乐升半导体
LEVETOP

LT165

Uart TFT LCD Display Controller

Specification

V1.0A

www.levetop.cn
Levetop Semiconductor Co., Ltd.

Version History

Version	Date	Description
V1.0	2025/07/08	● Priliminary Release
V1.0A	2025/07/17	● Schematic Modification: Serial port upgrade UART0 Change to use UART1； added reservation RTP (Resistive Touch Panel) circuit

Copyright

Contents

Version History 2
Copyright 2
Contents 3
Figure List 16
Table List 24
1. LT165 Introduction 27
1.1. Introduction 27
1.2. Internal Block Diagram 28
1.3. Features 29
1.3.1. 32bits RISC Core 29
1.3.2. 32K Bytes SRAM 29
1.3.3. 6K Bytes ROM 29
1.3.4. 2K Byte Cache 29
1.3.5. External Bus Interface (EBI) 29
1.3.6. DMA Module 29
1.3.7. RESET Module 30
1.3.8. Programmable Interrupt Timer (PIT) 30
1.3.9. Watch Dog Timer (WDT) 30
1.3.10. Real Time Clock (RTC) 30
1.3.11. EPORT 31
1.3.12. SPI Module 31
1.3.13. SSI / QSPI Module 31
1.3.14. SCI / UART Module 31
1.3.15. CANBus Controller 32
1.3.16. PWM Module 33
1.3.17. ADC Module 33
1.3.18. I2C Module 33
1.3.19. Analog Comparator 34
1.3.20. Touch Sensor 34
1.3.21. Power Management Unit (PMU) 34
1.3.22. Voltage Detector 34
1.3.23. Internal Oscillator 34
1.3.24. External Crystal Oscillator 34
1.4. System Block Diagram 35
1.5. System Memory Map 35

1.5.1. Introduction 35
1.5.2. Memory Address Map 36
2. Signal Description 38
2.1. Pin Assignment 38
2.2. Signal Properties Summary 40
2.3. Signal Description 42
2.4. LT165A vs. LT165B 49
3. Hardware Interface 50
3.1. Host Communication Interface 50
3.2. TFT LCD Panel Interface 50
3.3. QSPI Interface 51
3.4. LCD Touch Panel Interface 51
3.5. Clock Interface 52
3.6. Can Bus Interface 52
3.7. LCD Backlight Control Circuit 53
3.8. Real Time Clock (RTC) 54
3.9. Reset Interface 54
4. Embedded Interrupt Controller (EIC) 55
4.1. Introduction 55
4.2. Features 55
4.3. Memory Map and Registers 56
4.3.1. Memory Map 56
4.3.2. Registers 57
4.4. Function Description 63
4.4.1. Interrupt Handling Without Confliction 63
4.4.2. Interrupt With Confliction 64
4.4.3. Pend Trap Function 65
4.5. Interrupts 65
5. RISC Core Introduction 69
5.1. Features 69
5.2. Microarchitecture Summary 69
5.3. Programming Model 70
5.4. Data Format Summary 71
5.5. Operand Addressing Capabilities 72
5.6. Instruction Set Overview 72
6. Embedded Programmable Timer (EPT) 75

6.1. Introduction 75
6.2. Memory Map and Registers 75
6.2.1. Memory Map 75
6.2.2. Registers 76
6.3. Function Description 79
6.3.1. Count Timing 79
7. System Integration Module (SIM) 80
7.1. Introduction 80
7.2. Features 80
7.3. Modes of Operation 81
7.3.1. Normal Mode 81
7.3.2. Debug Mode 81
7.4. Memory Map 81
7.5. Register Descriptions 82
7.5.1. Pad Configuration Registers (SIM_PCR) 82
7.5.2. GPIO Pin Data Output Registers (SIM_GPDO0_3 ~ SIM_GPDO28_31) 86
7.5.3. GPIO Pin Data Input Registers (SIM_GPDI0_3 ~ SIM_GPDI_28_31) 90
7.5.4. Parallel GPIO Pin Data Output Register (SIM_PGPDO0) 95
7.5.5. Parallel GPIO Pin Data Input Register (SIM_PGPDI0) 96
7.5.6. Masked Parallel GPIO Pin Data Output Register (SIM_MPGPDO0 - SIM_MPGPDO1) 97
7.5.7. WKUPC — Wakeup Configuration Register 99
7.5.8. QSPIXIPMCFR — QSPI XIP Mode Configuration Register 101
7.5.9. QSPILKEYR — QSPI 32bit Key Register 102
7.6. Functional Description 102
7.6.1. Pad Configuration 102
7.6.2. GPIO Operation 102
8. Clock and Power Control Module (CLKM) 103
8.1. Overview 103
8.2. Features 103
8.3. Clock Structure 104
8.4. Clock Source Select 104
8.4.1. Low-Power Options 104
8.5. Memory Map and Registers 105
8.5.1. Module Memory Map 105
8.5.2. Register Description 106
8.6. Functional Description 121

8.6.1. Turning On PLL....121
8.6.2. The Frequency of PLL Measurement....121
8.6.3. The Frequency of 128KHz Measurement....122
9. Reset Controller Module (RCM)....123
9.1. Overview....123
9.2. Features....123
9.3. Block Diagram....123
9.4. Memory Map and Registers....124
9.4.1. Reset Test Register....124
9.4.2. Reset Status Register....125
9.4.3. Reset Control Register....126
9.5. Functional Description....126
9.5.1. Reset Sources....126
9.5.2. Reset Control Flow....128
10. Static Random Access Memory (SRAM)....129
10.1. Introduction....129
10.2. Modes of Operation....129
10.3. Low-Power Modes....129
10.4. Reset Operation....129
10.5. Interrupts....129
11. Cache Module (CACHEM)....130
11.1. Introduction....130
11.2. Block Diagram....131
11.3. Memory Map/Register Definition....132
11.4. Register Description....132
11.4.1. Cache Control Register (LMEM_CCR)....132
11.4.2. Cache Line Control Register (LMEM_CLCR)....133
11.4.3. Cache Search Address Register (LMEM_CSAR)....135
11.4.4. Cache Read/Write Value Register (LMEM_CCVR)....136
11.4.5. Cache Access Register(LMEM_ACRG)....137
11.4.6. Cache Page Invalidation Base Address Register (LMEM_PAGE_INV_BADDR)....138
11.4.7. Cache Page Invalidation Base Size Register (LMEM_PAGE_INV_SIZE)....139
11.4.8. Cache Clock Enable Register (LMEM_CACHE_CLK_EN)....140
11.5. Cache Function....140
11.6. Cache Control....142
11.6.1. Cache Set Commands....142
11.6.2. Cache Line Commands....143

12. Crossbar Switch (XBAR) 145
12.1. Introduction 145
12.1.1. Overview 145
12.1.2. Features 145
12.2. Memory Map and Registers 146
12.2.1. Register Summary 146
12.2.2. XBAR Register Descriptions 147
12.3. Function 151
12.3.1. General Operation 151
12.3.2. Register Coherency 152
12.3.3. Arbitration 152
12.3.4. Priority Assignment 153
12.4. Initialization/Application information 153
13. Direct Memory Access (DMA) 154
13.1. Information Specific to This Device 154
13.1.1. Device-Specific Features 154
13.1.2. Channel Assignments 155
13.2. Introduction 156
13.2.1. Features 157
14. Option Byte (OPB) 158
14.1. Register Memory Map 158
14.1.1. Register Descriptions 158
15. External Bus Interface (EBI) 168
15.1. Introduction 168
15.2. Features 168
15.3. Signal Description 168
15.4. Module Memory Map 168
15.5. Register Descriptions 169
15.5.1. EBI Control Registers 0 (EBICR0) 169
15.5.2. EBI Control Registers 1 (EBICR1) 171
15.6. Functional Description 172
15.6.1. EBI_WR#, EBI_RD# Signal Timing 172
15.6.2. EBI Functional Description 173
16. Programmable Interrupt Timer Modules (PIT) 175
16.1. Introduction 175
16.2. Block Diagram 175

16.3. Modes of Operation 175
16.3.1. Wait Mode 175
16.3.2. Doze Mode 175
16.3.3. Stop Mode 175
16.3.4. Debug Mode 175
16.4. Signals 176
16.5. Memory Map and Registers 176
16.5.1. Memory Map 176
16.5.2. Registers 176
16.6. Functional Description 180
16.6.1. Set-and-Forget Timer Operation 180
16.6.2. Free-Running Timer Operation 180
16.6.3. Timeout Specifications 181
16.7. Interrupt Operation 181
17. Watchdog Timer Module (WDT) 182
17.1. Introduction 182
17.2. Modes of Operation 182
17.2.1. Wait Mode 182
17.2.2. Doze Mode 182
17.2.3. Stop Mode 182
17.2.4. Debug Mode 182
17.3. 17.3 Block Diagram 183
17.4. Signals 183
17.5. Memory Map and Registers 183
17.5.1. Memory Map 183
17.5.2. Registers 184
18. Real Time Controller (RTC) 188
18.1. Introduction 188
18.2. Features 188
18.3. Test Mode 188
18.4. Block Diagram 188
19. Edge Port Module (EPORT) 189
19.1. Introduction 189
19.2. Low-Power Mode Operation 189
19.2.1. Wait and Doze Modes 189
19.2.2. Stop Mode 189
19.3. Interrupt/General-Purpose I/O Pin Descriptions 190

19.4. 19.4 Memory Map and Registers 190
19.4.1. Memory Map 190
19.4.2. Registers 190
20. CANBus Controller (CANBC) 197
20.1. Introduction 197
20.1.1. Overview 198
20.1.2. CANBus Module Features 198
20.1.3. Modes of Operation 199
20.2. External Signal Description 200
20.2.1. Overview 200
20.2.2. Signal Descriptions 200
21. Serial Communication Interface Module (SCI) 201
21.1. Introduction 201
21.2. Features 201
21.3. Modes of Operation 202
21.4. Block Diagram 202
21.5. Modes of Operation 203
21.5.1. Stop Mode 203
21.5.2. Wait Mode 203
21.6. Signal Description 203
21.7. Memory Map and Registers 204
21.7.1. SCI Version ID Register 204
21.7.2. SCI Parameter Register 205
21.7.3. SCI Reset Register 205
21.7.4. SCI Pin Register 206
21.7.5. SCI Baud Rate Register 207
21.7.6. SCI Status Register 209
21.7.7. SCI Control Register 213
21.7.8. SCI Data Register 217
21.7.9. SCI Match Address Register 218
21.7.10. SCI Modem IrDA Register 219
21.7.11. SCI FIFO Register 221
21.7.12. SCI Watermark Register 223
21.7.13. SCI Oversampling Ratio Register 224
21.8. Functional Description 225
21.9. Baud Rate Generation 225
21.10. Transmitter Functional Description 225

21.10.1. Send Break And Queued Idle 226
21.11. Receiver Functional Description 227
21.11.1. Data Sampling Technique 227
21.11.2. Receiver Wakeup Operation 228
21.11.3. Infrared Decoder 230
21.12. Additional SCI Functions 231
21.12.1. 8bit, 9bit and 10bit Data Modes 231
21.12.2. Idle Length 231
21.12.3. Single-wire Operation 232
21.12.4. Loop Mode 232
21.13. Infrared Interface 232
21.13.1. Infrared Transmit Encoder 232
21.13.2. Infrared Receive Decoder 233
21.14. Interrupts and Status Flags 233
22. Synchronous Serial Interface (SSI) 234
22.1. Introduction 234
22.2. Features 234
22.3. Modes of Operation 234
22.4. Block Diagram 235
22.5. Module Memory Map 236
22.6. Register Descriptions 237
22.6.1. Control Register 0(CTRLR0) 237
22.6.2. Control Register 1(CTRLR1) 240
22.6.3. SSI Enable Register(SSIENR) 241
22.6.4. Microwire Control Register(MWCR) 242
22.6.5. Slave Enable Register(SER) 243
22.6.6. Baud Rate Select(BAUDR) 244
22.6.7. Transmit FIFO Threshold Level(TXFTLR) 245
22.6.8. Receive FIFO Threshold Level(RXFTLR) 246
22.6.9. Transmit FIFO Level Register(TXFLR) 247
22.6.10. Receive FIFO Level Register(RXFLR) 247
22.6.11. Status Register(SR) 248
22.6.12. Interrupt Mask Register(IMR) 249
22.6.13. Interrupt Status Register(ISR) 250
22.6.14. Raw Interrupt Status Register(RISR) 251
22.6.15. Transmit FIFO Overflow Interrupt Clear Registers.(TXOICR) 252
22.6.16. Receive FIFO Overflow Interrupt Clear Register(RXOICR) 253
22.6.17. Receive FIFO Underflow Interrupt Clear Register(RXUICR) 253

22.6.18. Interrupt Clear Register(ICR) 254
22.6.19. DMA Control Register(DMACR) 255
22.6.20. DMA Transmit Data Level(DMATDLR) 256
22.6.21. DMA Receive Data Level(DMARDLR) 257
22.6.22. Identification Register(IDR) 258
22.6.23. Version ID Register(VIDR) 258
22.6.24. SSI Data Register(DRx) 259
22.6.25. RX Sample Delay Register (RXSDR) 260
22.6.26. SPI Control Register 0(SPICTRLR0) 261
22.6.27. XIP Mode Bits(XIPMBR) 262
22.6.28. XIP Incr Inst Register(XIPIIR) 263
22.6.29. XIP Wrap Inst Register(XIPWIR) 264
22.6.30. XIP Control Register(XIPCR) 265
22.6.31. XIP Slave Enable Register(XIPSER) 267
22.6.32. XIP Receive FIFO Overflow Interrupt Clear Register(XRXIOCR) 268
22.6.33. XIP Continues Transfer Time Out Register(XIPCTTOR) 269
22.7. Functional Description 270
22.7.1. Master Mode 270
22.7.2. Clock Ratios 270
22.7.3. Receive and Transmit FIFO Buffers 271
22.7.4. DMA Operation 271
22.7.5. SSI Interrupts 271
22.7.6. Enhanced SPI Modes 272
22.7.7. Execute In Place (XIP) Mode 273
22.7.8. Continuous Transfer Mode in XIP 273
22.7.9. Data Pre-fetch in XIP Operations 274
23. Serial Peripheral Interface Module (SPI) 275
23.1. Introduction 275
23.2. Features 275
23.3. Modes of Operation 275
23.4. Block Diagram 276
23.5. Signal Description 276
23.5.1. MISO (Master In/Slave Out) 277
23.5.2. MOSI (Master Out/Slave In) 277
23.5.3. SCK (Serial Clock) 277
23.5.4. SS (Slave Select) 277
23.6. Memory Map and Registers 278
23.6.1. SPI Control Register 279

23.6.2. SPI Control Register 2 280
23.6.3. SPI Baud Rate Register 281
23.6.4. SPI Frame Register 283
23.6.5. SPI RX FIFO Control Register 284
23.6.6. SPI TX FIFO Control Register 284
23.6.7. SPI RX FIFO TimeOut Counter Register 285
23.6.8. SPI TX FIFO TimeOut Counter Register 286
23.6.9. SPI Port Data Direction Register 286
23.6.10. SPI Pullup and Reduced Drive Register 287
23.6.11. SPI After SCK Delay Register 288
23.6.12. SPI Before SCK Delay Register 288
23.6.13. SPI Port Data Register 289
23.6.14. SPI Transmit Counter Register 289
23.6.15. SPI Data Register 290
23.6.16. SPI Status Register 290
23.6.17. SPI RX FIFO Status Register 292
23.6.18. SPI TX FIFO Status Register 293
23.6.19. SPI DMA Control Register 293
23.6.20. SPI DMA Threshold Register 294
23.6.21. SPI FIFO Debug Control Register 294
23.6.22. SPI Interrupt Control Register 295
23.6.23. SPI RX FIFO Debug Register 295
23.6.24. SPI TX FIFO Debug Register 296
23.7. Functional Description 296
23.7.1. Master Mode 297
23.7.2. Slave Mode 297
23.7.3. FIFO Operation 298
23.7.4. Transmission Formats 298
23.7.5. SPI Baud Rate Generation 302
23.7.6. Slave-Select Output 302
23.7.7. Bidirectional Mode 303
23.7.8. DMA Operation 303
23.7.9. High Speed Mode 304
23.7.10. Low-Power Mode Options 305
23.8. Reset 305
23.9. Interrupts 306
23.9.1. Mode Fault (MODF) Flag 306
23.9.2. EOT Interrupt Flag (EOTF) 306

