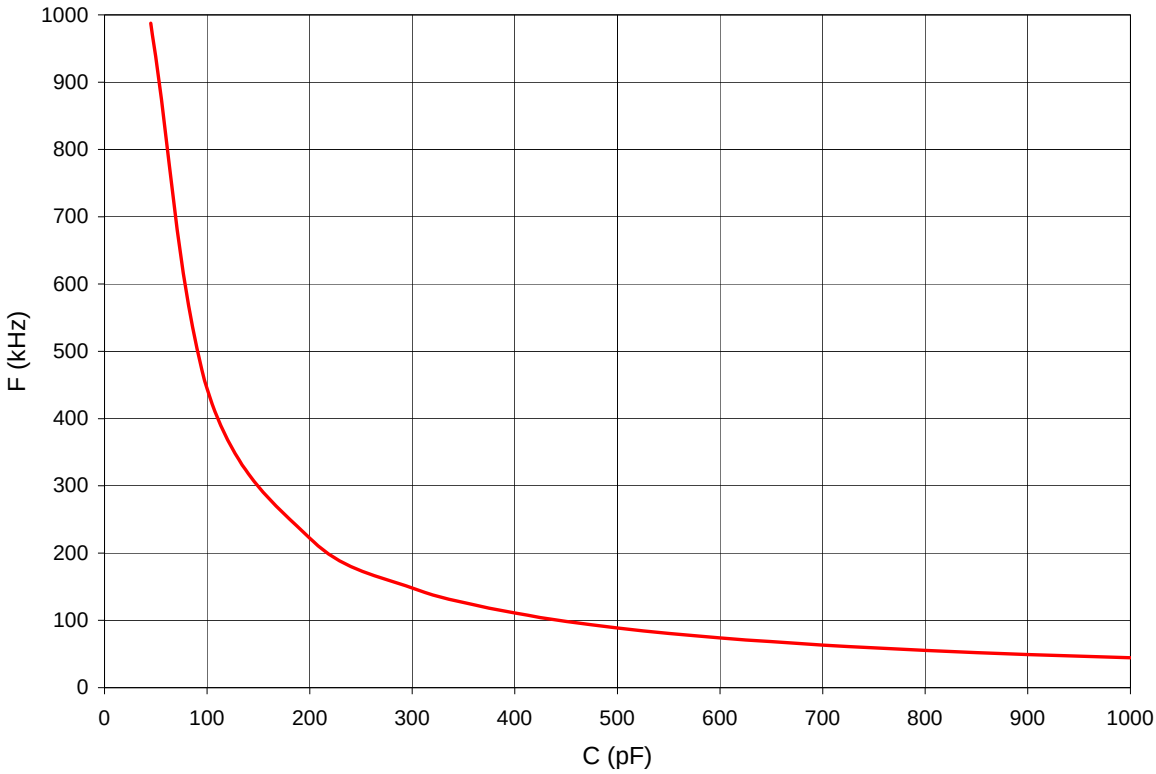


Figure 72. Typical RC Oscillator Frequency as a Function of the External Capacitance with a 15 kΩ Resistor

Oscillator Control

The ZNEO® Z16F Series uses three possible user-selectable clocking schemes:

- Trimmable internal precision oscillator
- On-chip oscillator using off-chip crystal/resonator or external clock driver
- On-chip low-precision Watchdog Timer oscillator

In addition, ZNEO Z16F Series contain clock failure detection and recovery circuitry, allowing continued operation despite a failure of the primary oscillator.

The on-chip system clock frequency is reduced through a clock divider allowing reduced dynamic power dissipation. The FLASH is powered down during portions of the clock period when running slower than 10 MHz.

Operation

This section explains the logic used to select the system clock, divide down the system clock and handle oscillator failures. A description of the specific operation of each oscillator is outlined elsewhere in this document. See the Watchdog Timer chapter on page 238, the Internal Precision Oscillator chapter on page 336 and the On-Chip Oscillator chapter on page 327.

System Clock Selection

The oscillator control block selects from the available clocks. Table 182 details each clock source and its usage.

Table 182. Oscillator Configuration and Selection

Clock Source	Characteristics	Required Setup
Internal Precision Oscillator	• 5.5 MHz • High precision possible when trimmed • No external components required	• The reset default.
External Crystal/ Resonator/ External Clock Drive	• 0 to 20 MHz • Very high accuracy (dependent on crystal/resonator or external source) • Requires external components	• Configure Option Bits for correct external oscillator mode • Unlock and write Oscillator Control Register (OSCCTL) to enable external oscillator • Wait for required stabilization time • Unlock and write Oscillator Control Register (OSCCTL) to select external oscillator
Internal Watchdog Timer Oscillator	• 10 kHz nominal • Low accuracy • No external components required • Low power consumption	• Unlock and write Oscillator Control Register (OSCCTL) to enable and select Internal WDT oscillator

Unintentional access to the Oscillator Control Register (OSCCTL) stops the chip by switching to a nonfunctioning oscillator. Accidental alteration of the OSCCTL Register is prevented by a locking/unlocking scheme. To write the register, unlock it by making two writes to the OSCCTL Register with the values E7h followed by 18h. A third write to the OSCCTL Register then changes the value of the register and returns the register to a locked state. Any other sequence of oscillator control register writes has no effect. The values written to unlock the register must be ordered correctly, but is not required to be consecutive. It is possible to access other registers within the locking/unlocking operation.

Clock Selection Following System Reset

The internal precision oscillator is selected following a System Reset. Startup code after the System Reset changes the system clock source by unlocking and configuring the OSCCTL Register. If the LPOPT bit in Program Memory Address 0003h is zero, Flash Low Power Mode is enabled during reset. When Flash Low Power Mode is enabled during reset, the FLPEN bit in the Oscillator Control Register (OSCCTL) will be set and the DIV field of the OSCDIV Register will be set to 08h.

Clock Failure Detection and Recovery

Primary Oscillator Failure

The ZNEO Z16F Series generates a System Exception when a failure of the primary oscillator occurs if the POFEN bit is set in the OSCCTL Register. To maintain system function in this situation, the clock failure recovery circuitry automatically forces the Watchdog Timer oscillator to drive the system clock. Although this oscillator runs at a much lower frequency than the original system clock, the CPU continues to operate, allowing execution of a clock failure vector and software routines that either remedy the oscillator failure or issue a failure alert. This automatic switch-over is not available if the WDT is the primary oscillator.

The primary oscillator failure detection circuitry asserts if the system clock frequency drops below 1 kHz ±50%. For operating frequencies below 2 kHz, do not enable the clock failure circuitry (POFEN must be deserted in the OSCCTL Register).

Watchdog Timer Failure

In the event of a Watchdog Timer oscillator failure, a System Exception is used if the WDFEN bit of the OSCCTL Register is set. This event does not trigger an attendant clock switch-over, but alerts the CPU of the failure. After a WDT failure, it is no longer possible to detect a primary oscillator failure.

The Watchdog Timer oscillator failure detection circuit counts system clocks while looking for a WDT clock. The logic counts 8000 system clock cycles before determining that a failure occurred. The system clock rate determines the speed at which the WDT failure is detected. A very slow system clock results in very slow detection times.

If the WDT is the primary oscillator or if the Watchdog Timer oscillator is disabled, deassert the WDFEN bit of the OSCCTL Register.

Oscillator Control Register Definitions

Oscillator Control Register

The Oscillator Control Register (OSCCTL), shown in Table 183, enables or disables the various oscillator circuits, enables/disables the failure detection/recovery circuitry, actively powers down the flash and selects the primary oscillator, which becomes the system clock.

The Oscillator Control Register must be unlocked before writing. Writing the two-step sequence E7h followed by 18h to the Oscillator Control Register address unlocks it. The register locks after completion of a register write to the OSCCTL.

Table 183. Oscillator Control Register (OSCCTL)

Bits	7	6	5	4	3	2	1	0
Field	INTEN	XTLEN	WDTEN	POFEN	WDFEN	FLPEN	SCKSEL	
RESET	1	0	1	0	0	0*	00	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Addr	FF_E0A0h							

Note: *The reset value is 1 if the option bit LPOPT is 0.

Bit	Description
[7] INTEN	**Internal Precision Oscillator Enable** 0 = Internal precision oscillator is disabled. 1 = Internal precision oscillator is enabled.
[6] XTLEN	**Crystal Oscillator Enable** 0 = Crystal oscillator is disabled. 1 = Crystal oscillator is enabled.
[5] WDTEN	**WDT Oscillator Enable** 0 = WDT oscillator is disabled. 1 = WDT oscillator is enabled.
[4] POFEN	**Primary Oscillator Failure Detection Enable** 0 = Failure detection and recovery of primary oscillator is disabled. This bit is cleared automatically if a primary oscillator failure is detected. 1 = Failure detection and recovery of primary oscillator is enabled.
[3] WDFEN	**WDT Oscillator Failure Detection Enable** 0 = Failure detection of WDT oscillator is disabled.This bit is cleared automatically if a WDT oscillator failure is detected. 1 = Failure detection of WDT oscillator is enabled.
[2] FLPEN	**Flash Low Power Mode Enable** 0 = Flash Low Power Mode is disabled. 1 = Flash Low Power Mode is enabled. The Flash will be powered down during idle periods of the clock and powered up during Flash reads. This bit must only be set if the frequency of the primary oscillator source is 8 MHz or lower. The reset value of this bit is controlled by the LPOPT option bit during reset.
[1:0] SCKSEL	**System Clock Oscillator Select** 00 = Internal precision oscillator functions as system clock at 5.6 MHz. 01 = Crystal oscillator or external clock driver functions as system clock. 10 = Reserved. 11 = Watchdog Timer oscillator functions as system clock.

Oscillator Divide Register

The Oscillator Divide Register (OSCDIV), shown in Table 184, provides the value to divide the system clock by. The Oscillator Divide Register must be unlocked before writing. Writing the two-step sequence E7h followed by 18h to the Oscillator Control Register address unlocks it. The register locks after completion of a register write to the OSCDIV.

Table 184. Oscillator Divide Register (OSCDIV)

Bits	7	6	5	4	3	2	1	0
Field	DIV							
RESET	00h*							
R/W	R/W							
Addr	FF_E0A1h							

Note: *The reset value is 08h if the option bit LPOPT is 0.

Bit	Value (H)	Description
[7:0] DIV	00h–FFh	**Oscillator Divide** 00h–divider is disabled; all other entries are the divide value for scaling the system clock.

Internal Precision Oscillator

The Internal Precision Oscillator (IPO) is designed for use without external components. Nominal untrimmed accuracy is approximately ±30%. You can manually trim the oscillator to achieve a ±4% frequency accuracy over the operating temperature and supply voltage range of the device.

The IPO features include:

- On-chip RC oscillator which does not require external components
- Nominal ±30% accuracy without trim or manually trim the oscillator to achieve a ± 4%
- Typical output frequency of 5.5296 MHz
- Trimming possible through Flash option bits with user override
- Eliminates crystals or ceramic resonators in applications where high timing accuracy is not required

Operation

The internal oscillator is an RC relaxation oscillator and has its sensitivity to power supply variation minimized. By using ratio tracking thresholds, the effect of power supply voltage is cancelled out. The dominant source of oscillator error is the absolute variance of chip level fabricated components, such as capacitors. An 8-bit trimming register, incorporated into the design, allows compensation of absolute variation of oscillator frequency. After it is calibrated, the oscillator frequency is relatively stable and does not require subsequent calibration.

By default, the oscillator is configured through the Flash Option bits. However, the user code overrides these trim values as described in the Option Bits chapter on page 292.

Electrical Characteristics

All data in this chapter is prequalification and precharacterization and is subject to change.

Absolute Maximum Ratings

Stress greater than those listed in Table 185 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 affects device reliability. For improved reliability, unused inputs must be tied to one of the supply voltages (V_{DD} or V_{SS}).

Table 185. 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	2
Maximum current on input and/or inactive output pin	–5	+5	µA	
Maximum output current from active output pin	–25	+25	mA	
100-Pin LQFP Maximum Ratings at –40°C to 70°C				
Total power dissipation		1325	mW	
Maximum current into V_{DD} or out of V_{SS}		368	mA	
100-Pin LQFP Maximum Ratings at 70°C to 125°C				
Total power dissipation		482	mW	
Maximum current into V_{DD} or out of V_{SS}		134	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	

Notes:
1. This voltage applies to 5 V tolerant pins which are Port A, C, D, E, F and G pins (except pins PC0 and PC1).
2. This voltage applies to V_{DD}, AV_{DD}, pins supporting analog input (Ports B and H), Pins PC0 and PC1, RESET, DBG and X_{IN} pins which are non 5 V tolerant pins.

Table 185. Absolute Maximum Ratings (Continued)

Parameter	Minimum	Maximum	Units	Notes
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		1.0	W	
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	W	
Maximum current into V_{DD} or out of V_{SS}		140	mA	
64-Pin LQFP Maximum Ratings at –40°C to 70°C				
Total power dissipation		1.0	W	
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	W	
Maximum current into V_{DD} or out of V_{SS}		150	mA	

Notes:
1. This voltage applies to 5 V tolerant pins which are Port A, C, D, E, F and G pins (except pins PC0 and PC1).
2. This voltage applies to V_{DD}, AV_{DD}, pins supporting analog input (Ports B and H), Pins PC0 and PC1, RESET, DBG and X_{IN} pins which are non 5 V tolerant pins.

DC Characteristics

Table 186 lists the DC characteristics of the ZNEO® Z16F Series products. All voltages are referenced to V_{SS}, the primary system ground. Any parameter value in the typical column is from characterization at 3.3 V and 0°C. These values are provided for design guidance only and are not tested in production.

Table 186. DC Characteristics

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
V_{DD}	Supply Voltage	2.7	—	3.6	V	
V_{IL1}	Low Level Input Voltage	–0.3		0.3*V_{DD}	V	For all input pins except RESET, DBG, X_{IN}
V_{IL2}	Low Level Input Voltage	–0.3	—	0.2*V_{DD}	V	For 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[1] except pins PC0 and PC1
V_{IH2}	High Level Input Voltage	0.7*V_{DD}	—	V_{DD}+0.3	V	Port B, H and pins PC0 and PC1
V_{IH3}	High Level Input Voltage	0.8*V_{DD}	—	V_{DD}+0.3	V	RESET, DBG and X_{IN} pins
V_{OL1}	Low Level Output Voltage Standard Drive	—	—	0.4	V	I_{OL} = 2 mA; V_{DD} = 3.0 V High Output Drive disabled
V_{OH1}	High Level Output Voltage Standard Drive	2.4	—	—	V	I_{OH} = –2 mA; V_{DD} = 3.0 V High Output Drive disabled
V_{OL2}	Low Level Output Voltage High Drive	—	—	0.6	V	I_{OL} = 20 mA; V_{DD} = 3.3 V 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} = 3.3 V 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} = 3.3 V High Output Drive enabled; TA = +70°C to +125°C
V_{OH3}	High Level Output Voltage High Drive	2.4	—	—	V	I_{OH} = 15 mA; V_{DD} = 3.3 V High Output Drive enabled; TA = +70°C to +125°C
I_{IL}	Input Leakage Current	–5	v	+5	µA	V_{DD} = 3.6 V; V_{IN} = V_{DD} or V_{SS}[1]

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

Table 186. DC Characteristics (Continued)

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
I_{TL}	Tri-State Leakage Current	–5	—	+5	µA	V_{DD} = 3.6 V
C_{PAD}	GPIO Port Pad Capacitance	—	8.0[2]	—	pF	
C_{XIN}	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} = 2.7 V to 3.6 V
I_{CCS1}	Supply Current in Stop Mode with VBO enabled		600		µA	V_{DD} = 3.0 V; 25°C
I_{CCS2}	Supply Current in Stop Mode with VBO disabled		2		µA	V_{DD} = 3.0 V; 25°C
I_{CCS3}	Supply Current in Stop Mode with VBO disabled and WDT disabled		1		µA	V_{DD} = 3.0 V; 25°C
I_{CCA}	Active I_{DD} at 20MHz	—	18	35	mA	Typ: V_{DD}=3.0 V/30°C Max: V_{DD}=3.6 V/125°C Peripherals enabled, no loads
I_{CCH}	I_{DD} in Halt Mode at 20MHz	—	4	6	mA	Typ: V_{DD}=3.0 V/30°C Max: V_{DD}=3.6 V/125°C Peripherals off, no loads

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

Figure 73 displays the typical current consumption while operating at 3.3 V at 30 ºC versus the system clock frequency.