23.9.3. Frame Lost Interrupt Flag (FLOST) 306
23.9.4. TXFIFO Timeout Interrupt Flag (TXFTO) 306
23.9.5. TXFIFO Overflow Interrupt Flag (TXFOVF) 306
23.9.6. TXFIFO Underflow Interrupt Flag (TXFUDF) 307
23.9.7. TXFIFO Service Interrupt Flag (TXFSER) 307
23.9.8. RXFIFO Timeout Interrupt Flag (RXFTO) 307
23.9.9. RXFIFO Overflow Interrupt Flag (RXFOVF) 307
23.9.10. RXFIFO Underflow Interrupt Flag (RXFUDF) 307
23.9.11. RXFIFO Service Interrupt Flag (RXFSER) 307
24. Inter-Integrated Circuit (I2C) 308
24.1. Introduction 308
24.2. Features 308
24.3. System and Block Diagram 309
24.4. Memory Map and Registers 309
24.4.1. I2C Status Register(I2CS) 310
24.4.2. I2C Clock Prescalar Register (I2CP) 311
24.4.3. I2C Control Register (I2CC) 312
24.4.4. I2C Slave Address Register (I2CSA) 313
24.4.5. I2C Port Control Register (I2CPCR) 314
24.4.6. I2C Slave High-Speed Mode Indicator Register(I2CSHIR) 315
24.4.7. I2C Slave SDA Hold Time Register(I2CSHT) 315
24.4.8. I2C Data Register(I2CD) 316
24.4.9. I2C Port Direction Register (I2CDDR) 316
24.4.10. I2C Port Data Register (I2CPDR) 317
24.5. Functional Description 317
24.5.1. Master Mode 317
24.5.2. Slave Mode 317
24.5.3. Protocol 318
24.5.4. Arbitration Procedure 319
24.5.5. Clock Synchronization 319
24.5.6. Handshaking 319
24.5.7. Clock Stretching 319
24.5.8. High-Speed Mode operation 320
24.5.9. Software Transaction Flow Diagrams 322
25. Pulse Width Modulator (PWM) 325
25.1. Introduction 325
25.2. Features 325
25.3. Block Diagram 326

25.4. Signal Description 326
25.5. Memory Map and Registers 327
25.5.1. Memory Map 327
25.5.2. Registers 328
25.6. Functional Descriptions 348
25.6.1. PWM Double Buffering and Automatic Reload 348
25.6.2. Modulate Duty Ratio 348
25.6.3. Dead-Zone Generator 349
25.6.4. PWM Timer Start Procedure 349
25.6.5. PWM Timer Stop Procedure 349
25.6.6. Capture Start Procedure 350
25.6.7. Capture Basic Timer Operation 350
26. Comparator Modules (COMP) 351
26.1. Introduction 351
26.2. Block Diagram 351
26.3. Modes of Operation 351
26.3.1. Wait Mode 351
26.3.2. Doze Mode 352
26.3.3. Stop Mode 352
26.4. Memory Map and Registers 352
26.4.1. Memory Map 352
26.4.2. Registers 352
26.5. Function Description 356
27. Analog-to-Digital Converter (ADC) 358
27.1. Introduction 358
27.2. ADC Main Features 358
27.3. ADC Functional Description 359
27.3.1. ADC On-Off Control (ADEN, ADDIS, ADRDY) 359
27.3.2. ADC Clock 361
27.3.3. Configuring the ADC 361
27.3.4. Channel Selection (CCWi) 361
27.3.5. Programmable Sampling Time (SMP) 362
27.3.6. Single Conversion Mode (CONT=0) 362
27.3.7. Continuous Conversion Mode (CONT=1) 363
27.3.8. Starting Conversions (ADSTART) 363
27.3.9. Timings 364
27.3.10. Stopping an Ongoing Conversion (ADSTP) 364

27.4. Conversion on External Trigger and Trigger Polarity (TRIGMODE, TRIGSCR) 365
27.4.1. Discontinuous Mode (DISCEN) 366
27.4.2. Programmable Resolution (RES) - Fast Conversion Mode 366
27.4.3. End of Conversion, End Of Sampling Phase (EOC, EOSMP Flags) 367
27.4.4. End of Conversion Sequence (EOSEQ Flag) 367
27.4.5. Example Timing Diagrams (Single/Continuous Modes Hardware / Software Triggers)....367
27.5. Data Management 368
27.5.1. Data FIFO & Data Alignment (ADC_FIFO, ALIGN) 368
27.5.2. ADC Overrun (OVR, OVRMOD) 369
27.5.3. Managing a Sequence of Data Converted Without Using The DMA 369
27.5.4. Managing Converted Data Without Using The DMA Without Overrun....369
27.5.5. Managing Converted Data Using The DMA 370
27.6. Low power Features.... 370
27.6.1. Wait Mode Conversion 370
27.6.2. Auto-off Mode (AUTOFF)....370
27.7. Analog Window Watchdog 371
27.8. Temperature Sensor 371
27.9. ADC Interrupts 372
27.10. Memory Map and Registers 373
27.10.1. Memory Map....373
27.10.2. Registers....374
28. Electrical Characteristic 387
28.1. Absolute Maximum Ratings 387
28.2. DC Electrical Specification 387
28.3. E Electrostatic Discharge (ESD) Protection 388
28.4. VDD Power Up Timin 389
29. Application Circuit 390
30. Package....392
30.1. LT165A (QFN-40pin) 392
30.2. LT165B (TSSOP-30pin)....393
30.3. PCB Design for Ground Pad 394

Figure List

Figure 1-1: LT165 Package Overview 28
Figure 1-2: Internal Block Diagram 28
Figure 1-3: System Block of LT165 35
Figure 1-4: Memory Address Map 36
Figure 2-1: LT165A (QFN-40) Pin Assignment 38
Figure 2-2: LT165B (TSSOP-30) Pin Assignment 39
Figure 3-1: UART Connection Between LT165 and Host MCU 50
Figure 3-2: LT165A Connect to 8bits 8080 I/F of TFT LCD Panel 50
Figure 3-3: LT165 Connect to SPI I/F of TFT LCD Panel 51
Figure 3-4: LT165 Connect to QSPI Flash 51
Figure 3-5: LT165 Connect to Resistive Touch Panel 51
Figure 3-6: LT165 Connect to Capacitive Touch Panel 52
Figure 3-7: External 12MHz Clock Circuit 52
Figure 3-8: Canbus Circuit Example 52
Figure 3-9: TFT LCD Backlight Circuit Example 1 53
Figure 3-10: TFT LCD Backlight Circuit Example 2 53
Figure 3-11: RTC Application Circuit 54
Figure 3-12: External Reset Circuit 54
Figure 4-1: Interrupt Control Status Register (ICSR) 57
Figure 4-2: Interrupt Enable Register (IER) 58
Figure 4-3: Interrupt Pend Set Register (IPSR) 59
Figure 4-4: Interrupt Pend Clear Register (IPCR) 60
Figure 4-5: Priority Level Select Registers (PLSR0 ~ PLSR31) 61
Figure 4-6: System Priority Level Select Registers (SYSPLSR) 62
Figure 4-7: One Pulse Interrupt Without Confliction 63
Figure 4-8: Level-Sensitive Interrupt Without Confliction 64
Figure 4-9: Two Interrupts Occur at The Same Time 64
Figure 4-10: A Lower Priority Interrupt Asserted With Confliction 64
Figure 4-11: A higher Priority Interrupt Asserted With Confliction 64
Figure 5-1: Programming Model 70
Figure 5-2: Data Organization in Memory 71
Figure 6-1: EPT Control Status Register (EPTCSR) 76
Figure 6-2: EPT Reload Register (EPTRLD) 77
Figure 6-3: EPT Counter Register (EPTCNT) 78
Figure 6-4: EPT Count Timing 79
Figure 7-1: SIM Block Diagram 80
Figure 7-2: Sample PCR Map 82
Figure 7-3: GPIO Pin Data Out Register 0 ~ 3 (SIM_GPDO0 ~ SIM_GPDO3) 86
Figure 7-4: GPIO Pin Data Out Register 4 ~ 7 (SIM_GPDO4 ~ SIM_GPDO7) 87

Figure 7-5: GPIO Pin Data Out Register 8 ~ 11 (SIM_GPDO8 ~ SIM_GPDO11) 87
Figure 7-6: GPIO Pin Data Out Register 12 ~ 15 (SIM_GPDO12 ~ SIM_GPDO15) 88
Figure 7-7: GPIO Pin Data Out Register 16 ~ 19 (SIM_GPDO16 ~ SIM_GPDO19) 88
Figure 7-8: GPIO Pin Data Out Register 20 ~ 23 (SIM_GPDO20 ~ SIM_GPDO23) 89
Figure 7-9: GPIO Pin Data Out Register 24 ~ 27 (SIM_GPDO24 ~ SIM_GPDO27) 89
Figure 7-10: GPIO Pin Data Out Register 28 ~ 31 (SIM_GPDO28 ~ SIM_GPDO31) 90
Figure 7-11: GPIO Pin Data in Register 0 ~ 3 (SIM_GPDI0 ~ SIM_GPDI3) 91
Figure 7-12: GPIO Pin Data in Register 4 ~ 7 (SIM_GPDI4 ~ SIM_GPDI7) 91
Figure 7-13: GPIO Pin Data in Register 8 ~ 11 (SIM_GPDI8 ~ SIM_GPDI11) 92
Figure 7-14: GPIO Pin Data in Register 12 ~ 15 (SIM_GPDI12 ~ SIM_GPDI15) 92
Figure 7-15: GPIO Pin Data in Register 16 ~ 19 (SIM_GPDI16 ~ SIM_GPDI19) 93
Figure 7-16: GPIO Pin Data in Register 20 ~ 23 (SIM_GPDI20 ~ SIM_GPDI23) 93
Figure 7-17: GPIO Pin Data in Register 24 ~ 27 (SIM_GPDI24 ~ SIM_GPDI27) 94
Figure 7-18: GPIO Pin Data in Register 28 ~ 31 (SIM_GPDI28 ~ SIM_GPDI31) 94
Figure 7-19: Parallel GPIO Pin Data Output Register (SIM_PGPDO0) 95
Figure 7-20: Parallel GPIO Pin Data Input Register(SIM_PGPDI0) 96
Figure 7-21: Masked Parallel GPIO Pin Data Output Register (SIM_MPGPDO0) 97
Figure 7-22: Masked Parallel GPIO Pin Data Output Register (SIM_MPGPDO1) 98
Figure 7-23: WKUPC — Wakeup Configuration Register 99
Figure 7-24: QSPIXIPMCFR — QSPI XIP Mode Configuration Register 101
Figure 7-25: QSPIKEYR — QSPI 32bit Key Register 102
Figure 8-1: Clock Structure 104
Figure 8-2: Synthesizer Control Register (SYNCR) 106
Figure 8-3: Low Speed Oscillator Control and Status Register (LOSCCSR) 110
Figure 8-4: PLL Configuration and Status Register (PLLCSR) 111
Figure 8-5: Module Stop Control Register (MSCR) 113
Figure 8-6: EPT External and CLKOUT Clock Source Control Register (ECCSCR) 115
Figure 8-7: OSC Bist Test Configuration Register1(OBTCR1) 116
Figure 8-8: OSC Bist Test Configuration Register2(OBTCR2) 117
Figure 8-9: OSC Bist Test Control Register(OBTCR) 118
Figure 8-10: OSC BIST Test Counter Register(OBTCNTR) 119
Figure 8-11: OSC BIST Test Result Register(OBTRR) 120
Figure 9-1: Reset Controller Block Diagram 123
Figure 9-2: Reset Test Register(RTR) 124
Figure 9-3: Reset Status Register(RSR) 125
Figure 9-4: Reset Control Register(RCR) 126
Figure 9-5: Reset Control Flow 128
Figure 11-1: Cache Module Block Diagram 131
Figure 11-2: Cache Control Register (LMEM_CCR) 132
Figure 11-3: Cache Line Control Register (LMEM_CLCR) 133
Figure 11-4: Cache Search Address Register (LMEM_CSAR) 135

Figure 11-5: Cache Search Address Register (LMEM_CSAR) 136
Figure 11-6: Cache Access Register(LMEM_ACRG) 137
Figure 11-7: Cache Page Invalidate Base Address Register (LMEM_PAGE_INV_BADDR) 138
Figure 11-8: Cache Page Invalidate Size Register (LMEM_PAGE_INV_SIZE) 139
Figure 11-9: Cache Clock Enable Register (LMEM_CACHE_CLK_EN) 140
Figure 11-10: Cache Tag and Data Access Structure 141
Figure 11-11: Cache Set Commands 142
Figure 11-12: Cache Line Commands 143
Figure 11-13: Line Command Results 144
Figure 12-1: XBAR Device-Specific Block Diagram 145
Figure 12-2: Master Priority Register (XBAR_MPRn) 147
Figure 12-3: Slave General Purpose Control Register (XBAR_SGPCRn) 149
Figure 13-1: DMA Block Diagram 156
Figure 14-1: PVDC — Programmable Voltage Detector Configuration Register 158
Figure 14-2: CCR — Customer Configuration Register 160
Figure 14-3: EIOSCST — External or Internal High Speed Oscillator Stable Time Configuration Register 161
Figure 14-4: PLLOCKCR — PLL Lock Time Configuration Register 161
Figure 14-5: RFEVR — RESET pin Filter Enable and Value Register 162
Figure 14-6: PVDFEVR — Programmable Voltage Detector Filter Enable and Value Register 163
Figure 14-7: IOSCTC — Internal High Speed Oscillator Trimming Configuration Register 164
Figure 14-8: VREFTCR — VREF Trimming Configuration Register 165
Figure 14-9: LDO1P5TC — LDO1P2 Trimming Configuration Register 165
Figure 14-10: RTCTCR — RTC Trimming Configuration Register 166
Figure 14-11: EOSCTCR — External Oscillator Trimming Configuration Register 167
Figure 15-1: EBI Control Register 0 (EBICR0) 169
Figure 15-2: EBI Control Register 1 (EBICR1) 171
Figure 15-3: EBI_WR#, EBI_RD# Assert/Negate Timing 172
Figure 16-1: PIT Block Diagram 175
Figure 16-2: PIT Modulus Register (PMR) 177
Figure 16-3: PIT Control and Status Register (PCSR) 177
Figure 16-4: PIT Count Register (PCNTR) 179
Figure 16-5: Counter Reloading from the Modulus latch 180
Figure 16-6: Counter in Free-Running Mode 180
Figure 17-1: Watchdog Timer Block Diagram 183
Figure 17-2: Watchdog Modulus Register (WMR) 184
Figure 17-3: Watchdog Control Register (WCR) 185
Figure 17-4: Watchdog Service Register (WSR) 187
Figure 17-5: Watchdog Count Register (WCNTR) 187
Figure 18-1: RTC Block Diagram 188
Figure 19-1: EPORT Block Diagram 189
Figure 19-2: EPORT Port Interrupt Enable Register (EPIER) 191

Figure 19-3: EPORT Data Direction Register (EPDDR) 191
Figure 19-4: EPORT Pin Assignment Register (EPPAR) 192
Figure 19-5: EPORT Pin Pull-up Enable Register (EPPUE) 193
Figure 19-6: EPORT Port Flag Register (EPFR) 193
Figure 19-7: EPORT Port Pin Data Register (EPPDR) 194
Figure 19-8: EPORT Port Data Register (EPDR) 194
Figure 19-9: EPORT Port Bit Set Register (EPBSR) 194
Figure 19-10: EPORT Digital Filter Control Register (EPFC) 195
Figure 19-11: EPORT Open Drain Enable Register (EPODE) 195
Figure 19-12: EPORT Level Polarity Register (EPLPR) 196
Figure 19-13: EPORT Port Bit Clear Register (EPBCR) 196
Figure 20-1: CANBus Block Diagram 197
Figure 21-1: SCI transmitter Block Diagram 202
Figure 21-2: SCI Receiver Block Diagram 203
Figure 21-3: SCI Version ID Register(SCI_VERID) 204
Figure 21-4: SCI Parameter Register(SCI_PARAM) 205
Figure 21-5: SCI Reset Register(SCI_RESET) 205
Figure 21-6: SCI Pin Register(SCI_PIN) 206
Figure 21-7: SCI Baud Rate Register (SCI_BAUD) 207
Figure 21-8: SCI Status Register (SCI_STAT) 209
Figure 21-9: SCI Control Register (SCI_CTRL) 213
Figure 21-10: SCI Data Register (SCI_DATA) 217
Figure 21-11: SCI Match Address Register (SCI_MATCH) 218
Figure 21-12: SCI Modem IrDA Register (SCI_MODIR) 219
Figure 21-13: SCI FIFO Register (SCI_FIFO) 221
Figure 21-14: SCI Watermark Register (SCI_WATER) 223
Figure 21-15: SCI Oversampling Ratio Register (SCI_OSR) 224
Figure 21-16: SCI Baud Rate Generation 225
Figure 22-1: SSI Block Diagram 235
Figure 22-2: Control Register 0(CTRLR0) 237
Figure 22-3: Control Register 1(CTRLR1) 240
Figure 22-4: SSI Enable Register(SSIENR) 241
Figure 22-5: Microwire Control Register(MWCR) 242
Figure 22-6: Slave Enable Register(SER) 243
Figure 22-7: Baud Rate Select(BAUDR) 244
Figure 22-8: Transmit FIFO Threshold Level(TXFTLR) 245
Figure 22-9: Receive FIFO Threshold Level(RXFTLR) 246
Figure 22-10: Transmit FIFO Level Register(TXFLR) 247
Figure 22-11: Receive FIFO Level Register(RXFLR) 247
Figure 22-12: Status Register(SR) 248
Figure 22-13: Interrupt Mask Register(IMR) 249