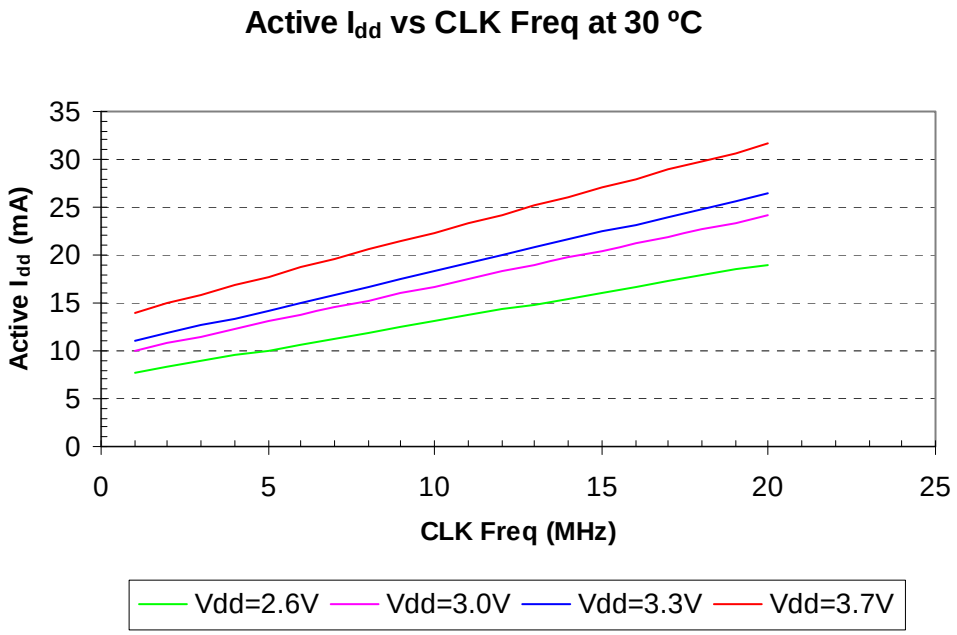


Figure 73. Typical I_{DD} Versus System Clock Frequency

Figure 74 displays typical current consumption while operating at 3.3 V at 30ºC in Halt Mode versus the system clock frequency.

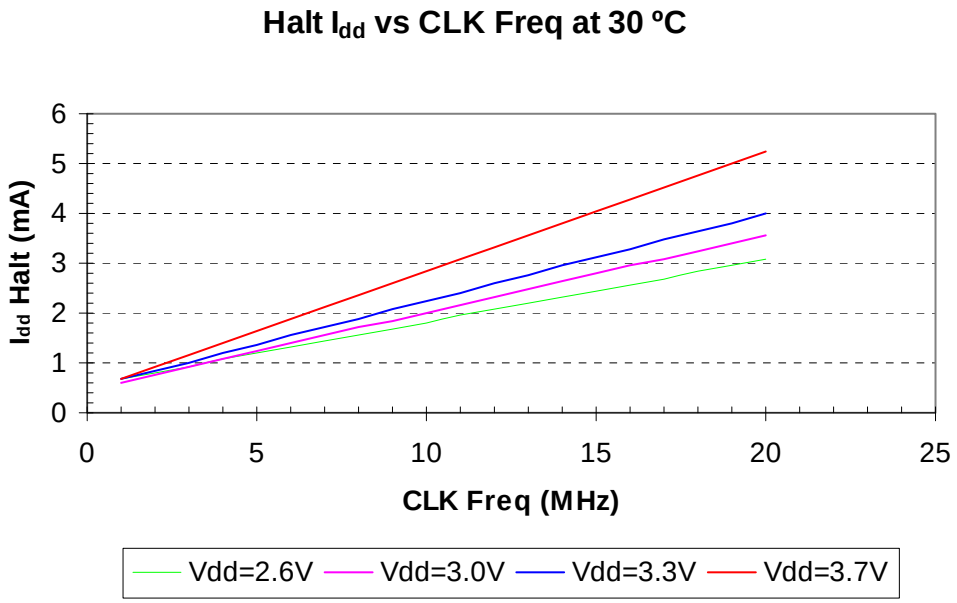


Figure 74. Typical Halt Mode I_{DD} Versus System Clock Frequency

Figure 75 displays the Stop Mode current consumption versus V_{DD} at ambient temperature with VBO and WDT disabled (I_{CCS2}).

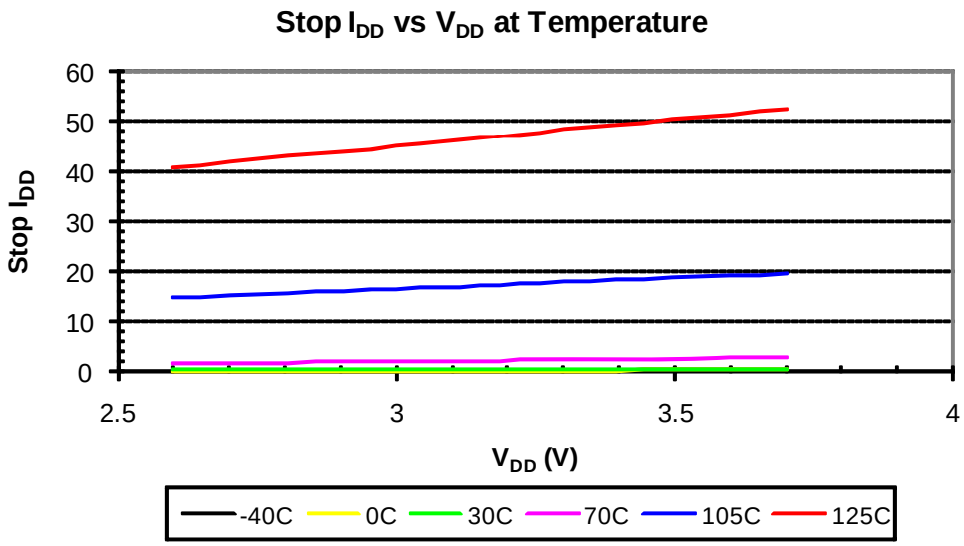


Figure 75. Stop Mode Current Versus V_{DD}

On-Chip Peripheral AC and DC Electrical Characteristics

Table 187 lists the POR and VBO electrical characteristics and timing. Table 188 lists the Reset and Stop Mode Recovery pin timing.

Table 187. POR and VBO Electrical Characteristics and Timing

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ[1]**	**Max**	**Units**	**Conditions**
V_{POR}	Power-On Reset voltage threshold	2.20	2.45	2.70	V	$V_{DD} = V_{POR}$
V_{VBO}	Voltage Brown-Out reset voltage threshold	2.15	2.40	2.65	V	$V_{DD} = V_{VBO}$
	V_{POR}–V_{VBO}		50	100	mV	
	Starting V_{DD} voltage to ensure valid POR	—	V_{SS}	—	V	
T_{ANA}	Power-On Reset analog delay	—	50	—	ms	$V_{DD} > V_{POR}$; T_{POR} Digital Reset delay follows T_{ANA}
T_{POR}	Power-On Reset digital delay	—	12	—	µs	66 IPO cycles
T_{VBO}	Voltage Brown-Out pulse rejection period	—	10	—	ms	$V_{DD} < V_{VBO}$ 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	
I_{CC}	Supply current		500		µA	V_{DD} = 3.3 V.