Figure 22-14: Interrupt Status Register(ISR) 250
Figure 22-15: Raw Interrupt Status Register(RISR)....251
Figure 22-16: Transmit FIFO Overflow Interrupt Clear Registers.(TXOICR)....252
Figure 22-17: Receive FIFO Overflow Interrupt Clear Register(RXOICR) 253
Figure 22-18: Receive FIFO Underflow Interrupt Clear Register(RXUICR)....253
Figure 22-19: Interrupt Clear Register(ICR)....254
Figure 22-20: DMA Control Register(DMACR) 255
Figure 22-21: DMA Transmit Data Level(DMATDLR)....256
Figure 22-22: DMA Receive Data Level(DMARDLR) 257
Figure 22-23: Identification Register(IDR)....258
Figure 22-24: Version ID Register(VIDR)....258
Figure 22-25: SSI Data Register(DRx)....259
Figure 22-26: RX Sample Delay Register (RXSDR)....260
Figure 22-27: SPI Control Register 0(SPICTRLR0)....261
Figure 22-28: XIP Mode Bits(XIPMBR)....262
Figure 22-29: XIP Incr Inst Register(XIPIIR)....263
Figure 22-30: XIP Wrap Inst Register(XIPWIR)....264
Figure 22-31: XIP Control Register(XIPCR)....265
Figure 22-32: XIP Slave Enable Register(XIPSER)....267
Figure 22-33: XIP Receive FIFO Overflow Interrupt Clear Register(XRXIOCR) 268
Figure 22-34: XIP Continues Transfer Time Out Register(XIPCTTOR)....269
Figure 22-35: SSI Configured as Master Device....270
Figure 22-36: Typical Write Operation Dual/Quad/Octal SPI Mode....272
Figure 22-37: Typical Read Operation Dual/Quad/Octal SPI Mode 272
Figure 22-38: XIP Transfer with Instruction Phase 273
Figure 23-1: SPI Block Diagram....276
Figure 23-2: SPI Control Register 1 (SPICR1)....279
Figure 23-3: SPI Control Register 2 (SPICR2)....280
Figure 23-4: SPI Baud Rate Register (SPIBR) 281
Figure 23-5: SPI Frame Register (SPIFR) 283
Figure 23-6: SPI RX FIFO Control Register (SPIRXFCR)....284
Figure 23-7: SPI TX FIFO Control Register (SPITXFCR)....284
Figure 23-8: SPI RX FIFO TimeOut Counter Register (SPIRXFTOCTR)....285
Figure 23-9: SPI TX FIFO TimeOut Counter Register (SPITXFTOCTR)....286
Figure 23-10: SPI Port Data Direction Register (SPIDDR)....286
Figure 23-11: SPI Pullup and Reduced Drive Register (SPIPURD)....287
Figure 23-12: SPI After SCK Delay Register (SPIASCDR) 288
Figure 23-13: SPI Before SCK Delay Register (SPIBSCDR) 288
Figure 23-14: SPI Port Data Register (SPIPORT)....289
Figure 23-15: SPI Transmit Counter Register (SPITCNT)....289
Figure 23-16: SPI Data Register (SPIDR)....290

Figure 23-17: SPI Status Register (SPISR) 290
Figure 23-18: SPI RX FIFO Status Register (SPIRXFSR) 292
Figure 23-19: SPI TX FIFO Status Register (SPITXFSR) 293
Figure 23-20: SPI DMA Control Register (SPIDMACR) 293
Figure 23-21: SPI DMA Threshold Register (SPIDMATHR) 294
Figure 23-22: SPI FIFO Debug Control Register (SPIFDCR) 294
Figure 23-23: SPI Interrupt Control Register (SPIICR) 295
Figure 23-24: SPI RX FIFO Debug Register (SPIRXFDBGR) 295
Figure 23-25: SPI TX FIFO Debug Register (SPITXFDBGR) 296
Figure 23-26: Full-Duplex Operation 296
Figure 23-27: SPI Clock Format 1 (CPHA = 1) 299
Figure 23-28: SPI Clock Format 0 (CPHA = 0) 300
Figure 23-29: Transmission Error Due to Master/Slave Clock Skew 301
Figure 23-30: TI Single Transfer 302
Figure 23-31: TI Continuous Transfer 302
Figure 23-32: High Speed Mode (CPHA = 0) 304
Figure 24-1: I2C Block Diagram 309
Figure 24-2: I2C Status Register(I2CS) 310
Figure 24-3: I2C Clock Prescalar Register (I2CP) 311
Figure 24-4: I2C Control Register (I2CC) 312
Figure 24-5: I2C Slave Address Register (I2CSA) 313
Figure 24-6: I2C Port Control Register (I2CPCR) 314
Figure 24-7: I2C Slave High-Speed Mode Indicator Register(I2CSHIR) 315
Figure 24-8: I2C Slave SDA Hold Time Register(I2CSHT) 315
Figure 24-9: I2C Data Register(I2CD) 316
Figure 24-10: I2C Port Direction Register (I2CDDR) 316
Figure 24-11: I2C Port Data Register (I2CPDR) 317
Figure 24-12: I2C Communication Protocol 318
Figure 24-13: Repeat Start Of I2C Protocol 319
Figure 24-14: SCL Synchronization 320
Figure 24-15: Data Transfer Format In HS Mode 321
Figure 24-16: A Complete HS Mode Transfer 321
Figure 24-17: Slave Mode Initialization 322
Figure 24-18: Master Mode Initialization 322
Figure 24-19: Interrupt Transaction 323
Figure 25-1: PWM Block Diagram 326
Figure 25-2: PWM Pre-scale Register (PPR) 328
Figure 25-3: PWM Clock Select Register(PCSR) 329
Figure 25-4: PWM Control Register(PCR) 330
Figure 25-5: PWM Counter Register (PCNR) 333
Figure 25-6: PWM Comparator Register (PCMR) 335

Figure 25-7: PWM Comparator Register Setting Timing 336
Figure 25-8: PWM Duty 336
Figure 25-9: PWM Timer Register (PTR) 338
Figure 25-10: PWM Interrupt Enable Register (PIER) 339
Figure 25-11: PWM Interrupt Flag Register (PIFR) 340
Figure 25-12: PWM Capture Control Register (PCCR0/1) 341
Figure 25-13: PWM Capture Rising Latch Register (PCRLR0/1/2/3) 344
Figure 25-14: PWM Capture Falling Latch Register (PCFLR0/1/2/3) 346
Figure 25-15: PWM Port Control Register (PPCR) 347
Figure 25-16: PWM Double Buffering Illustration 348
Figure 25-17: PWM Controller Output Duty Ratio 348
Figure 25-18: Dead Zone Generation Operation 349
Figure 25-19: Capture Basic Timer Operation 350
Figure 26-1: Comparator Block Diagram 351
Figure 26-2: Comparator Control Register (CPTCN) 353
Figure 26-3: Comparator Mode Selection Register(CPTMD) 353
Figure 26-4: Comparator MUX Selection Register(CPTMX) 354
Figure 26-5: Comparator Output Filter Selection Register(CPTFLS) 355
Figure 26-6: Comparator Hysteresis Plot 357
Figure 27-1: ADC Block Diagram 359
Figure 27-2: Enabling/Disabling the ADC 360
Figure 27-3: ADC Clock Scheme 361
Figure 27-4: Analog to Digital Conversion Time 364
Figure 27-5: Stopping an Ongoing Conversion 365
Figure 27-6: Single Conversions of A Sequence, Software Trigger 367
Figure 27-7: Continuous Conversion of A Sequence, Software Trigger 367
Figure 27-8: Single Conversions of A Sequence, Hardware Trigger 368
Figure 27-9: Continuous Conversions of A Sequence, Hardware Trigger 368
Figure 27-10: Data Alignment And Resolution 368
Figure 27-11: Analog Watchdog Guarded Area 371
Figure 27-12: ADC Interrupt And Status Register (ADC_ISR) 374
Figure 27-13: ADC Interrupt Enable Register (ADC_IER) 376
Figure 27-14: ADC Control Register (ADC_CR) 377
Figure 27-15: ADC Configuration Register 1 (ADC_CFGR1) 379
Figure 27-16: ADC Configuration Register 2 (ADC_CFGR2) 381
Figure 27-17: ADC Sampling Time Register (ADC_SMPR) 382
Figure 27-18: ADC Watch Dog Register (ADC_WDG) 383
Figure 27-19: ADC Watchdog Threshold Register (ADC_TR) 384
Figure 27-20: ADC Channel Selection Register 1(ADC_CHSELR1) 385
Figure 27-21: ADC Channel Selection Register 2(ADC_CHSELR2) 385
Figure 27-22: ADC FIFO Access Register (ADC_FIFO) 386

Figure 28-1: VDD Power up Timing Requirement 389
Figure 29-1: AP Circuit of LT165A for 8bit 8080 Interface TFT LCD Panel 390
Figure 29-2: AP Circuit of LT165A for SPI Interface TFT LCD Panel 391
Figure 30-1: LT165A Package Overview 392
Figure 30-2: LT165B Package Overview 393
Figure 30-3: PCB Design Recommendation for the LT165A Bottom Ground Pad 394

Table List

Table 1-1: LT165 Models....27
Table 1-2: The Hardware Module and Register Address Location Map....36
Table 2-1: Signal Properties....40
Tabel 2-2: Signal Description....42
Table 2-3: LT165x Comparison....49
Table 4-1: Interrupt Controller Module Memory Map....56
Table 4-2: Priority Level Value Adjustment....61
Table 4-3: EPT Priority Value Adjustment....62
Table 4-4: Software Interrupt Priority Value Adjustment....63
Table 4-5: Interrupt Source Assignment....65
Table 5-1: RISC Core Instruction Set....72
Table 6-1: Programmable Timer Module Memory Map....75
Table 7-1: SIM Address Map....81
Table 7-2: SIM_PCR Field Descriptions....83
Table 7-3: PCR Priority for Multiple Source Input....85
Table 7-4: SIM_PCRn Settings....85
Table 7-5: SIM_GPDOx Register Field Descriptions....90
Table 7-6: SIM_GPDIx Register Field Descriptions....95
Table 7-7: SIM_PGPDO0 Field Descriptions....95
Table 7-8: SIM_PGPDI0 Field Descriptions....96
Table 7-9: SIM_MPGPDO0 ~ SIM_MPGPDO31 Field Descriptions....98
Table 7-10: WKUPSEN and the Corresponding Wakeup Source....100
Table 8-1: Clock Memory Map....105
Table 8-2: FIRC150M or FXOSC Clock Divider....106
Table 8-3: PLL Clock Divider....107
Table 8-4: TOUCH 48MHz Clock Divider....107
Table 8-5: ADC Clock Divider....108
Table 8-6: CLKOUTSEL Mode....108
Table 8-7: Sleep Operation Control Bit in Sleep Mode....109
Table 8-8: PLL Feedback Divider Value....112
Table 8-9: PLL VCO Output Clock Divider....112
Table 8-10: PLL Input Divider....112
Table 8-11: MS[29:0] Bits Corresponding Modules....114
Table 8-12: EPT Clock Divider....115
Table 8-13: CLKOUT Divider....116
Table 9-1: Reset Controller Address Map....124
Table 9-2: Reset Source Summary....126
Table 11-1: Cache Module Memory Map....132
Table 12-1: XBAR Register Configuration Summary....146

Table 12-2: XBAR Master Priority Register Field Descriptions 147
Table 12-3: XBAR Slave General Purpose Control Register Field Descriptions 149
Table 13-1: DMA Channel Assignment 155
Table 14-1: Register Memory Map 158
Table 14-2: PVDC Setting Table 159
Table 14-3: Clock Mode Setting Table 160
Table 15-1: Signal Properties 168
Table 15-2: Register Memory Map 168
Table 15-3: EBI_CS# Chip Select Wait States Encoding 170
Table 15-4: EBI_WR# and EBI_RD# Assert and Negate Timing (Unit: System Cycle) 172
Table 15-5: Chip Select Address Range Encoding 173
Table 15-6: 8bit Port Data Transfer of EBI_CS# (big endian) 174
Table 15-7: 16bit Port Data Transfer of EBI_CS# (big endian) 174
Table 16-1: Programmable Interrupt Timer Module Memory Map 176
Table 16-2: Prescaler Select Encoding 178
Table 16-3: PIT Interrupt Requests 181
Table 17-1: Watchdog Timer Module Memory Map 183
Table 17-2: Watchdog Timer Prescaler 186
Table 19-1: Module Memory Map 190
Table 19-2: EPPAx Field Settings 192
Table 20-1: CANBus Signals 200
Table 21-1: Signal Properties 203
Table 21-2: SCI Module Memory Map[1] 204
Table 21-3: Break Character Length 226
Table 21-4: Receiver Wakeup Options 228
Table 22-1: SSI Memory Map 236
Table 23-1: Signal Properties 276
Table 23-2: SPI Memory Map 278
Table 23-3: SS Pin I/O Configurations 280
Table 23-4: Bidirectional Pin Configurations 281
Table 23-5: SPI Baud Rate Selection (10-MHz Module Clock) 282
Table 23-6: SPI Baud Rate Selection (10-MHz Module Clock) 282
Table 23-7: Normal Mode and Bidirectional Mode 303
Table 23-8: SPI Interrupt Request Sources 306
Table 24-1: I2C Memory Map 309
Table 25-1: PWM Signal Description 326
Table 25-2: Module Memory Map 327
Table 25-3: PWM Clock Divider Setting 329
Table 26-1: Comparator Module Memory Map 352
Table 26-2: Comparator Mode Selection 354
Table 26-3: Comparator Negative Input MUX Selection 354

Table 26-4: Comparator Positive Input MUX Selection 355
Table 27-1: Channel Decode 362
Table 27-2: Configuring the Trigger Polarity 365
Table 27-3: External Triggers 365
Table 27-4: Analog Watchdog Channel Selection 371
Table 27-5: ADC Interrupts 372
Table 27-6: QADC Memory Map 373
Table 28-1: Absolute Maximum Rating 387
Table 28-2: IO Static Characteristic (3.3V) 387
Table 28-3: Power Characteristic 388
Table 28-4: Electrostatic Discharge Protection Characteristic 388
Table 28-5: VDD Power Up Characteristic 389
Table 30-1: LT165A Package Parameter 392
Table 30-2: LT165B Package Parameter 393

1. LT165 Introduction

1.1. Introduction

LT165 is a series of low-cost and high-performance Uart serial panel control chips, which embedded a 32bit RISC core architecture internally The main function of this chip is to provide Uart serial communication, allowing the main control MCU to easily transmit the content to be displayed on the TFT panel to the driver on the TFT panel through simple communication commands. The internal hardware and serial program of LT165 provide high-speed image processing capabilities, which can achieve excellent display efficiency and reduce the time required for the main control MCU to process graphic displays. LT165 supports 8bit parallel TFT panel with a resolution of 320 * 240 or TFT LCD panel with SPI serial ports.

The internal frequency of LT165 can reach 150MHz and contains 32KB SRAM. In addition to providing serial communication, it also provides a QSPI Flash interface for quickly reading program code, images, animations, and other information stored in external SPI Flash LT165 can be used in conjunction with the serial panel development software (UI-Editor) and simulation software (UI-Eimulator) developed by Levetop Semiconductor to directly develop the UI display interface of the product on the PC computer. The display functions it supports include image display, GIF animation display, loop image display, progress bar display, display, text string display, PWM (DMA mode) audio playback, and multi variable control display combined with touch function In addition to improving display efficiency, it also significantly shortens the development cycle of TFT LCD displays. In addition, LT165 also provides 2 sets of SCI (Uart) interfaces that can be connected to Bluetooth modules or WiFi modules, as well as CanBus, analog input AIN, PWM, INT interrupt interfaces, and capacitive touch input interfaces. These interfaces can also be used as general purpose IO interfaces and come with an RTC clock. The functions of this chip increase the practicality and applicability of the serial port TFT LCD panel, and the chip characteristics also comply with automotive standard design and application. LT165 can support dual panel display or parallel panel display applications, with good smoothness of deisplay and extremely high cost-effectiveness.