Note:
1. Data in the typical column is from characterization at 3.3 V and 0°C. These values are provided for design guidance only and are not tested in production.

Table 188. Reset and Stop Mode Recovery Pin Timing

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**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 189 lists the Flash Memory electrical characteristics and timing. Table 190 lists the WDT electrical characteristics and timing.

Table 189. Flash Memory Electrical Characteristics and Timing

	V_{DD} = 2.7 to 3.6V T_A = –40°C to 125°C				
Parameter	**Min**	**Typ**	**Max**	**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[1]
Data Retention	100	—	—	years	25°C
Endurance	10,000	—	—	cycles	Program/erase cycles

Note:
1. This parameter is only an issue when bypassing the Flash Controller.

Table 190. Watchdog Timer Electrical Characteristics and Timing

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
F_{WDT}	WDT Oscillator Frequency	5	10	20	kHz	

Table 191 lists the Analog-to-Digital Converter (ADC) electrical characteristics and timing.

Table 191. ADC Electrical Characteristics and Timing

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
	Resolution	10	—	—	bits	External V_{REF} = 2.0 V
	Throughput Conversion	13			CLKs	ADC clock cycles
	ADCCLK Frequency			20	MHz	
DNL	Differential Nonlinearity[1]	–0.99		2	LSB	Typical system config[2]
INL	Integral Nonlinearity[1]	–3		3	LSB	Typical system config[2]
	Offset Error[1]	–30		30	mV	Typical system config[2]
	Gain Error[1]	–4.5		4.5	LSB	Typical system config[2]
VREF	On-Chip Voltage Reference[3]	1.9	2	2.1	V	
	Externally supplied Voltage Reference	1.9	2	2.1	V	
	Analog Input Voltage Range	0		VREF	V	
	Analog Input Current			500	nA	
	Reference Input Current		2.0		mA	Worst case code
	Analog Input Capacitance			15	pF	
AVDD	Operating Supply Voltage	2.7		3.6	V	
	Operating Current, AVDD		9		mA	Active conversion @ 20 MHz
	Power Down Current		<1		µA	

Notes:
1. These parameters are guaranteed by design and not tested on every part.
2. Typical system configuration is defined as, 20 MHz clock with ADC clock divide by 4, 1 µs sample hold time, 0.5 µs sample settling time.
3. On-chip voltage reference cannot be used if AVDD is below 3.0 V.

Table 192 provides electrical characteristics and timing information for the on-chip comparator.

Table 192. Comparator Electrical Characteristics

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
V_{COFF}	Input offset	—	5		mV	V_{DD} = 3.3 V; V_{IN} = V_{DD} ÷ 2
T_{CPROP}	Propagation delay	—	200		ns	V_{COMM} mode = 1 V V_{DIFF} = 100 mV
I_B	Input bias current			1	µA	
CMVR	Common-mode voltage range	–0.3		V_{DD}–1	V	
I_{CC}	Supply current		40		µA	V_{DD} = 3.6 V
T_{wup}	Wake up time from off state			5	µs	CINP = 0.9 V CINN= 1.0 V

Table 193 provides electrical characteristics and timing information for the on-chip operational amplifier.

Table 193. Operational Amplifier Electrical Characteristics

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
V_{OS}	Input offset		5	15	mV	V_{DD} =3.3V; V_{CM} = V_{DD} ÷ 2
TC_{VOS}	Input offset Average Drift		1		µV/C	
I_B	Input bias current		TBD		µA	
I_{OS}	Input offset current		TBD		µA	
CMVR	Common-Mode Voltage Range	–0.3		V_{DD} – 1	V	
V_{OL}	Output Low			0.1	V	I_{SINK} = 100 µA
V_{OH}	Output High	V_{DD} – 1			V	I_{SOURCE} = 100 µA
CMRR	Common-Mode Rejection Ratio		70		dB	0 < V_{CM} < 1.4V; T_A = 25ºC

Table 193. Operational Amplifier Electrical Characteristics (Continued)

		T_A = –40°C to 125°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
PSRR	Power Supply Rejection Ratio		80		dB	V_{DD} = 2.7 V – 3.6 V; T_A = 25 ºC
A_{VOL}	Voltage Gain		80		dB	
SR+	Slew Rate while rising		12		V/µs	R_{LOAD} = 33 K; C_{LOAD} = 50 pF; A_{VCL} = 1, V_{IN} = 0.7 V to 1.7 V
SR-	Slew Rate while falling		16		V/µs	R_{LOAD} = 33 K; C_{LOAD} = 50 pF; A_{VCL} = 1, V_{IN} = 1.7 V to 0.7 V
GBW	Gain-Bandwidth Product	5			MHz	
FM	Phase Margin		50		degree	
I_S	Supply Current			1	mA	V_{DD} = 3.6 V; V_{OUT}= V_{DD} ÷ 2
T_{WUP}	Wake up time from off state			20	µs	

AC Characteristics

The section provides information about the AC characteristics and timing. All AC timing information assumes a standard load of 50 pF on all outputs. Table 194 lists the ZNEO Z16F Series AC characteristics and timing.

Table 194. AC Characteristics

		T_A = –40°C to 125°C			
Symbol	**Parameter**	**Min**	**Max**	**Units**	**Conditions**
F_{sysclk}	System Clock Frequency	—	20.0	MHz	Read-only from Flash memory
		0.032768	20.0	MHz	Program or erasure of the Flash memory
F_{XTAL}	Crystal Oscillator Frequency	1.0	20.0	MHz	System clock frequencies below the crystal oscillator minimum require an external clock driver
T_{XIN}	System Clock Period	50	—	ns	$T_{CLK} = 1/F_{sysclk}$
T_{XINH}	System Clock High Time	20	30	ns	T_{CLK} = 50 ns
T_{XINL}	System Clock Low Time	20	30	ns	T_{CLK} = 50 ns
T_{XINR}	System Clock Rise Time	—	3	ns	T_{CLK} = 50 ns
T_{XINF}	System Clock Fall Time	—	3	ns	T_{CLK} = 50 ns

General Purpose I/O Port Input Data Sample Timing

Figure 76 displays timing of the GPIO port input sampling. The input value on a GPIO port pin is sampled on the rising edge of the system clock. The port value is then available to the ZNEO CPU on the second rising clock edge following the change of the port value. Table 195 lists the GPIO port input timing.

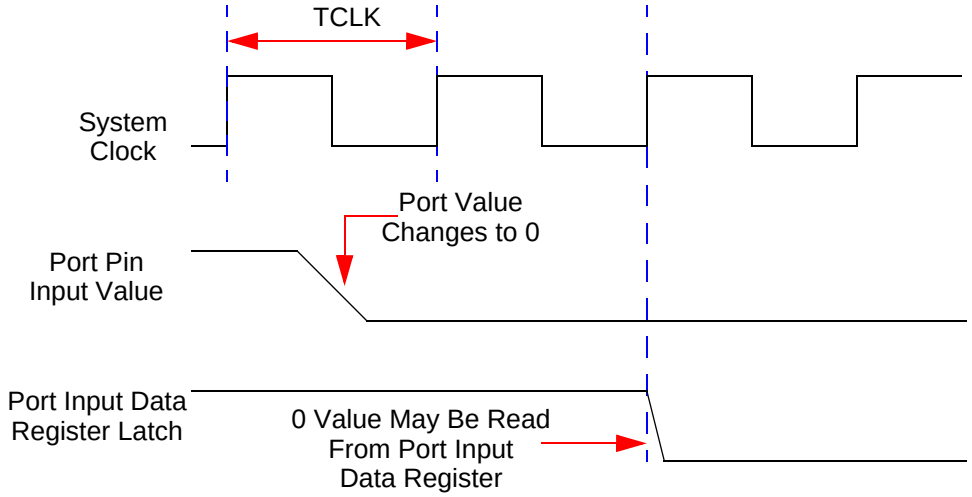


Figure 76. Port Input Sample Timing

Table 195. GPIO Port Input Timing

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
T_{SMR}	GPIO Port Pin Pulse Width to ensure Stop Mode Recovery (for GPIO Port Pins enabled as SMR sources)	1µs	

On-Chip Debugger Timing

Table 196 provides timing information for the DBG pin. The DBG pin timing specifications assume a 4µs maximum rise and fall time.

Table 196. On-Chip Debugger Timing

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
DBG	Debug frequency.	—	System Clock/4

SPI Master Mode Timing

Figure 77 and Table 197 provides 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.

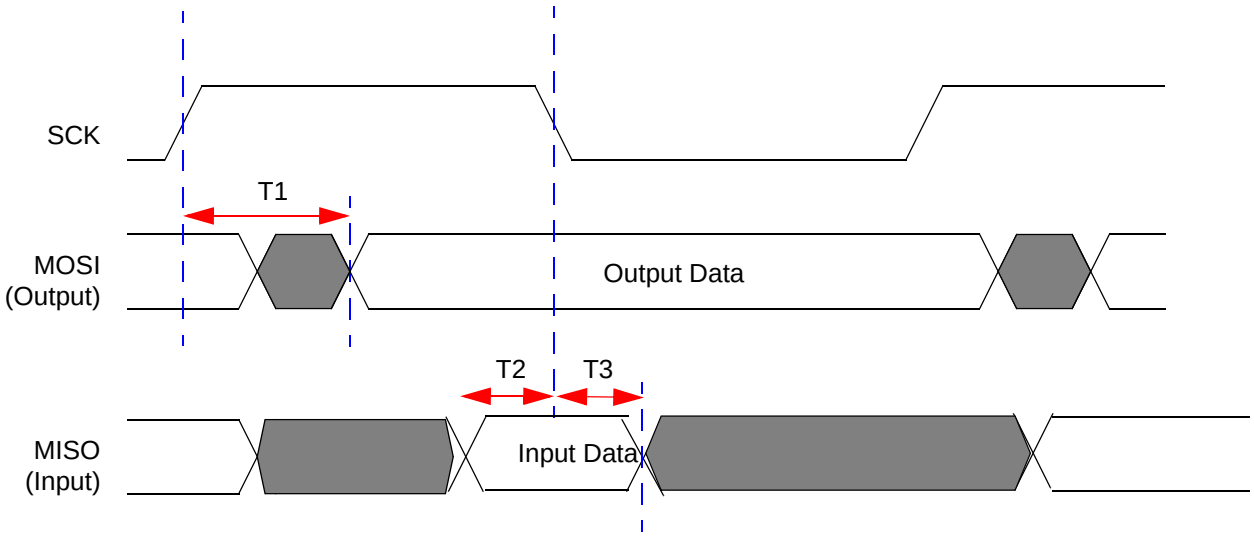


Figure 77. SPI Master Mode Timing

Table 197. SPI Master Mode Timing

		Delay (ns)	
Parameter	**Description**	**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 78 and Table 198 provide timing information for the SPI Slave Mode pins. Timing is shown with SCK rising edge used tosource MISO output data, SCK falling edge used to sample MOSI input data.

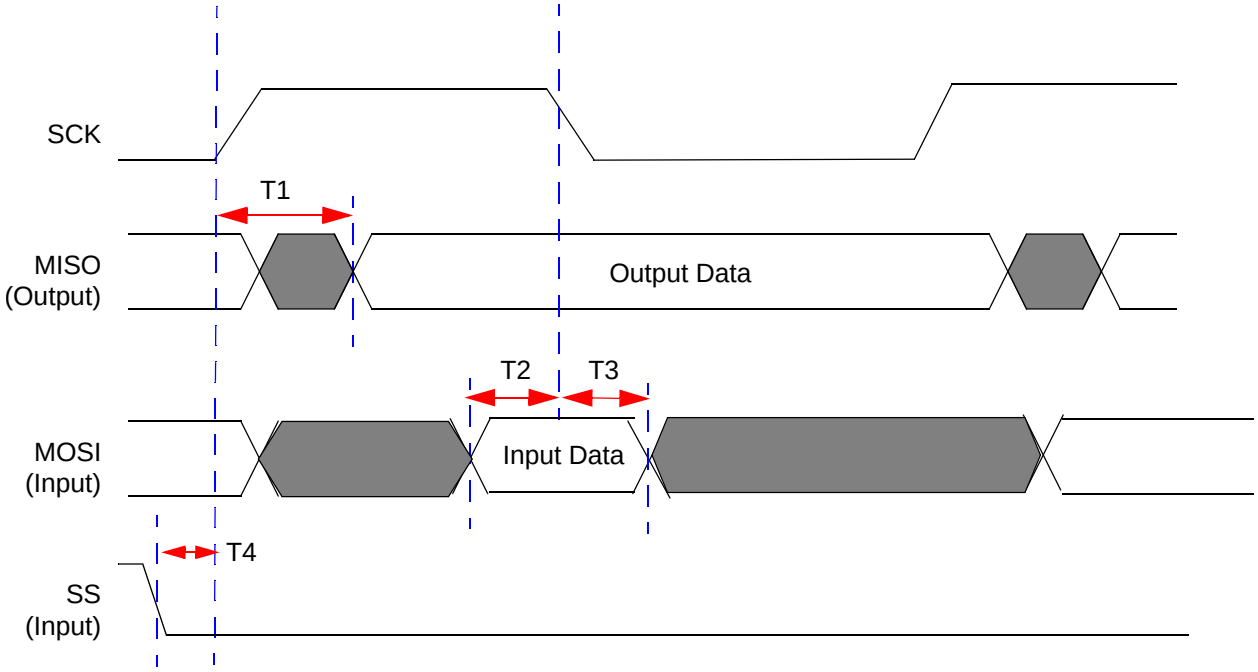


Figure 78. SPI Slave Mode Timing

Table 198. SPI Slave Mode Timing

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
SPI Slave			
T_1	SCK (transmit edge) to MISO output Valid Delay	2 * X_{IN} period	3 * X_{IN} period + 20 ns
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 79 and Table 199 provide timing information for I²C pins.

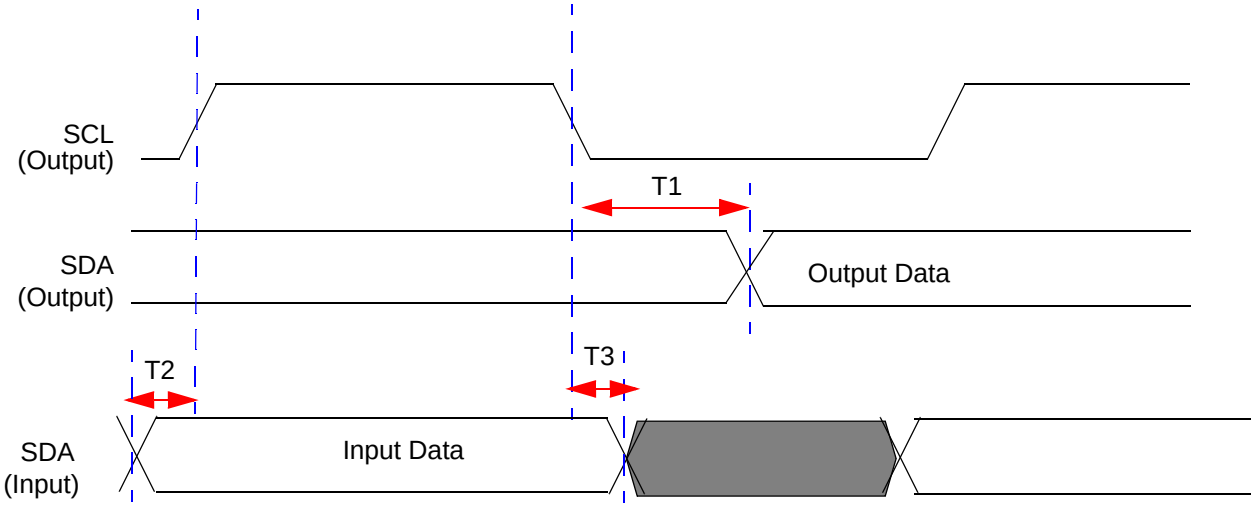


Figure 79. I²C Timing

Table 199. I²C Timing

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
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 80 and Table 200 provide timing information for the UART pins in situations in which 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.