On many small electronic products, LT165 can also use internal resources as the main control MCU, and the main control and TFT display functions can be completed by one LT165. Its display function is very suitable for use in electronic products with low resolution TFT-LCD panels, such as replacing original monochrome panel products, or increasing product texture and grade, without causing too much loading on the original MCU of the product's main control terminal. It can be applied to various small electronic products such as smart home appliances, handheld control devices, industrial control boards, electronic instruments, testing equipment, small electric motorcycles, personal medical aesthetics, small testing equipment, charging equipment, water and electricity meters, smart speakers with TFT LCD panel, robot eyes and other products.

The LT165 has two different package models, as shown below:

Table 1-1: LT165 Models

Type	Package	SRAM	TFT Panel
LT165A	QFN-40	32KB	● 8bits 8080 I/F TFT Panel ● SPI I/F TFT Panel
LT165B	TSSOP-30	32KB	● SPI I/F TFT Panel

LT165A
QFN-40 (5.0*5.0 mm²)

LT165B
TSSOP-30 (7.8*6.4mm²)

Figure 1-1: LT165 Package Overview

1.2. Internal Block Diagram

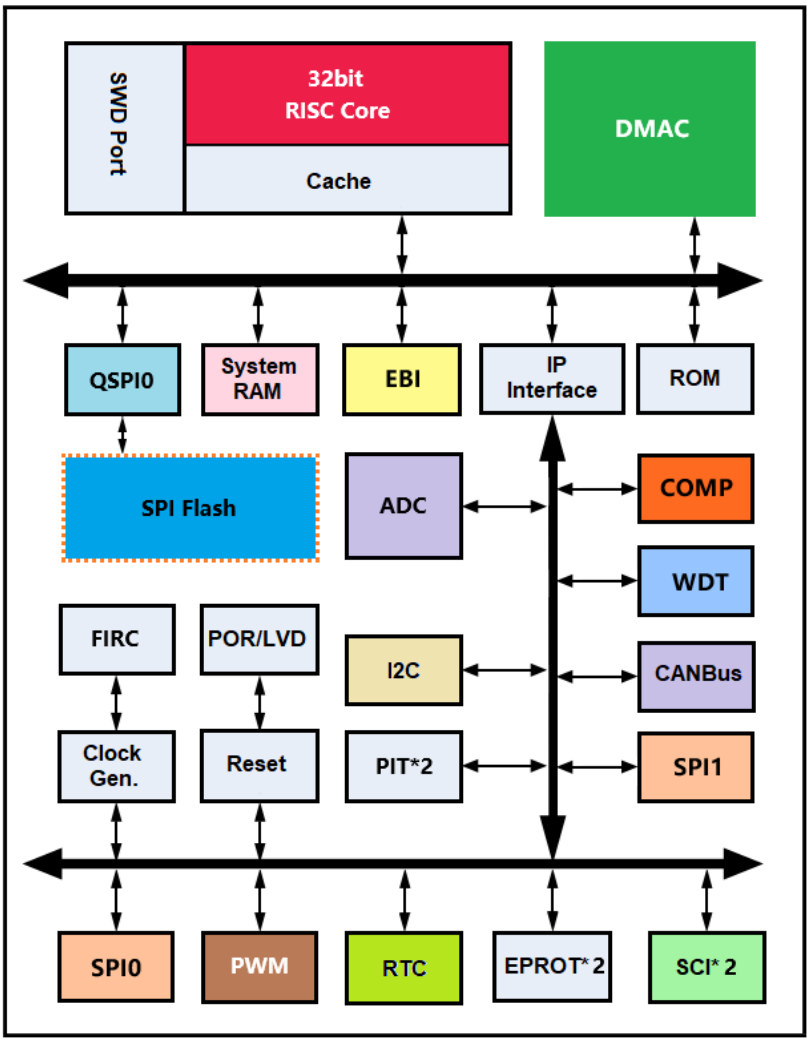


Figure 1-2: Internal Block Diagram

1.3. Features

1.3.1. 32bits RISC Core

- 32bit load/store reduced instruction set computer (RISC) architecture with fixed 16bit instruction length
- 16 entry 32bit general-purpose register file
- Efficient 3-stage execution pipeline, hidden from application software
- Single-cycle instruction execution for many Instructions, three cycles for branches
- Support for byte/halfword/word memory accesses
- Embedded interrupt controller, support nested vector interrupts.
- Single-cycle 32bit x 32bit hardware integer multiplier array
- 3~13 cycles hardware integer divider array

1.3.2. 32K Bytes SRAM

- Single cycle byte, half-word (16bit), and word (32bit) reads and writes
- Two segments for improving performance at certain application
  - System RAM0: 16Kbytes and address range from 0x0080_0000 to 0x0080_3FFF
  - System RAM1: 16Kbytes and address range from 0x0080_4000 to 0x0080_7FFF

1.3.3. 6K Bytes ROM

- Single cycle byte, half-word (16bit), and word (32bit) reads access

1.3.4. 2K Byte Cache

- 2-way set -associative organization
- Two AHB bus interfaces, a master and a slave interface

1.3.5. External Bus Interface (EBI)

- Programmable wait states -up to 256 wait states can be programmed before the access terminated
- One programmable asynchronous active-low chip selects.
- Programmable chip selects wait cycle
- To interface with various panels, up to 256 chip selects asserted cycle can be programmed.
- Programmable read/write asserted cycle
- Programmable read/write negated cycle
- Support 8bits port size.
- Support 8080 standard bus

1.3.6. DMA Module

- 16 programmable channels to support independent 8, 16 or 32bit single value or block transfers
- Support of variable sized queues and circular queues
- Source and destination address registers independently configured to post-incrementor remain constant
- Each transfer initiated by peripheral, CPU, periodic timer interrupt or DMA channel request
- Peripheral DMA request sources possible from QSPI, QADC
- Each DMA channel able to optionally send interrupt request to CPU on completion of single value or block transfer
- DMA transfers possible between system memories and all accessible memory mapped locations

including peripheral and registers

- DMA supports the following functionality:
  - Scatter Gather
  - Channel Linking
  - Inner Loop Offset
  - Arbitration
  - Fixed Group, fixed channel
  - Round Robin Group, fixed channel
  - Round Robin Group, Round Robin Channel
  - Fixed Group, Round Robin Channel
  - Channel preemption
  - Cancel channel transfer
- Interrupts – The DMA has a single interrupt request for each implemented channel and a combined DMA Error interrupt to flag transfer errors to the system

1.3.7. RESET Module

- Internal power on reset circuit
- Five sources of reset:
  - Power-on reset
  - External pin
  - Software reset
  - Watchdog timer
  - Program Voltage Detect Reset
- Status flag indicates source of last reset

1.3.8. Programmable Interrupt Timer (PIT)

- 16bit counter with modulus "initial count" register
- Selectable as free running or count down
- 16 selectable prescalers — 2^0 to 2^{15}
- support DMA interface

1.3.9. Watch Dog Timer (WDT)

- 16bit counter with modulus "initial count" register
- Pause option for low-power modes
- Up to 2000ms service time

1.3.10. Real Time Clock (RTC)

- Support loading time data to and read time data from seconds, minutes, hours and days counters
- Support alarm settings
- Interrupt sources:
  - second, minute, hour, day interrupts
  - programmable alarm interrupts
  - 1KHz/32KHz periodic interrupts

1.3.11. EPORT

- Eight Channels for each EPORT
- Rising/falling edge select
- Low/High level sensitive
- Interrupt pins configurable as general-purpose I/O

1.3.12. SPI Module

- Master mode and slave mode
- Wired-OR mode
- Slave-select output
- Mode fault error flag with central processor unit (CPU) interrupt capability
- Double-buffered operation
- Serial clock with programmable polarity and phase
- Control of SPI operation during doze mode
- Reduced drive control for lower power consumption

1.3.13. SSI / QSPI Module

- Serial-master operation
- DMA controller interface
- Enables the SSI to interface to a DMA controller over the bus using handshaking interface for transfer requests.
- Clock stretching support in enhanced SPI transfers
- Data item size (4 to 32bits) – Item size of each data transfer under control of the programmer
- Configurable depth of the transmit and receive FIFO buffers from 2 to 256 words deep. The FIFO width is fixed at 32bits
- Enhanced SPI support
- Execute in Place (XIP) mode support

1.3.14. SCI / UART Module

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13bit modulo divider) with configurable oversampling ratio from 4x to 256x
- Interrupt, polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
  - Receive data match
- Hardware parity generation and checking
- Programmable 8bit, 9bit or 10bit character length
- Programmable 1bit or 2bit stop bits
- Three receiver wakeup methods:
  - Idle line wakeup

  - Address mark wakeup
  - Receive data match
- Automatic address matching to reduce ISR overhead:
  - Address mark matching
  - Idle line address matching
  - Address match start, address match end
- Optional 13bit break character generation / 11bit break character detection
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64 or 128 idle characters
- Selectable transmitter output and receiver input polarity
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse width
- Independent FIFO structure for transmit and receive
  - Separate configurable watermark for receive and transmit requests
  - Option for receiver to assert request after a configurable number of idle characters if receive FIFO is not empty

1.3.15. CANBus Controller

- Full implementation of the CAN protocol specification, version 2.0B
  - Standard data and remote frames
  - Extended data and remote frames
  - 0 ~ 8 bytes data length
  - Programmable bit rate up to 1 Mbit/s
  - Content-related addressing
- 16 Message Buffers of zero to eight bytes data length
- Each MB configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Message Buffer
- Includes 288bytes (16 MBs) of SRAM used for MB storage
- Includes 64 bytes (16 MBs) of SRAM used for individual Rx Mask Registers
- Full featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 8 extended, 16 standard or 32 partial (8bits) IDs, with individual masking capability
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused MB and Rx Mask Register space can be used as general purpose SRAM space
- Listen-only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power mode
- Hardware cancellation on Tx message buffers

1.3.16. PWM Module

- Four channel each PWM controller
- Programmable period
- Programmable duty cycle
- Two Dead-Zone generator
- Capture function
- Pins can be configured as general-purpose I/O

1.3.17. ADC Module

- High performance
  - 12bit, 10bit, 8bit or 6bit configurable resolution
  - ADC conversion time: 1.0 μs for 12bit resolution (1 MHz), 0.88 μs conversion time for 10bit resolution, faster conversion times can be obtained by lowering resolution.
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support
- Low power
  - Application can reduce PLCK frequency for low power operation while still keeping optimum ADC performance. For example, 1.0 μs conversion time is kept, whatever the frequency of PCLK.
  - Wait mode: prevents ADC overrun in applications with low frequency PLCK
  - Auto off mode: ADC is automatically powered off except during the active conversion phase. This dramatically reduces the power consumption of the ADC.
- Analog input channels
  - 3 external analog inputs
  - 1 channel for internal temperature sensor
- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity
- Conversion modes
  - Can convert a single channel or can scan a sequence of channels. Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events.
- Analog watchdog
- Single-ended and differential-input configurations

➢ Converter uses an internal reference or an external reference

1.3.18. I2C Module

- Supports 7bit addressing.
- Supports Standard Mode, Fast Mode and High-Speed Mode
- Software option to select between High-Speed mode and Standard/Fast mode
- Compatibility with standard and fast-mode of I2C bus version 2.1 standard.
- Multiple-master operation.

- Software-programmable for one of 64 different serial clock frequencies.
- Software-selectable acknowledge bit.
- Interrupt-driven, byte-by-byte data transfer.
- Arbitration-lost interrupt with automatic mode switching from master to slave.
- Transfer completion and read configure interrupt.
- Start and stop signal generation/detection.
- Repeated START signal generation.LT32A05_SPEC_ENG/V0.0
- Acknowledge bit generation/detection.
- Bus-busy detection.
- Option slave address receiving enable when system clock stop mode
- SCL or SDA line gpio function supported

1.3.19. Analog Comparator

- Programmable response time
- Programmable hysteresis
- Support analog input multiplexer with nine selections
- Two optional outputs: filtered or asynchronous output
- Selectable rising/falling edge interrupt

1.3.20. Touch Sensor

- Support four touch keys
- Support three clock mode with charge or discharge function
  - Frequency range from 369KHz to 6MHz with Fixed clock divider
  - Frequency with PRS 1.5MHz follow Normal Distribution
  - Frequency with PRS 1.5MHz follow even Distribution
- Programmable counter clock frequency with 24/12/6/4MHz
- Programmable counter width range from 9 to 16bits
- Support synchronous scan mode

1.3.21. Power Management Unit (PMU)

- Support on-chip 1.2V LDO with maximum load current 150mA
- 1.2V LDO support two mode: lower power, high power

1.3.22. Voltage Detector

- Programmable voltage detector

1.3.23. Internal Oscillator

- 128KHz on-chip oscillator clock for watchdog and PMU
- Fast Internal RC clock which can be used for system clock

1.3.24. External Crystal Oscillator

- 32.768KHz external crystal Oscillator clock which can be used for RTC
- Fast external crystal Oscillator clock which can be used for system clock

1.4. System Block Diagram

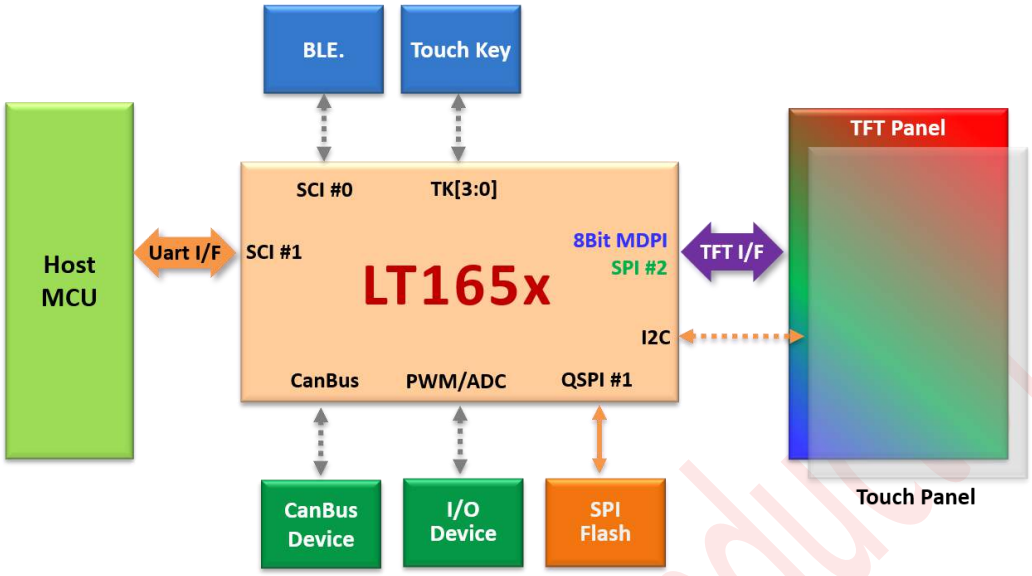


Figure 1-3: System Block of LT165

1.5. System Memory Map

1.5.1. Introduction

The built-in memory, registers, and external memory of LT165 include:

- Up to 128M Bytes of External QSPI Flash
- 8K Bytes Internal Boot ROM
- 32K Bytes Internal Static SRAM
- Internal Memory Mapped Registers

1.5.2. Memory Address Map

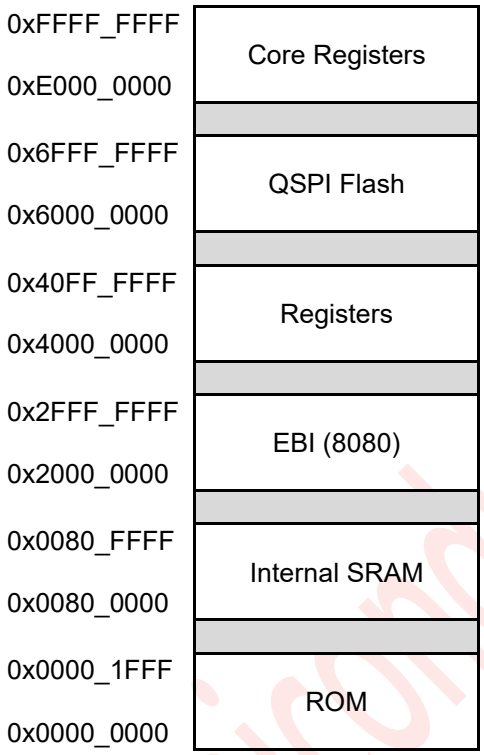


Figure 1-4: Memory Address Map

Table 1-2: The Hardware Module and Register Address Location Map

Address	Maximum Size	Hardware Module	Chapter
0x4000_0000	64Kbyte	Direct Memory Access Controller (DMAC)	13
0x4001_0000	64Kbyte	System Integration Module (SIM)	7
0x4002_0000	64Kbyte	Reset Control Module (RCM)	9
0x4003_0000	64Kbyte	Clock Control Module (CLKM)	8
0x4004_0000	64Kbyte	Programmable Interrupt Timer 0 (PIT0)	16
0x4005_0000	64Kbyte	Programmable Interrupt Timer 1 (PIT1)	16
0x4006_0000	64Kbyte	Reserved	-
0x4007_0000	64Kbyte	Reserved	-
0x4008_0000	64Kbyte	Serial Communication Interface 1 (SCI1)	21
0x4009_0000	64Kbyte	Serial Communication Interface 0 (SCI0)	21
0x400A_0000	64Kbyte	Analog Comparator 0 (COMP0)	26
0x400B_0000	64Kbyte	Reserved	-
0x400C_0000	64Kbyte	Reserved	-
0x400D_0000	64Kbyte	Pulse Width Modulator 0 (PWM0)	25
0x400E_0000	64Kbyte	Reserved	-

Address	Maximum Size	Hardware Module	Chapter
0x400F_0000	64Kbyte	Edge Port Module 0 (EPORT0)	19
0x4010_0000	64Kbyte	Edge Port Module 1 (EPORT1)	19
0x4011_0000	64Kbyte	Analog-to-Digital Convertor (ADC)	27
0x4012_0000	64Kbyte	Option Byte (OPB)	14
0x4013_0000	64Kbyte	WatchDog Timer (WDT)	17
0x4014_0000	64Kbyte	Real Time Controller (RTC)	18
0x4015_0000	64Kbyte	Inter-Integrated Circuit (I2C)	24
0x4016_0000	64Kbyte	Touch Controller	
0x4017_0000	64Kbyte	Crossbar Switch (XBAR)	12
0x4018_0000	64Kbyte	External Bus Interface (EBI)	15
0x4019_0000	64Kbyte	CACHE Module (CACHEM)	11
0x401A_0000	64Kbyte	Reserved	-
0x401B_0000	64Kbyte	Reserved	-
0x401C_0000	64Kbyte	CANBus Controller (CANBC)	20
0x401D_0000	64Kbyte	Reserved	-
0x401E_0000	64Kbyte	Serial Peripheral Interface Module - SPI0	23
0x401F_0000	64Kbyte	Serial Peripheral Interface Module - SPI1	23
0x6000_0000	64Kbyte	Synchronous Serial Interface 0 (SSI0) - QSPI0	22
0xE000_0000	4Kbyte	Embedded Interrupt Controller (EIC)	4
0xE000_1000	4Kbyte	Embedded Programmable Timer (EPT)	6

2. Signal Description

2.1. Pin Assignment

LT165 is available in the following packages:

- QFN-40 (5.0*5.0 mm^2) – LT165A
- TSSOP-30 (7.8*6.4 mm^2) – LT165B

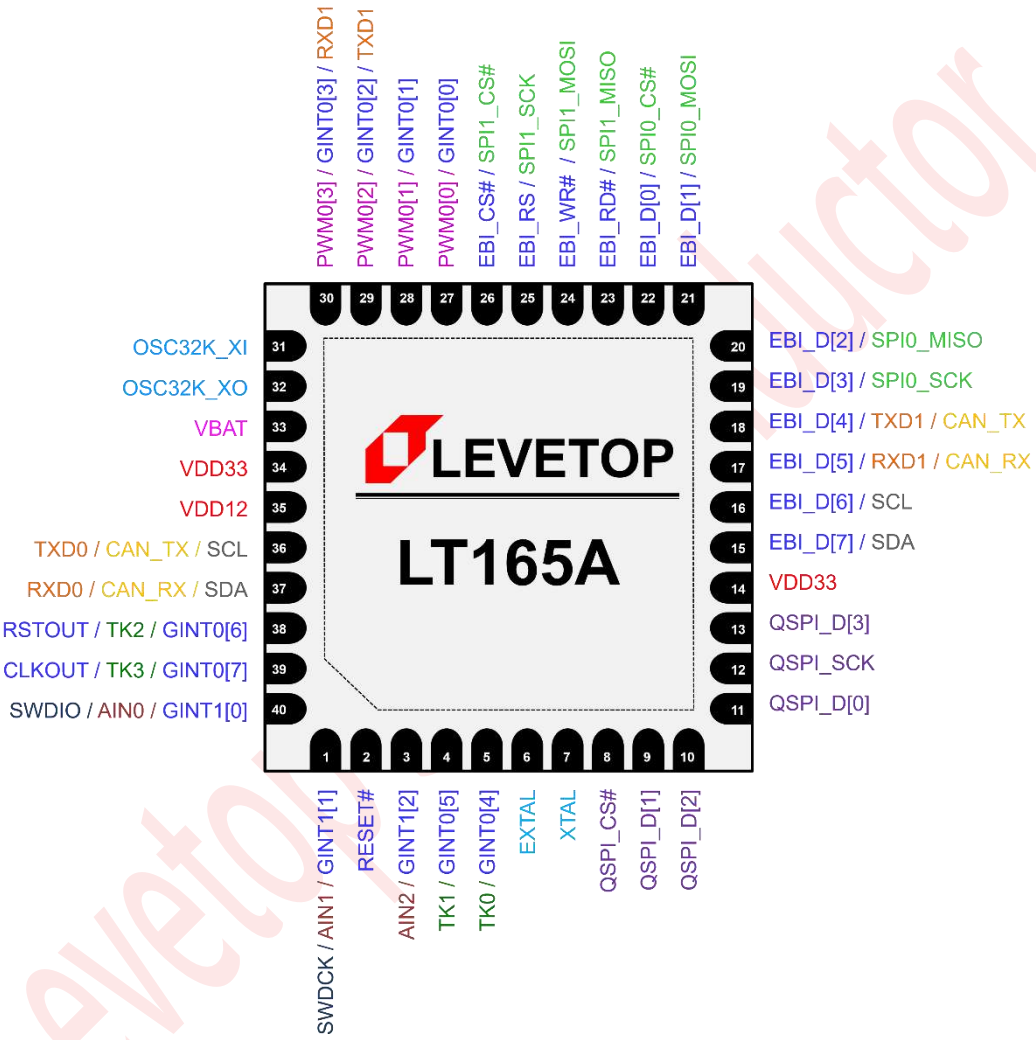


Figure 2-1: LT165A (QFN-40) Pin Assignment

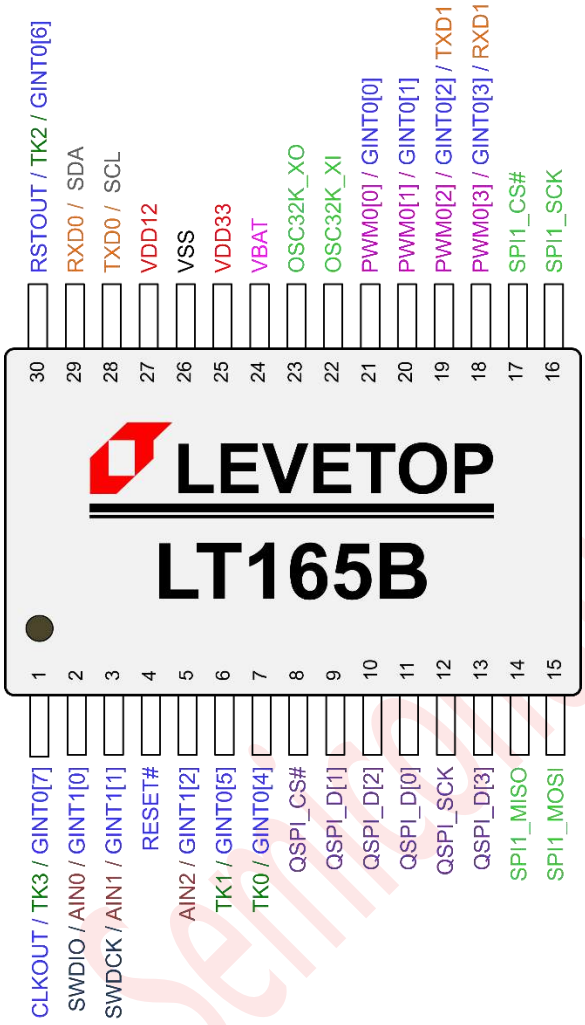


Figure 2-2: LT165B (TSSOP-30) Pin Assignment

2.2. Signal Properties Summary

Table 2-1: Signal Properties

Name	Alternate 0	Alternate 1	GPIO	Default State	Default Dir *1	Pullup *2	IO Type *3
SCI (2)							
TXD0	SCL	CAN_TX	GIOP29	HiZ	--	--	I/O
RXD0	SDA	CAN_RX	GPIO30	HiZ	--	--	I/O
QSPI (6)							
QSPI_CS#	--	--	GPIO19	HiZ	--	--	I/O
QSPI_D[0]	--	--	GPIO20	HiZ	--	--	I/O
QSPI_D[1]	--	--	GPIO21	HiZ	--	--	I/O
QSPI_D[2]	--	--	GPIO22	HiZ	--	--	I/O
QSPI_D[3]	--	--	GPIO23	HiZ	--	--	I/O
QSPI_SCK	--	--	GPIO24	HiZ	--	--	I/O
PWM0 (4)							
PWM0[0]	GINT0[0]	COMP0_OUT	GPIO0	HiZ	--	--	I/O
PWM0[1]	GINT0[1]	--	GPIO1	HiZ	--	--	I/O
PWM0[2]	GINT0[2]	TXD1	GPIO2	HiZ	--	--	I/O
PWM0[3]	GINT0[3]	RXD1	GPIO3	HiZ	--	--	I/O
ADC (1)							
AIN2	GINT1[2]	--	GPIO18	HiZ	--	--	I/O
Touch (2)							
TK0	GINT0[4]	--	GPIO4	HiZ	--	--	I/O
TK1	GINT0[5]	--	GPIO5	HiZ	--	--	I/O
Programming Port (2)							
SWDIO	GINT1[0]	AIN0	GPIO16	I	I	PullUp	I/O
SWDCK	GINT1[1]	AIN1	GPIO17	I	I	PullUp	I/O
CLOCK (5)							
EXTAL	--	--	--	HiZ/I	--/I	--	I
XTAL	--	--	--	HiZ/O	--/O	--	O
OSC32K_XI	--	--	--	I	I	-	I
OSC32K_XO	--	--	--	O	O	--	O
CLKOUT	GINT0[7]	TK3	GPIO7	O	O	--	I/O
RESET (2)							
RESET#	--	--	--	I	I	PullUp	I
RSTOUT	GINT0[6]	TK2	GPIO6	O	O	--	I/O
EBI (12)							

Name	Alternate 0	Alternate 1	GPIO	Default State	Default Dir *1	Pullup *2	IO Type *3
EBI_CS#	SPI1_CS#	--	GPIO25	HiZ	--	--	I/O
EBI_RS	SPI1_SCK	--	GPIO28	HiZ	--	--	I/O
EBI_RD#	SPI1_MISO	--	GPIO27	HiZ	--	--	I/O
EBI_WR#	SPI1_MOSI	--	GPIO26	HiZ	--	--	I/O
EBI_D[0]	SPI0_CS#	--	GPIO8	HiZ	--	--	I/O
EBI_D[1]	SPI0_MOSI	--	GPIO9	HiZ	--	--	I/O
EBI_D[2]	SPI0_MISO	--	GPIO10	HiZ	--	--	I/O
EBI_D[3]	SPI0_SCK	--	GPIO11	HiZ	--	--	I/O
EBI_D[4]	TXD1	CAN_TX	GPIO12	HiZ	--	--	I/O
EBI_D[5]	RXD1	CAN_RX	GPIO13	HiZ	--	--	I/O
EBI_D[6]	SCL	--	GPIO14	HiZ	--	--	I/O
EBI_D[7]	SDA	--	GPIO15	HiZ	--	--	I/O
Power Supply (5)							
VDD33	--	--	--	P	--	--	--
VDD12	--	--	--	P	--	--	--
AVDD	--	--	--	P	--	--	--
VBAT	--	--	--	P	--	--	--
VSS	--	--	--	G	--	--	--

NOTE:

1. 'Default Dir' refers to the direction after resetting. 'I' represents input, 'O' represents output, 'O (H)' represents output high, 'O (L)' represents output low, 'Hiz' represents both input and output disabled, and pull-up/pull-down is also disabled.

2. When the signal is set to output, all pull-up and pull-down are disconnected.

3. 'IO TYPE' refers to the pin design: 'I' represents a pin that only has input function; 'O' represents only the output function pin; 'I/O' represents a pin that has both input and output functions.

2.3. Signal Description

This chapter provides a brief explanation of pin signals. For more detailed information, please refer to the specific module section.

Tabel 2-2: Signal Description

Pin Name	Pin Number		Pin Description
	LT165A	LT165B	
Serial Communications Interface – 0, SCI0			
RXD0	37	29	**SCI (Uart) 0 Receive Data** This signal is used for SCI0 receiver data input. This signal can be configured as the SCI0 receiver data input, or be configured as SDA, CAN_RX or GPIO30. Please refer to Table 2-1 for information on share-used signals.
TXD0	36	28	**SCI (Uart) 0 Transmit Data** This signal is used for SCI0 transmitter data output. This signal can be configured as the SCI0 transmitter data output, or be configured as SCL, CAN_RX or GPIO29.
Serial Communications Interface – 1, SCI1			
RXD1	17 / 30	18	**SCI (Uart) 1 Receive Data** This signal is used for SCI1 receiver data input This signal has two multiplexing sources, one of which is share-used with EBI-D [5], CAN-RX, or GPO13 signals. The other is to share-used with PWM0[3], GINT0 [3], or GPIO3 signals.
TXD1	18 / 29	19	**SCI (Uart) 1 Transmit Data** This signal is used for SCI1 transmitter data output. This signal has two multiplexing sources, one of which is share-used with EBI-D [4], CAN_TX, or GPO12 signals. The other is share-used with PWM0[2], GINT0 [2], or GPIO2 signals.
CAN Bus			
CAN_RX	17 / 37	--	**Can Bus Receive Data** This signal comes from the receiving pin of the CANBus transceiver. The explicit state is represented by the logic level '0'. The implicit state is represented by the logic level "1". This signal has two multiplexing sources. When not configured for CANBUS operation, one is share-used with EBI-D [5], RXD1, or GPOI13 signals, and the other is share-used with RXD0, SDA, or GPIO30 signals.

Pin Name	Pin Number		Pin Description
	LT165A	LT165B	
CAN_TX	18 / 36	--	**Can Bus Transmit Data** This signal is the sending pin of the CANBus transceiver. The explicit state is represented by the logic level '0'. The implicit state is represented by the logic level "1". This signal has two multiplexing sources. When not configured for CANBUS operation, one is share-used with EBI-D [4], TXD1, or GPIO12 signals, and the other is share-used with TXD0, SCL, or GPIO29 signals.
External Bus Interface, EBI The external bus interface is responsible for controlling the information transmission between the internal bus and the external MCU type TFT LCD panel. Only LT165A supports 8bit EBI interface.			
EBI_CS#	26	--	**EBI Chip Selection (8080 I/F Panel Chip Selection)** LT165A provides an External Bus Interface (EBI) to drive the TFT LCD display panel of the 8bit 8080 interface, and this signal is the chip selection of the external bus interface. This signal is share-used with SPI1_CS# or GPIO25.
EBI_RS	25	--	**EBI Register Selection Signal** This signal is connected to the RS or A0 of the 8Bit LCD panel. This signal is share-used with SPI1_SCK or GPIO28.
EBI_RD#	23	--	**EBI Data Reading Control Signal** This signal is the control signal for LT165 to read data from the external 8Bit LCD panel. This signal is share-used with SPI1_MISO or GPIO27.
EBI_WR#	24	--	**EBI Data Writing Control Signal** This signal is the control signal for LT165 to write data to the external 8Bit LCD panel. This signal is share-used with SPI1_MOSI or GPIO26.
EBI_D[0]	22	--	**EBI Data Signal 0** This signal is the data Bit0 transmission signal from LT165 to the external 8Bit LCD panel. This signal is share-used with SPI0_CS# or GPIO8.
EBI_D[1]	21	--	**EBI Data Signal 1** This signal is the data Bit0 transmission signal from LT165 to the external 8Bit LCD panel. This signal is share-used with SPI0_MOSI or GPIO9.
EBI_D[2]	20	--	**EBI Data Signal 2** This signal is the data Bit0 transmission signal from LT165 to the external 8Bit LCD panel. This signal is share-used with SPI0_MISO or GPIO10 signal.
EBI_D[3]	19	--	**EBI Data Signal 3** This signal is the data Bit0 transmission signal from LT165 to the external 8Bit LCD panel. This signal is share-used with SPI0_SCK or GPIO11.