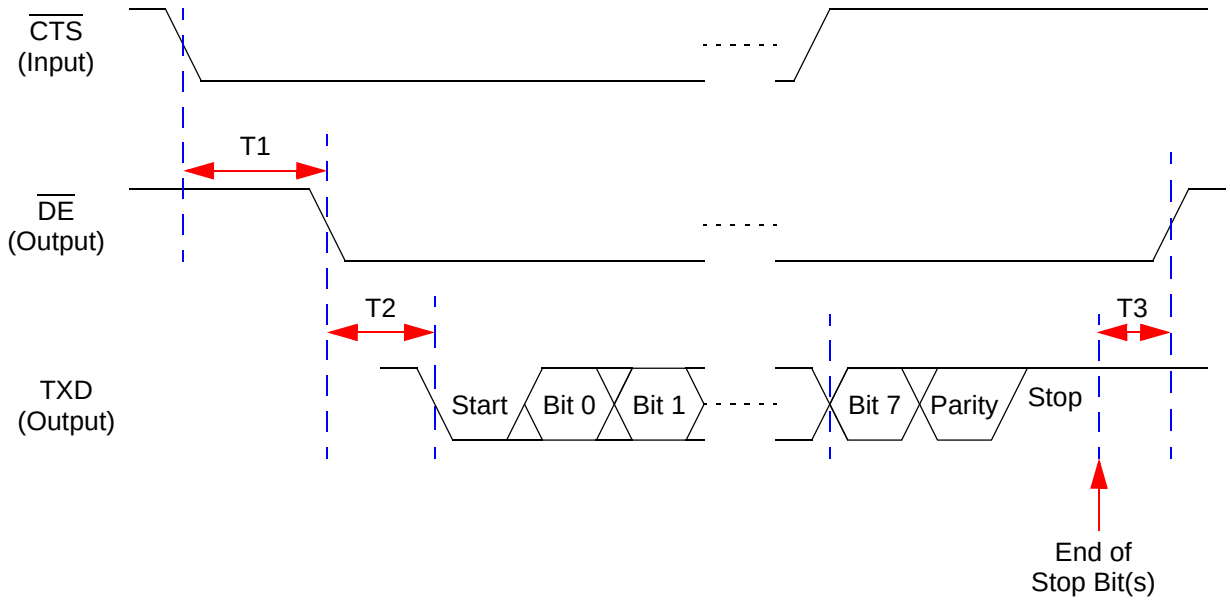


Figure 80. UART Timing with $\overline{CTS}$

Table 200. UART Timing with $\overline{CTS}$

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
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 81 and Table 201 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.

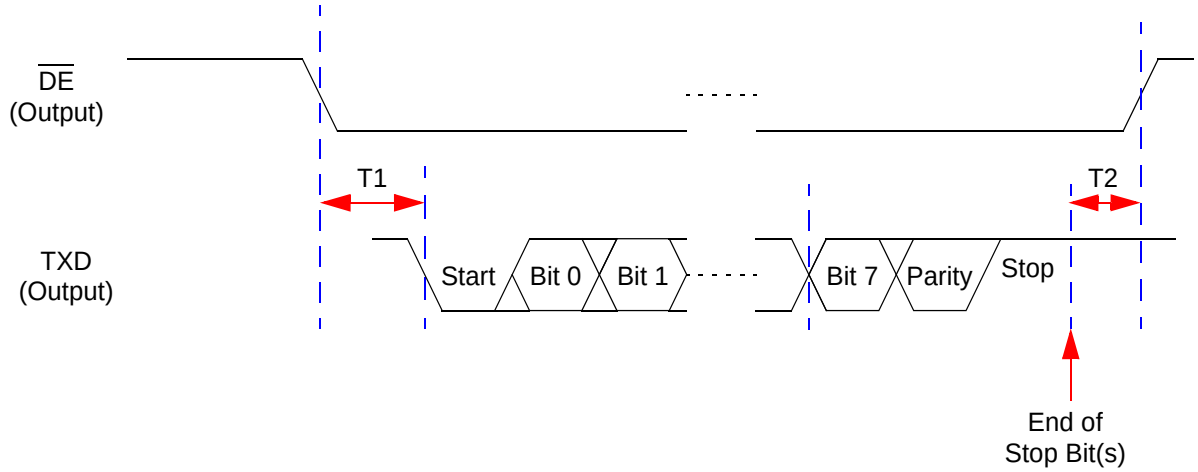


Figure 81. UART Timing without $\overline{CTS}$

Table 201. UART Timing without $\overline{CTS}$

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
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

Packaging

Zilog's ZNEO Z16F Series is comprised of the Z16FMC28, Z16FMC32 and Z16FMC64 MCUs, which are based on the ZNEO CPU and are available in the 64-pin Low-Profile Quad Flat Package (LQFP).

Current diagrams for this package are published in Zilog's Packaging Product Specification (PS0072), which is available free for download from the Zilog website.

Ordering Information

Table 202 identifies the basic features and package styles available for each device within the ZNEO product line.

Table 202. ZNEO Part Selection Guide

Part Number	Flash (KB)	RAM (KB)	External Interface	I/O	Multi-Channel Timers with PWM	Standard Timers with PWM	ADC Inputs	UARTs with LIN and IrDA	I^2C Master/Slave	ESPI	64/68-pin packages	80-pin package	100-pin package
Z16F2811	128	4	Yes	76	1	3	12	2	1	1			X
	128	4	Yes	60	1	3	12	2	1	1		X	
Z16F2810	128	4	No	60	1	3	12	2	1	1		X	
	128	4	No	46	1	3	12	2	1	1	X		
Z16F6411	64	4	Yes	76	1	3	12	2	1	1			X
	64	4	Yes	60	1	3	12	2	1	1		X	
Z16F3211	32	2	Yes	76	1	3	12	2	1	1			X
	32	2	Yes	60	1	3	12	2	1	1		X	

You can order the ZNEO Z16F Series from Zilog® by providing the part numbers listed in Table 203. For more information regarding ordering, contact your local Zilog sales office. Our website (www.zilog.com) lists all regional offices and provides additional information about ZNEO Z16F Series product.

Table 203. ZNEO Z16F Series Part Numbering

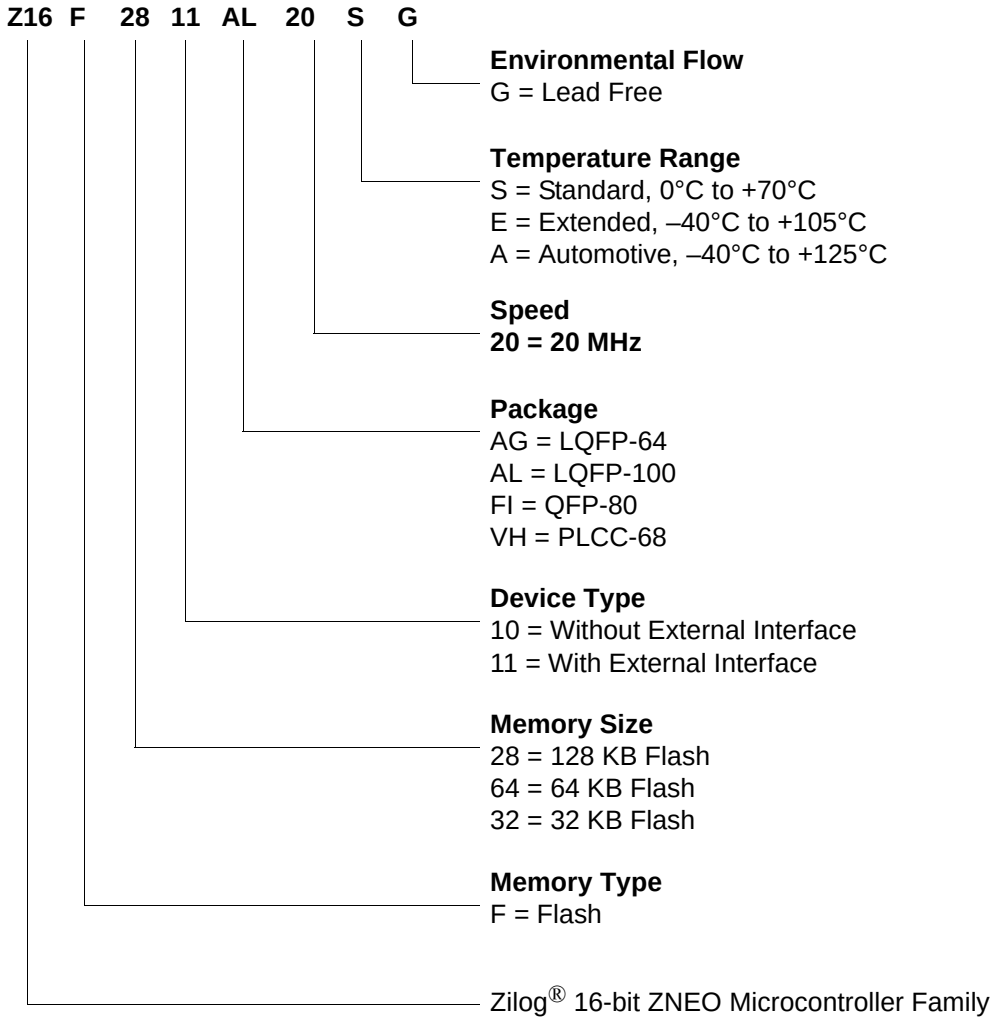
Part Number	Flash (Kbytes)	RAM (Kbytes)	External Interface	I/O	Multi-Channel timers with PWM	Standard Timers with PWM	ADC Inputs	I^2C Master/Slave	UART with LIN and IrDA	ESPI	Package
ZNEO Z16F Series											
Standard Temperature: 0°C to +70°C											
Z16F2811AL20SG	128	4	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F2811FI20SG	128	4	Yes	60	1	3	12	1	2	1	80-pin QFP
Z16F2810FI20SG	128	4	No	60	1	3	12	1	2	1	80-pin QFP
Z16F2810AG20SG	128	4	No	46	1	3	12	1	2	1	64-pin LQFP
Z16F2810VH20SG	128	4	No	46	1	3	12	1	2	1	68-pin PLCC
Z16F6411AL20SG	64	4	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F6411FI20SG	64	4	Yes	60	1	3	12	1	2	1	80-pin QFP
Z16F3211AL20SG	32	2	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F3211FI20SG	32	2	Yes	60	1	3	12	1	2	1	80-pin QFP
Extended Temperature: –40°C to +105°C											
Z16F2811AL20EG	128	4	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F2811FI20EG	128	4	Yes	60	1	3	12	1	2	1	80-pin QFP
Z16F2810FI20EG	128	4	No	60	1	3	12	1	2	1	80-pin QFP
Z16F2810AG20EG	128	4	No	46	1	3	12	1	2	1	64-pin LQFP
Z16F2810VH20EG	128	4	No	46	1	3	12	1	2	1	68-pin PLCC
Z16F6411AL20EG	64	4	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F6411FI20EG	64	4	Yes	60	1	3	12	1	2	1	80-pin QFP
Z16F3211AL20EG	32	2	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F3211FI20EG	32	2	Yes	60	1	3	12	1	2	1	80-pin QFP

Table 203. ZNEO Z16F Series Part Numbering

Part Number	Flash (Kbytes)	RAM (Kbytes)	External Interface	I/O	Multi-Channel timers with PWM	Standard Timers with PWM	ADC Inputs	I^2C Master/Slave	UART with LIN and IrDA	ESPI	Package
Automotive Temperature: –40°C to +125°C											
Z16F2811AL20AG	128	4	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F2811FI20AG	128	4	Yes	60	1	3	12	1	2	1	80-pin QFP
Z16F2810FI20AG	128	4	No	60	1	3	12	1	2	1	80-pin QFP
Z16F2810AG20AG	128	4	No	46	1	3	12	1	2	1	64-pin LQFP
Z16F2810VH20AG	128	4	No	46	1	3	12	1	2	1	68-pin PLCC
Z16F6411AL20AG	64	4	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F6411FI20AG	64	4	Yes	60	1	3	12	1	2	1	80-pin QFP
Z16F3211AL20AG	32	2	Yes	76	1	3	12	1	2	1	100-pin LQFP
Z16F3211FI20AG	32	2	Yes	60	1	3	12	1	2	1	80-pin QFP

ZNEO Z16F Series Development Tools	
Z16F2800100ZCOG	ZNEO® Z16F Series Development Kit
ZUSBSC00100ZACG	USB Smart Cable Accessory Kit
ZUSBOPTSC01ZACG	Opto-Isolated USB Smart Cable Accessory Kit
ZENETSC0100ZACG	Ethernet Smart Cable Accessory Kit

Part Number Suffix Designations



Note: Packages are not available for all memory sizes. See the Ordering Information section on page 356 for available packages.

Precharacterization Product

The product represented by this document is newly introduced and Zilog® has not completed the full characterization of the product. The document states what Zilog knows about this product at this time, but additional features or nonconformance with some aspects of the document might be found, either by Zilog or its customers in the course of further application and characterization work. In addition, Zilog cautions that delivery might be uncertain at times, due to start-up yield issues. For more information, please visit www.zilog.com.

Index

Numerics

10-bit ADC 4

A

absolute maximum ratings 337
AC characteristics 349
ADC
    block diagram 242
    electrical characteristics and timing 346
    overview 243
ADC Channel Register 1 (ADCCTL) 246
ADC Data High Byte Register (ADCDH) 247, 251
ADC Data Low Bit Register (ADCDL) 248, 249, 250, 251
analog block/PWM signal synchronization 245
analog block/PWM signal zynchronization 245
analog signals 14
analog-to-digital converter
    overview 243
architecture
    voltage measurements 243

B

baud rate generator, UART 150
block diagram 2
bus
    width 17
bus width
    non-volatile memory (internal) 21
    RAM (internal) 21

C

characteristics, electrical 337
clock phase (SPI) 181
comparator
    definition 251
    non-inverting/inverting input 252
    operation 252
control register
    external interface 42, 285
control register definition, UART 153
control register, I2C 229
control registers
    CPU 19
CPU
    control registers 19
CPU and peripheral overview 3
current measurement
    architecture 243
    operation 243
Customer Support 366

D

data
    width 17
data register, I2C 227
DC characteristics 339
debugger, on-chip 298
device, port availability 66
DMA
    controller 4

E

electrical characteristics 337
    ADC 346
    flash memory and timing 345
    GPIO input data sample timing 350
    watchdog timer 345
electrical noise 243
external interface 37
    control register 42, 285
    ISA-compatible mode 42
    operation 41
    signals 37, 290

external memory 17
external pin reset 59

F

flash
controller 4
option bit address space 292
program memory address 0000H 292
program memory address 0001H 294, 295
flash memory 255
arrangement 256
code protection 258
configurations 255
control register definitions 261
controller bypass 260
electrical characteristics and timing 345
flash status register 262
mass erase 260
operation 257
operation timing 257
page erase 259
page select register 265
FPS register 265
FSTAT register 262

G

general-purpose I/O 66
generator, wait state 41
GPIO 4, 66
alternate functions 67
architecture 66
input data sample timing 350
interrupts 70
port A-H alternate function sub-registers 74
port A-H data direction sub-registers 73
port A-H input data registers 71
port A-H output control sub-registers 74, 76
port A-H output data registers 72
port A-H stop mode recovery sub-registers 77, 78, 79
port availability by device 66
port input timing 350