Pin Name	Pin Number		Pin Description
	LT165A	LT165B	
EBI_D[4]	18	--	**EBI Data Signal 4** This signal is the data Bit0 transmission signal from LT165 to the external 8Bit LCD panel. This signal is share-used with TXD1, CAN_TX or GPIO12.
EBI_D[5]	17	--	**EBI Data Signal 5** This signal is the data Bit0 transmission signal from LT165 to the external 8Bit LCD panel. This signal is share-used with RXD1, CAN_RX or GPIO13.
EBI_D[6]	16	--	**EBI Data Signal 6** This signal is the data Bit0 transmission signal from LT165 to the external 8Bit LCD panel. This signal is share-used with SCL or GPIO14.
EBI_D[7]	15	--	**EBI Data Signal 7** This signal is the data Bit0 transmission signal from LT165 to the external 8Bit LCD panel. This signal is share-used with SDA or GPIO15.
I2C			
SCL	16 / 36	28	**I2C Clock Signal** This signal is used for I2C clock line signal. This signal has two multiplexing sources. When not configured for I2C operation, one is share-used with EBI-D [6] or GPIO14 signals, and the other is share-used with TXD0, CAN_TX, or GPIO29.
SDA	15 / 37	29	**I2C Data** This signal is used for I2C data line signals. This signal has two multiplexing sources. When not configured for I2C operation, one is share-used with EBI-D [7] or GPIO15 signals, and the other is share-used with RXD0, CAN-RX, or GPIO30 signals.
QSPI			
QSPI_D[3:0]	13, 10, 9 11	13, 10, 9, 11	**QSPI Data Input/Output** These signals are the data outputs or inputs of QSPI in master mode.
QSPI_CS#	8	8	**QSPI Chip Select Signal** This signal is the chip select output of QSPI in master mode, and active low.
QSPI_SCK	12	12	**QSPI Clock Signal** This signal is the clock output of QSPI in master mode.
SPI			
SPI0_MOSI	21	--	**SPI #0 Data Output** This signal is the data output of the first group SPI. This signal is share-used with EBI_D[1] or GPIO9.

Pin Name	Pin Number		Pin Description
	LT165A	LT165B	
SPI0_MISO	20	--	**SPI #0 Data Input** This signal is the data input of the first group SPI. This signal is share-used with EBI_D[2] or GPIO10.
SPI0_CS#	22	--	This signal is the chip select output of the first group SPI. This signal is share-used with EBI_D[0] or GPIO8.
SPI0_SCK	19	--	**SPI #0 Serial Clock Signal** This signal is the clock output of the first group SPI. This signal is share-used with EBI_D[3] or GPIO11.
SPI1_MOSI	24	15	**SPI #1 Data Output** This signal is the data output of the second group SPI. This signal is share-used with EBI_WR# or GPIO26.
SPI1_MISO	23	14	**SPI #1 Data Input** This signal is the data input of the second group SPI. This signal is share-used with EBI_RD# or GPIO27.
SPI1_CS#	26	17	**SPI #1 Chip Select Signal** This signal is the chip select output of the second group SPI. This signal is share-used with EBI_CS# or GPIO25.
SPI1_SCK	25	16	**SPI #1 Serial Clock Signal** This signal is the clock output of the second group SPI. This signal is share-used with EBI_RS or GPIO28.
Touch Key			
TK0	5	7	**Touch Key 0** Input This signal is share-used with GINT0[4] or GPIO4.
TK1	4	6	**Touch Key 1** Input This signal is share-used with GINT0[5] or GPIO5.
TK2	38	30	**Touch Key 2** Input This signal is share-used with RSTOUT, GINT0[6] or GPIO6.
TK3	39	1	**Touch Key 2** Input This signal is share-used with CLKOUT, GINT0[7] or GPIO7. ◦
EPORT 0			
GINT0[0]	27	21	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with PWM0[0], COMP0_OUT or GPIO0.

Pin Name	Pin Number		Pin Description
	LT165A	LT165B	
GINT0[1]	28	20	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with PWM0[1] or GPIO1.
GINT0[2]	29	19	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with PWM0[2], TXD1 or GPIO2.
GINT0[3]	30	18	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with PWM0[3], RXD1 or GPIO3.
GINT0[4]	5	7	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with TK0 or GPIO4.
GINT0[5]	4	6	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with TK1 or GPIO5.
GINT0[6]	38	30	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with RSTOUT, TK2 or GPIO6.
GINT0[7]	39	1	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with CLKOUT, TK3 or GPIO7.
EPORT 1			
GINT1[0]	40	2	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with SWDIO, AIN0 or GPIO16.
GINT1[1]	1	3	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with SWDCK, AIN1 or GPIO17.
GINT1[2]	3	5	**Interrupt Input / GPIO** These bidirectional signals can be used as external interrupt sources or GPIO. This signal is share-used with AIN2 or GPIO18.
PWM 0			

Pin Name	Pin Number		Pin Description
	LT165A	LT165B	
PWM0[3:0]	30, 29, 28, 27	18, 19, 20, 21	**PWM Output** These output signals can be used as PWM0 outputs, GINT0[3:0] or GPIO[3:0].
ADC			
AIN2	3	5	**Analog Input Signal** This analog signal is used as an ADC analog input channel. When not configured as an analog input, this signal can also be used for GINT1[2] or GPIO18.
AIN1	1	3	**Analog Input Signal** This analog signal is used as an ADC analog input channel. When not configured as an analog input, this signal can also be used for SWDCK, GINT1[1] or GPIO17.
AIN0	40	2	**Analog Input Signal** This analog signal is used as an ADC analog input channel. When not configured as an analog input, this signal can also be used for SWDIO, GINT1[0] or GPIO16.
Programming Signals			
SWDCK	1	3	**MCU Code Programming Clock** This input signal is the clock signal used for programming internal flash memory. When not configured as an analog input, this signal can also be used for GINT1[1], AIN1 or GPIO17.
SWDIO	40	2	**MCU Code Programming Data** This signal is used as a data signal for programming internal flash memory. When not configured as SWDIO, this signal can also be used for GINT1[0], AIN0 or GPIO16.
Clock			
EXTAL	6	--	**System Oscillator Input** The signal is the input of 24MHz Oscillator Pad.
XTAL	7	--	**System Oscillator Output** The signal is the output 24MHz Oscillator pad.
OSC32K_XI	31	22	**32.768KHz Oscillator Input** The signal is the input of 32.768KHz Oscillator pad.
OSC32K_XO	32	23	**32.768KHz Oscillator Output** The signal is the output 32.768KHz Oscillator pad.

Pin Name	Pin Number		Pin Description
	LT165A	LT165B	
CLKOUT	39	1	**Clock Out** This output signal reflects the internal system clock. When not configured as Clock output, this signal can also be configured as GINT0[7], TK3 or GPIO7.
Reset			
RESET#	2	4	**Reset Input Signal** When RESET#=0, a reset action will be performed on the internal MCU. Except for a few registers that can only be reset by POR, most registers controlled by MCU will return to their default values.
RSTOUT	38	30	**Reset Output Signal** This output signal indicates that the internal reset controller is resetting the chip. 0=The chip is in a reset state 1=chip not reset state When not configured as a reset output, this signal can also be share-used for GINT0 [6], TK2, or GPIO6.
Power and Grounding Signals These signals provide system power and grounding for the chip. It is necessary to ensure that the chip can obtain sufficient current capability. All power signals must have sufficient bypass capacitance to suppress high-frequency noise.			
VDD33	14, 34	25	**3.3V Power Input** This signal supplies 3.3V positive power to the I/O pads and LDO.
VDD12	35	27	**1.2V LDO Output** This 1.2V LDO output signal is used to supply the power of the core logic. One 1uF and one 0.1uF ceramic bypass capacitors are required to externally connect between the pad and VSS.
AVDD	--	--	**3.3V Analog Power Input** This signal supplies 3.3V positive power to ADC module.
VBAT	33	24	**RTC Power** This signal supplies battery power to RTC module.
VSS	41(*)	26	This signal supplies 3.3V negative supply (ground) to the I/O pads and LDO.

NOTE: *This is a thermal pad zone that must be connected to VSS or GND. When making PCB layout, special attention should be paid to the solder surface design of the pads. Please refer to Section 30.3 for details.*

2.4. LT165A vs. LT165B

Table 2-3: LT165x Comparison

Functions		LT165A	LT165B
Items	**Description**		
TFT LCD Panel	8bit 8080 I/F	V	V
	SPI I/F	V	V
MCU	MCU Core	32bits RISC	32bits RISC
	MCU Clock	150MHz	150MHz
	SRAM	32KB	32KB
Interface	SCI (Uart)	V (x2)	V (x2)
	SPI	V (x2)	V (x1)
	QSPI	V (x1)	V (x1)
	PWM O/P	V (x4)	V (x3)
	Can Bus	V (x1)	V (x1)
	ADC I/P	V (x3)	V (x3)
	Touch Key	V (x4)	V (x4)
	CTP I2C I/F	V	V
	GPIO Port	V (x11)	V (x9)
	RTC	V	V
Application	UI_Editor-II	V	V
	UI_Emulator-II	V	V
	Uart Port Upgrade	V	V
	Support 2nd Develop	V	V
Power & Package	Power	3.3V	3.3V
	Package	QFN-40	TSSOP-30

3. Hardware Interface

3.1. Host Communication Interface

The communication between LT165 and the Host MCU is through SCI (UART) interface. The UART interfaces TX and RX on both sides must be cross docked, as shown in the following figure. If the connection distance is long, an RS232 driver chip needs to be added to avoid signal attenuation affecting communication. The software settings and communication protocol for serial Uart can refer to the application note (UI-Editor-II_CH_Vxx.pdf) of Levetop Semiconductor.

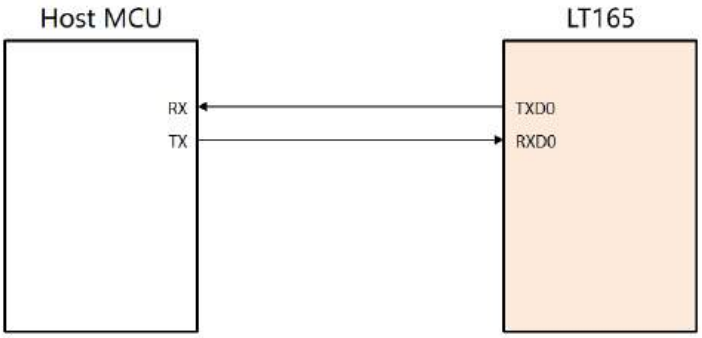


Figure 3-1: UART Connection Between LT165 and Host MCU

3.2. TFT LCD Panel Interface

LT165A provides an External Bus Interface (EBI) for driving a TFT LCD panel with a parallel 8bit 8080 interface. The schematic diagram is shown in Figure 3-2 below:

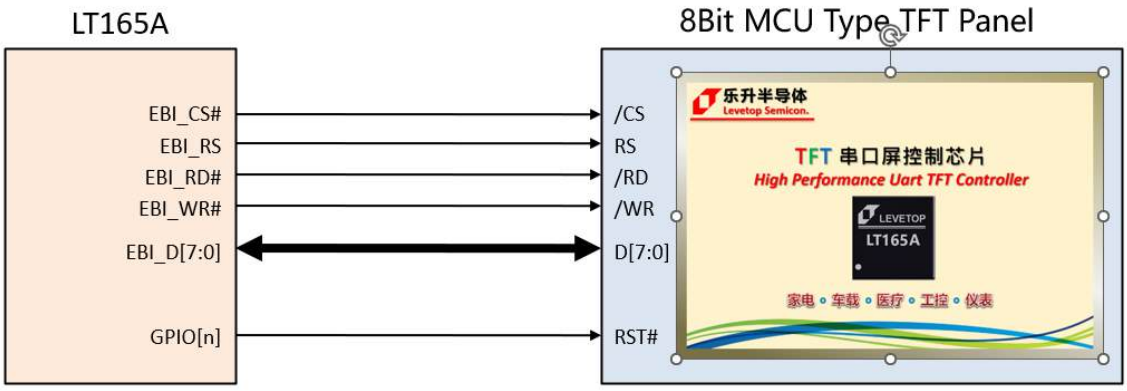


Figure 3-2: LT165A Connect to 8bits 8080 I/F of TFT LCD Panel

NOTE: *LT165B does not support the EBI module, so it cannot connect to 8bits 8080 I/F of TFT LCD Panel. LT165A or LT165B can also connect to SPI I/F of TFT LCD Panel. The schematic diagram is shown in Figure 3-3 below:*

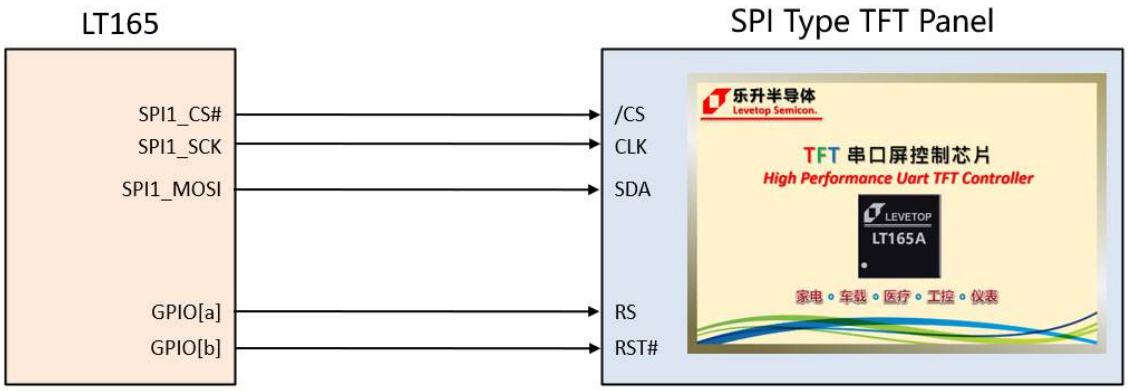


Figure 3-3: LT165 Connect to SPI I/F of TFT LCD Panel

3.3. QSPI Interface

LT165 has a set of QSPI interfaces for connecting to external QSPI Flash. This external Flash is used to store program code, display images, animations, text, and other information. When LT165 receives a serial command sent by the main control through the Uart interface, it will extract images or other display related information from QSPI Flash according to the command and transmit it to the LCD panel. The reference schematic is shown in the following figure.

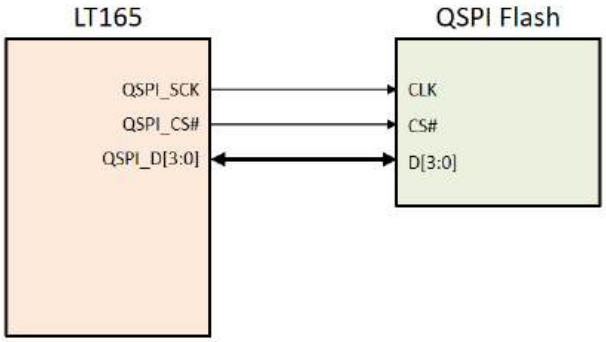


Figure 3-4: LT165 Connect to QSPI Flash

3.4. LCD Touch Panel Interface

LT165 has an ADC module and I2C interface that can be used to interface directly to resistive or capacitive touch panel. Upon receiving the touch information, LT165 will process it and transmit it to the Host MCU. The reference schematic is as followings:

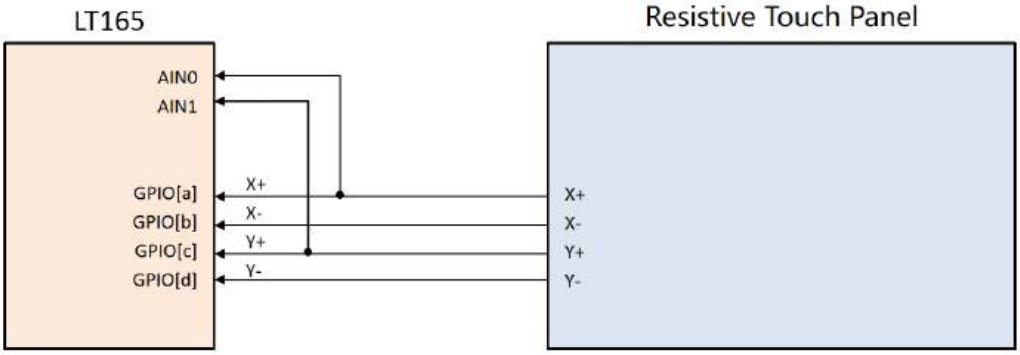


Figure 3-5: LT165 Connect to Resistive Touch Panel

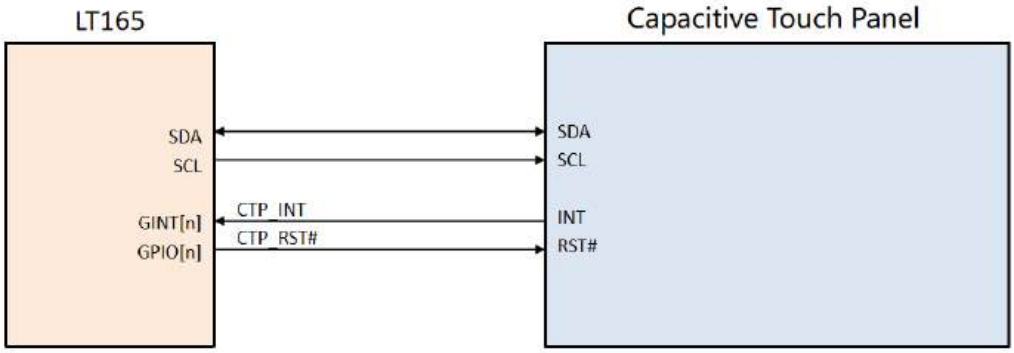


Figure 3-6: LT165 Connect to Capacitive Touch Panel

3.5. Clock Interface

LT165A requires an external 12MHz crystal oscillator as the internal system clock source. The reference schematic is as follows:

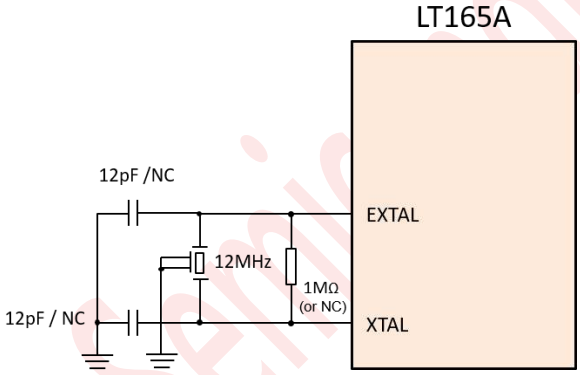


Figure 3-7: External 12MHz Clock Circuit

3.6. Can Bus Interface

LT165 supports the Can Bus protocol and provides a set of Can Bus interfaces When in use, an additional Can Bus driver chip is required to communicate with the external environment. Please refer to the schematic diagram below:

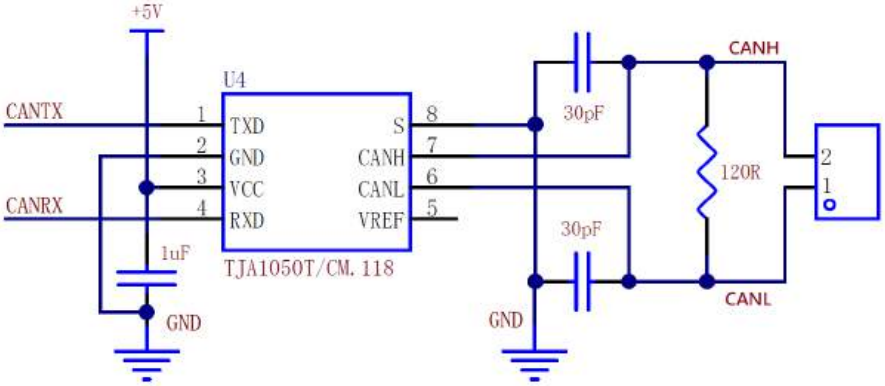


Figure 3-8: Canbus Circuit Example

3.7. LCD Backlight Control Circuit

LT165 uses PWM1[3] to provide a backlight control signal - "BL_PWM" that can be used to control TFT LCD panel backlight. The reference schematics are as followings:

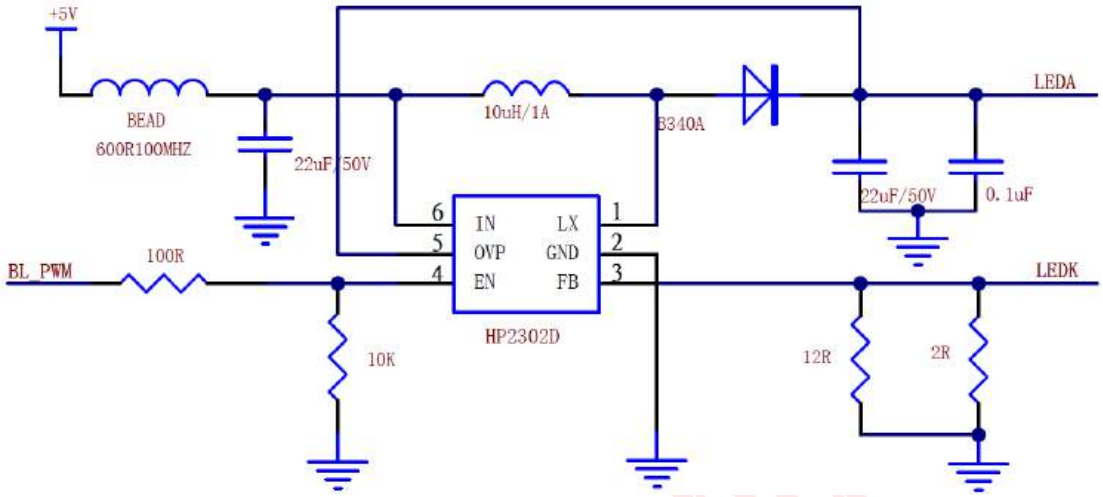


Figure 3-9: TFT LCD Backlight Circuit Example 1

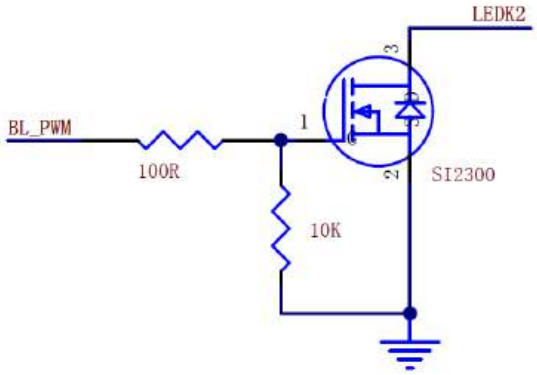


Figure 3-10: TFT LCD Backlight Circuit Example 2

3.8. Real Time Clock (RTC)

The LT165 has an RTC (Real Time Clock) clock module inside If using this RTC clock, a 32.768KHz crystal oscillator circuit is required RTC is independently powered, and if RTC continues to operate even when the external power is turned off, an external battery power can be added The reference schematic diagram is as follows:

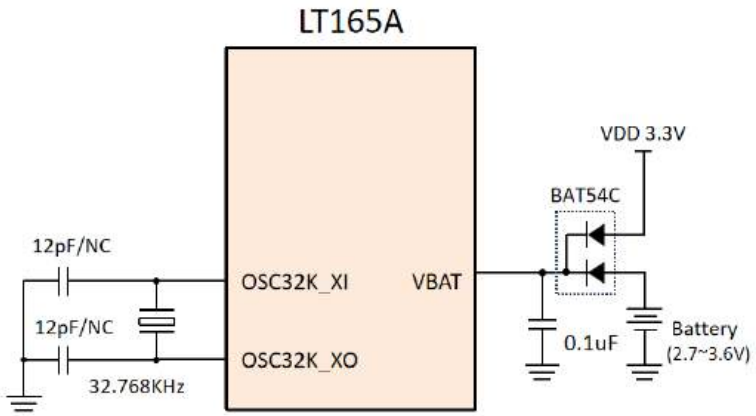


Figure 3-11: RTC Application Circuit

3.9. Reset Interface

There are two hardware reset sources for LT165, both of which are synchronized by the internal clock:

– Power on Reset

– External Reset Pin (RESET#)

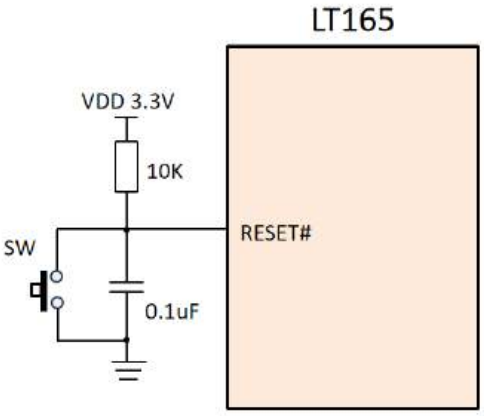


Figure 3-12: External Reset Circuit

4. Embedded Interrupt Controller (EIC)

This section describes the Embedded Interrupt Controller of MCU.

4.1. Introduction

The interrupt controller collects requests from multiple interrupts sources and provides an interface to the CPU interrupt logic.

4.2. Features

Features of the interrupt controller module include:

- Interrupt sources configurable, up to 32
- 32 unique programmable priority levels for each interrupt source
- Independent enable/disable of pending interrupts based on priority level
- A fixed vector number for each interrupt source
- Support both level-sensitive and pulse interrupts
- Support PendTrap function
- Support Software reset

4.3. Memory Map and Registers

This subsection describes the memory map (see Table 4-1) and registers.

4.3.1. Memory Map

EIC module base address(EIC_BASEADDR) is defined in RISC Core internal parameter. The default value is 0xE000_0000. The EIC registers actual address is EIC_BASEADDR plus the offset address of each EIC registers. The core internal modules occupies 64K address area. The system should avoid mapping the other registers to the area from EIC_BASEADDR to EIC_BASEADDR+0x0000_FFFF.

Table 4-1: Interrupt Controller Module Memory Map

Offset Address	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0	Access(1)
0x0000_0000	Interrupt control status register (ICSR)				S/U
0x0000_0004	Reserved				S/U
0x0000_0008	Reserved				S/U
0x0000_000C	Reserved				S/U
0x0000_0010	Interrupt Enable Register (IER)				S/U
0x0000_0014	Reserved				S/U
0x0000_0018	Interrupt Pending Set Register (IPSR)				S/U
0x0000_001C	Interrupt Pending Clear Register (IPCR)				S/U
0x0000_0020 through 0x0000_003C	Unimplemented(2)				—
Priority level select registers (PLSR0-PLSR31)					
0x0000_0040	PLSR3	PLSR2	PLSR1	PLSR0	S/U
0x0000_0044	PLSR7	PLSR6	PLSR5	PLSR4	S/U
0x0000_0048	PLSR11	PLSR10	PLSR9	PLSR8	S/U
0x0000_004C	PLSR15	PLSR14	PLSR13	PLSR12	S/U
0x0000_0050	PLSR19	PLSR18	PLSR17	PLSR16	S/U
0x0000_0054	PLSR23	PLSR22	PLSR21	PLSR20	S/U
0x0000_0058	PLSR27	PLSR26	PLSR25	PLSR24	S/U
0x0000_005C	PLSR31	PLSR30	PLSR29	PLSR28	S/U
0x0000_0060	System Priority level select register(SYSPLSR)				S/U
0x0000_0064 through 0x0000_007C	Unimplemented(2)				—

NOTE:

1. In RISC Core, there register can be accessed in any case.

2. Accesses to unimplemented address locations have no effect and result in a cycle termination transfer error.

4.3.2. Registers

This subsection contains a description of the interrupt controller module registers.

4.3.2.1. Interrupt Control Status Register

The 32bit interrupt control register (ICSR) reflects the state of the interrupt controller outputs to the CPU.

Offset Address: 0x0000

	31	30	29	28	27	26	25	24
R	SRST	0	0	SetPTrap	ClrPTrap	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	VEC6	VEC5	VEC4	VEC3	VEC2	VEC1	VEC0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 4-1: Interrupt Control Status Register (ICSR)

SRST — Software Reset Bit

The write-only bit is used to create a software reset request. Setting this bit will generate a pulse on SYSRESETREQ signal. Reads always return 0.

SetPTrap — Set PendTrap Bit

The read/write bit is used to create a pending software interrupt. The action is similar to execute “trap” instruction. However, the pending software interrupt will not be entered until all the higher priority exceptions/interrupts exit. When the software interrupt entered, the bit will be cleared automatically. Reset also clears this bit.

On reads :

1 = the software interrupt is pending
0 = the software interrupt is not pending

On writes :

1 = set software interrupt to pending
0 = no effect

ClrPTrap — Clear PendTrap Bit

The read/write ClrDSI bit is used to cancel the pending software interrupt(PendTrap). Reset clears this bit.

On reads :

1 = the software interrupt is pending

0 = the software interrupt is not pending

On writes :

1 = cancel the pending software interrupt

0 = no effect

VEC[6:0] — Interrupt Vector Number Field

The read-only VEC[6:0] field contains the 7bit interrupt vector number. Reset clears VEC[6:0].

4.3.2.2. Interrupt Enable Register

The read/write, 32bit Interrupt Enable Register (IER) individually enables any current pending interrupts which are assigned to each priority level as a normal interrupt source. Enabling an interrupt source which has an asserted request causes that request to become pending, and a request to the CPU is asserted if not already outstanding.

Offset Address: 0x0010

	31	30	29	28	27	26	25	24
R W	IE31	IE30	IE29	IE28	IE27	IE26	IE25	IE24
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	IE23	IE22	IE21	IE20	IE19	IE18	IE17	IE16
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 4-2: Interrupt Enable Register (IER)

IE[31:0] — Interrupt Enable Field

The read/write IE[31:0] field enables interrupt requests from sources at the corresponding priority level as interrupt requests. Reset clears IE[31:0].

1 = interrupt request enabled

0 = interrupt request disabled

4.3.2.3. Interrupt Pend Set Register

Offset Address: 0x0018

	31	30	29	28	27	26	25	24
R W	SetPend31	SetPend30	SetPend29	SetPend28	SetPend27	SetPend26	SetPend25	SetPend24
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R W	SetPend23	SetPend22	SetPend21	SetPend20	SetPend19	SetPend18	SetPend17	SetPend16
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R W	SetPend15	SetPend14	SetPend13	SetPend12	SetPend11	SetPend10	SetPend9	SetPend8
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	SetPend7	SetPend6	SetPend5	SetPend4	SetPend3	SetPend2	SetPend1	SetPend0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 4-3: Interrupt Pend Set Register (IPSR)

SetPend[31:0] — Interrupt Pend Set Field

The read/write SetPend[31:0] field set pend to associated interrupt and indicate whether the associated interrupt is pending. Reset clears SetPend[31:0].

On reads :

1 = the associated interrupt is pending
0 = the associated interrupt is not pending

On writes :

1 = change the state of associated interrupt to pending
0 = no effect

4.3.2.4. Interrupt Pend Clear Register

Offset Address: 0x001C

	31	30	29	28	27	26	25	24
R W	ClrPend31	ClrPend30	ClrPend29	ClrPend28	ClrPend27	ClrPend26	ClrPend25	ClrPend24
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R W	ClrPend23	ClrPend22	ClrPend21	ClrPend20	ClrPend19	ClrPend18	ClrPend17	ClrPend16
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R W	ClrPend15	ClrPend14	ClrPend13	ClrPend12	ClrPend11	ClrPend10	ClrPend9	ClrPend8
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	ClrPend7	ClrPend6	ClrPend5	ClrPend4	ClrPend3	ClrPend2	ClrPend1	ClrPend0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 4-4: Interrupt Pend Clear Register (IPCR)

ClrPend[31:0] — Interrupt Pend Clr Field

The read/write ClrPend[31:0] field clear pend to associated interrupt and indicate whether the associated interrupt is pending. Reset clears ClrPend[31:0].

On reads :

1 = the associated interrupt is pending
0 = the associated interrupt is not pending

On writes :

1 = change the state of associated interrupt to not pending
0 = no effect

4.3.2.5. Priority Level Select Registers

The read/write 8bit Priority Level Select Registers (PLSRx) are 32 read/write, 8bit priority level select registers PLSR0 ~ PLSR31, one for each of the interrupt source. The PLSRx register assigns a priority level to interrupt source x.