H

HALT mode 65

I

I/O memory 17
precautions 19
I^2C 4
10-bit address read transaction 215
10-bit address transaction 212
10-bit addressed slave data transfer format 212, 220
7-bit address transaction 209, 217
7-bit address, reading a transaction 214
7-bit addressed slave data transfer format 211, 218
7-bit receive data transfer format 215, 221, 222
baud high and low byte registers 230, 232, 236, 237
C status register 228, 232
control register definitions 227
controller 203, 242
controller signals 13
interrupts 206
operation 206
SDA and SCL signals 206
stop and start conditions 208
I2CBRH register 231, 233, 236, 237
I2CBRL register 231
I2CCTL register 229
I2CDATA register 227
I2CSTAT register 228, 232
infrared encoder/decoder (IrDA) 172
interface, external 37
interrupt controller 4, 80
architecture 80
interrupt assertion types 84
interrupt vectors and priority 83
operation 82
register definitions 84
interrupt request 0 register 86
interrupt request 1 register 87
interrupt request 2 register 88
interrupt vector listing 80

interrupts
SPI 190
UART 147
introduction 1
IrDA
architecture 151, 172
block diagram 152, 172
control register definitions 175
operation 152, 172
receiving data 174
transmitting data 173
IRQ0 enable high and low bit registers 90
IRQ1 enable high and low bit registers 91
IRQ2 enable high and low bit registers 92
ISA-compatible mode 42

L

low power modes 64

M

master interrupt enable 82
master-in, slave-out and-in 178
memory
bus widths 17
external 17
I/O 17
internal 17, 18, 19
map 17
non-volatile 17, 18
parallel access 17
RAM 17, 19
random access 17, 19
memory access
quad 21
word 21
memory map 17
MISO 178
mode
ISA-compatible 42
MOSI 178
motor control measurements
ADC Control register definitions 246
interrupts 245, 246
overview 243
multiprocessor mode, UART 142

N

noise, electrical 243
non-volatile memory 17, 18
bus width 21

O

OCD
architecture 298
baud rate limits 302
block diagram 299
commands 309
timing 350
on-chip debugger 4
on-chip debugger (OCD) 298
on-chip debugger signals 15
on-chip oscillator 327
operation 245
current measurement 243
voltage measurement timing diagram 244, 245
operational amplifier
operation 252
overview 251
Operational Description 95, 114, 135, 331, 336
option bits 18
oscillator signals 15

P

parallel access
memory 17
peripheral AC and DC electrical characteristics 344
memory
internal 17
PHASE=0 timing (SPI) 182
PHASE=1 timing (SPI) 183
pin characteristics 16
port availability, device 66
port input timing (GPIO) 350

power supply signals 15
power-on and voltage brown-out 344
precautions, I/O memory 19

Q

quad mode
memory access 21

R

RAM 17, 19
bus width 21
random-access memory 17, 19
receive
7-bit data transfer format (I^2C) 215, 221, 222
IrDA data 174
receiving UART data-interrupt-driven method 140
receiving UART data-polled method 139
register 197
baud low and high byte (I^2C) 230, 232, 236, 237
baud rate high and low byte (SPI) 201
control (SPI) 194
control, I^2C 229
data, SPI 193
external interface control 42, 285
flash page select (FPS) 265
flash status (FSTAT) 262
GPIO port A-H alternate function sub-registers 75
GPIO port A-H data direction sub-registers 73
I^2C baud rate high (I2CBRH) 231, 233, 236, 237
I^2C control (I2CCTL) 229
I^2C data (I2CDATA) 227
I^2C status 228, 232
I^2C status (I2CSTAT) 228, 232
I2Cbaud rate low (I2CBRL) 231
mode, SPI 197
SPI baud rate high byte (SPIBRH) 202
SPI baud rate low byte (SPIBRL) 202
SPI control (SPICTL) 194
SPI data (SPIDATA) 193, 194
SPI status (SPISTAT) 198
status, SPI 198
UARTx baud rate high byte (UxBRH) 166
UARTx baud rate low byte (UxBRL) 167
UARTx Control 0 (UxCTL0) 160, 166
UARTx control 1 (UxCTL1) 162, 164, 165
UARTx receive data (UxRXD) 154
UARTx status 0 (UxSTAT0) 155, 156
UARTx status 1 (UxSTAT1) 158
UARTx transmit data (UxTXD) 154
watchdog timer control (WDTCTL) 334, 335
watchdog timer reload high byte (WDTH) 241
watchdog timer reload low byte (WDTL) 241
register file address map 23
registers
ADC channel 1 246
ADC data high byte 247, 251
ADC data low bit 248, 249, 250, 251
reset
and STOP mode characteristics 56
and STOP mode recovery 56
controller 4

S

SCK 178
SDA and SCL (IrDA) signals 206
serial clock 178
serial peripheral interface (SPI) 176
signal descriptions 12
SIO 4
slave data transfer formats (I^2C) 212, 220
slave select 179
SPI
architecture 176
baud rate generator 192
baud rate high and low byte register 201
clock phase 181
configured as slave 189
control register 194
control register definitions 193
data register 193
error detection 189
interrupts 190

mode fault error 189
mode register 197
multi-master operation 186
operation 178
overrun error 189, 190
signals 178
single master, multiple slave system 187
single master,single slave system 187
status register 198
timing, PHASE = 0 182
timing, PHASE=1 183
SPI controller signals 13
SPI mode (SPIMODE) 197
SPIBRH register 202
SPIBRL register 202
SPICTL register 194
SPIDATA register 193, 194
SPIMODE register 197
SPISTAT register 198
SS, SPI signal 178
STOP mode 64
STOP mode recovery
sources 60
using a GPIO port pin transition 61
using watchdog timer time-out 61
system 18
system and core resets 57
system vectors 18

T

tiing diagram, voltage measurement 245
timer signals 14
timers 4, 95
architecture 95, 114
block diagram 96, 115
capture mode 102, 103
capture/compare mode 103
compare mode 104
continuous mode 99
counter mode 99
gated mode 105
one-shot mode 97
operating mode 97
PWM mode 101
reading the timer count values 106
reload high and low byte registers 108, 123
timer control register definitions 106, 121
triggered one-shot mode 98
timers 0-3
control registers 109, 111
high and low byte registers 106, 109, 122, 124
timing diagram, voltage measurement 244
transmit
IrDA data 173
transmitting UART data-polled method 137

U

UART 4
architecture 135
asynchronous data format without/with parity 137
baud rate generator 150
baud rates table 169
control register definitions 153
controller signals 14
data format 136
interrupts 147
multiprocessor mode 142
receiving data using interrupt-driven method 140
receiving data using the polled method 139
transmitting data using the polled method 137
x baud rate high and low registers 166
x control 0 and control 1 registers 160, 162
x status 0 and status 1 registers 155, 158
UxBRH register 166
UxBRL register 167
UxCTL0 register 160, 166
UxCTL1 register 162, 164, 165
UxRXD register 154
UxSTAT0 register 155, 156
UxSTAT1 register 158
UxTXD register 154

V

vectors 18
    interrupts 18
    system exceptions 18
voltage brownout reset (VBR) 58
voltage measurement timing diagram 244, 245

W

wait state generator 41
watch-dog timer
    approximate time-out delays 336
    control register 333, 335
    interrupt in STOP mode 239
    operation 336
    refresh 239
    reload unlock sequence 240
    reload upper, high and low registers 241
    reset 59
    reset in normal operation 240
    reset in STOP mode 240
    time-out response 239
watchdog timer
    approximate time-out delay 239
    electrical characteristics and timing 345
    interrupt in normal operation 239
WDTCTL register 334, 335
WDTH register 241
WDTL register 241
word mode
    memory access 21

Z

ZNEO
    block diagram 2
    introduction 1
ZNEO CPU features 3

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 or consider participating in the Zilog 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.