Offset Address: 0x0040 ~ 0x005C

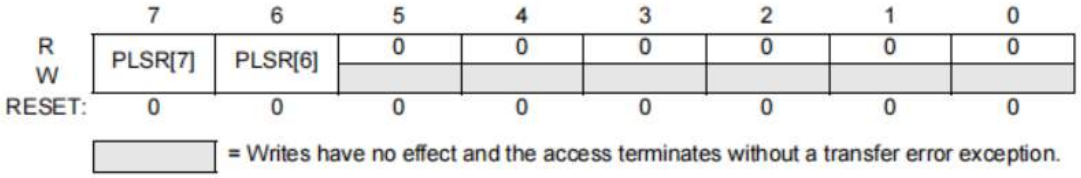


Figure 4-5: Priority Level Select Registers (PLSR0 ~ PLSR31)

PLSRx[7:6] — Priority Level Select Field

IRQ0~31 has a default priority value 0~31. The lower the value, the higher the priority. That means IRQ0 priority > IRQ1 > ... > IRQ31 as default. However, user can set PLSRx[7:6] to adjust the interrupt priority. The actual value of priority level is the default value plus PLSRx[7:6] *64. For instance, if PLSR1[7:6] = 2, IRQ1's priority value is 1+2*64 = 129, then IRQ1's priority is lower than any IRQ with lower priority value.

Table 4-2: Priority Level Value Adjustment

PLSRx[7:6]	Priority Level Value
00	0
01	64
10	128
11	192

4.3.2.6. System Priority Level Select Registers

Offset Address: 0x0060

	31	30	29	28	27	26	25	24
R	EPTPRI[7]	EPTPRI[6]	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	SIPRI[7]	SIPRI[6]	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	0	0						
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 4-6: System Priority Level Select Registers (SYSPLSR)

EPTPRI[7:6] — EPT Priority Level Select Field

EPT interrupt's default priority is -2. That means EPT interrupt priority is higher than the other normal IRQs and system software interrupt(PendTrap) as default. The lower the value, the higher the priority. However, user can set PRI[7:6] to adjust the EPT interrupt priority. The actual value of priority level is the default value plus PRI[7:6] *64. For instance, if PRI[7:6] = 2, EPT interrupt priority value is -2+2*64 = 126.

Table 4-3: EPT Priority Value Adjustment

EPTPRI[7:6]	EPT Priority Value
00	0
01	64
10	128
11	192

SIPRI[7:6] — Software Interrupt Priority Level Select Field

Software interrupt(PendTrap) default priority is -1. That means EPT interrupt priority is higher than the other normal IRQs as default. The lower the value, the higher the priority. However, user can set SIPRI[7:6] to adjust the software interrupt priority. The actual value of priority level is the default value plus SIPRI[7:6] *64. For instance, if SIPRI[7:6] = 2, software interrupt priority value is -1+2*64 = 127.

Table 4-4: Software Interrupt Priority Value Adjustment

SIPRI[7:6]	Software Interrupt Priority Value
00	0
01	64
10	128
11	192

4.4. Function Description

EIC supports both level-sensitive and pulse interrupts. Interrupt source number is from 1 to 32.
The interrupt becomes pending because one of the following reasons:

- the EIC detects that the interrupt signal is active and the corresponding interrupt is not active
- the EIC detects a rising edge on the interrupt signal

The pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt. This changes the state of the interrupt from pending to active. Then:

  – For a level-sensitive interrupt, when the processor returns from the ISR, the EIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.

  – For a pulse interrupt, the EIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.

- Software writes to the corresponding interrupt Pend Clear Register bit.

4.4.1. Interrupt Handling Without Confliction

If an interrupt is pulsed, the state of the interrupt changes to pending. Without confliction, the interrupt causes the processor to immediately enter the ISR. When the processor returns from the ISR, the state of the interrupt changes to inactive.

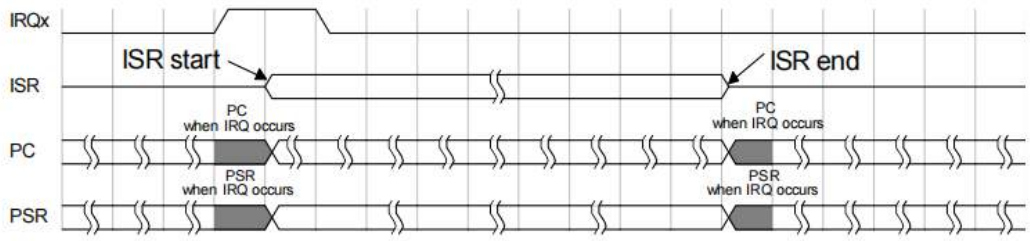


Figure 4-7: One Pulse Interrupt Without Confliction

For a level-sensitive interrupt, the state of the interrupt changes to pending if the signal is asserted. Without confliction, the interrupt causes the processor to immediately enter the ISR. When the processor returns from the ISR, EIC continues to samples the interrupt signal. If the signal is not cleared, the processor will re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.

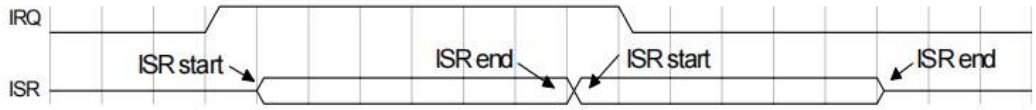


Figure 4-8: Level-Sensitive Interrupt Without Confliction

4.4.2. Interrupt With Confliction

When two interrupt signals are asserted at the same time, the Interrupt Arbiter will judge which one has the greater priority. For instance, if the priority of IRQx is greater than IRQy, the processor will enter ISRx and IRQy becomes pending. After the processor returns from ISRx, the processor will enter ISRy immediately.

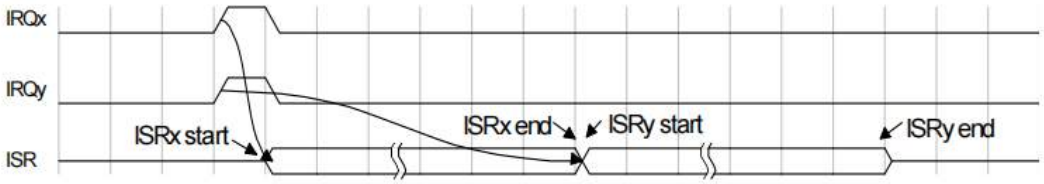


Figure 4-9: Two Interrupts Occur at The Same Time

If an interrupt signal is asserted during another interrupt handling, there are two cases:

1. The asserted interrupt priority is lower than the handling interrupt priority. In this case, the asserted interrupt state is pending until the handling interrupt ends.
2. The asserted interrupt priority is higher than the handling interrupt priority. In this case, the higher priority interrupt handling will be nested in the lower priority interrupt routine.

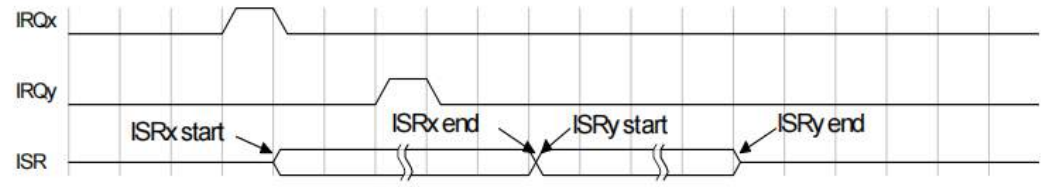


Figure 4-10: A Lower Priority Interrupt Asserted With Confliction

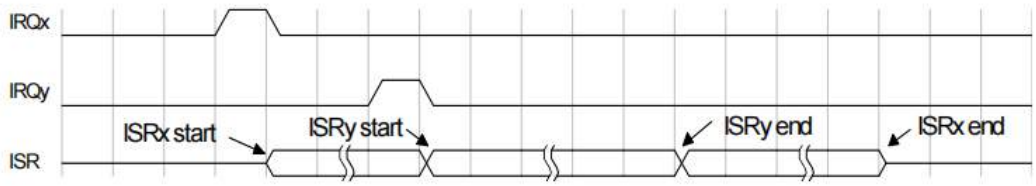


Figure 4-11: A higher Priority Interrupt Asserted With Confliction

4.4.3. Pend Trap Function

SetPTrap / ClrPTrap bits in ICSR are used to create/cancel a “pending” software interrupt request while a higher priority interrupt is handling. As soon as the processor returns from the higher priority interrupt, the “pending” software interrupt will be accepted by the processor.

Usually, Pend Trap function is used for OS task passive switch.

4.5. Interrupts

The interrupt controller assigns a number to each interrupt source, as 4.5 shows.

Table 4-5: Interrupt Source Assignment

Source	Module	Flag	Source Description	Flag Clearing Mechanism
0	ADC			
1	QSPI0	XRXOIS	XIP Receive FIFO Overflow Interrupt	
		RXFIS	Receive FIFO Full Interrupt	
		RXOIS	Receive FIFO Overflow Interrupt	
		RXUIS	Receive FIFO Underflow Interrupt	
		TXOIS	Transmit FIFO Overflow Interrupt	
		TXEIS	Transmit FIFO Empty Interrupt	
2	SCI0	TDRE	Transmit Data Register Empty Flag	
		TC	Transmission Complete	
		TXOF	Transmitter Buffer Overflow Flag	
		LBKDIF	LIN Break Detect Interrupt Flag	
		IDLE	Idle Line Flag	
		RXEDGIF	RXD0 Pin Active Edge Interrupt Flag	
		RDRF	Receive Data Register Full Flag	
		MA1F	Match 1 Flag	
		MA2F	Match 2 Flag	
		OR	Receiver Overrun Flag	
		NF	Noise Flag	
		FE	Framing Error Flag	
		PF	Parity Error Flag	
		RXUF	Receiver Buffer Underflow Flag	
3	COMP0	CPRIF		
		CPFIF		
4	Reserved			
5	DMAC	DONE[0]		Write DONE[0]=1

Source	Module	Flag	Source Description	Flag Clearing Mechanism
		DONE[1]		Write DONE[1]=1
		...		...
		DONE[14]		Write DONE[14]=1
		DONE[15]		Write DONE[15]=1
		DMA_ESR[CPE]	Group Priority Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[G PE]	Channel Priority Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[S AE]	Source Address Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[S OE]	Source Offset Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[DAE]	Destination Address Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[DOE]	Destination Offset Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[NCE]	Nbytes/Citer Configuration Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[S GE]	Scatter/Gather Configuration Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[S BE]	Source Bus Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[DBE]	Destination Bus Error	Write channel number to CERR[6:0] to clear error status
6	WDT0	IF		
7	PWM0	PIFR[0]		
		PIFR[1]		
		PIFR[2]		
		PIFR[3]		
8	Reserved			
9	PIT0	PIF	PIT Flag	Writing a 1 to it or writing to PMR
10	PIT1	PIF	PIT Flag	Writing a 1 to it or writing to PMR
11	Reserved			
12	Reserved			
13	RTC	Day_intf	Day pulse flag	
		Hou_intf	Hour pulse flag	
		Min_intf	Minute pulse flag	
		Sec_intf	Seconds pulse flag	
		Ala_intf	Alarm flag	
		1KHz_intf	1KHz pulse flag	
		32KHz_intf	32KHz pulse flag	
14	TOUCH			
15	I2C	I2C Flag	I2C Flag	
16	Reserved			
17	PVD	PVDO	PVD Flag	
18	CANBUS	CAN_IFRH[BUF15:BUF0]	CAN buffers 15 ~ 0 interrupts	This bit is cleared by writing it to '1'

Source	Module	Flag	Source Description	Flag Clearing Mechanism
19	CANBUS	CAN_ESR[BOFF_INT]	CAN bus off interrupt	This bit is cleared by writing it to '1'
20	CANBUS	CAN_ESR[ERR_INT]	CAN error interrupt	This bit is cleared by writing it to '1'
21	CANBUS	CAN_ESR[TWRN_INT]	CAN transmit warning interrupt	This bit is cleared by writing it to '1'
22	CANBUS	CAN_ESR[RWRN_INT]	CAN receive warning interrupt	This bit is cleared by writing it to '1'
23	CANBUS	CAN_ESR[WKUP_INT]	Wake Up Interrupt	This bit is cleared by writing it to '1'
24	Reserved			
25	Reserved			
26	SPI0	MODF	Mode fault	
		EOTF	Transmission complete	Write-One Clear
		TXFTO	TXFIFO timeout	Write-One Clear
		TXFOVF	TXFIFO overflow	Write-One Clear
		TXFUDF	TXFIFO underflow	Write-One Clear
		TXFSER	TXFIFO service	Write-One Clear
		RXFTO	RXFIFO timeout	Write-One Clear
		RXFOVF	RXFIFO overflow	Write-One Clear
		RXFUDF	RXFIFO underflow	Write-One Clear
		RXFSER	RXFIFO service	Write-One Clear
27	SPI1	MODF	Mode fault	
		EOTF	Transmission complete	Write-One Clear
		TXFTO	TXFIFO timeout	Write-One Clear
		TXFOVF	TXFIFO overflow	Write-One Clear
		TXFUDF	TXFIFO underflow	Write-One Clear
		TXFSER	TXFIFO service	Write-One Clear
		RXFTO	RXFIFO timeout	Write-One Clear
		RXFOVF	RXFIFO overflow	Write-One Clear
		RXFUDF	RXFIFO underflow	Write-One Clear
		RXFSER	RXFIFO service	Write-One Clear
28	SCI1	TDRE	Transmit Data Register Empty Flag	
		TC	Transmission Complete	
		TXOF	Transmitter Buffer Overflow Flag	
		LBKDIF	LIN Break Detect Interrupt Flag	
		IDLE	Idle Line Flag	
		RXEDGIF	RXD1 Pin Active Edge Interrupt Flag	
		RDRF	Receive Data Register Full Flag	
		MA1F	Match 1 Flag	
		MA2F	Match 2 Flag	

Source	Module	Flag	Source Description	Flag Clearing Mechanism
		OR	Receiver Overrun Flag	
		NF	Noise Flag	
		FE	Framing Error Flag	
		PF	Parity Error Flag	
		RXUF	Receiver Buffer Underflow Flag	
29	Reserved			
30	EPORT0	EPF0	Edge port 0 flag 0	Write EPF0 = 1
		EPF1	Edge port 0 flag 1	Write EPF1 = 1
		EPF2	Edge port 0 flag 2	Write EPF2 = 1
		EPF3	Edge port 0 flag 3	Write EPF3 = 1
		EPF4	Edge port 0 flag 4	Write EPF4 = 1
		EPF5	Edge port 0 flag 5	Write EPF5 = 1
		EPF6	Edge port 0 flag 6	Write EPF6 = 1
		EPF7	Edge port 0 flag 7	Write EPF7 = 1
31	EPORT1	EPF0	Edge port 1 flag 0	Write EPF0 = 1
		EPF1	Edge port 1 flag 1	Write EPF1 = 1
		EPF2	Edge port 1 flag 2	Write EPF2 = 1

5. RISC Core Introduction

This document describes the functionality of the RISC Core microprocessor, which is based on M*Core instruction set/architecture and designed for extremely low-power and cost-sensitive embedded control applications.

For the smaller size and power dissipation, RISC Core is built on a new 3-stage pipeline von Neumann architecture.

RISC Core also integrates an EIC(embedded interrupt controller) to reduce system area.

The external bus interface protocol is AHB-lite. More configurable options are available in RISC Core design. Leveraging these configurable options, tradeoff among performance, functionality and cost are more flexible. The gate count of RISC Core varies from 12K to 20K with different configurations.

5.1. Features

The main features of the RISC Core are as follows:

- 32bit load/store reduced instruction set computer (RISC) architecture with fixed 16bit instruction length
- 16 entry 32bit general-purpose register file
- Efficient 3-stage execution pipeline, hidden from application software
- Single-cycle instruction execution for many Instructions, three cycles for branches
- Support for byte/halfword/word memory accesses
- Embedded interrupt controller, support nested vector interrupts and low power mode wakeup
- Single-cycle 32bit x 32bit hardware integer multiplier array
- 3~13 cycles hardware integer divider array
- AHB-lite external bus

5.2. Microarchitecture Summary

The RISC Core utilizes a 3-stage pipeline for instruction execution. The instruction fetch, instruction decode/register file read, execute/writeback stages operate in an overlapped fashion, allowing single clock instruction execution for most instructions.

16 general-purpose registers are provided for source operands and instruction results. Register R15 is used as the link register to hold the return address for subroutine calls, and Register R0 is associated with the current stack pointer value by convention.

A dual entry 32bit instruction buffer is provided to allow instruction prefetching to obtain two instructions per clock cycle from memory with a maximum of three buffered instructions, thus reducing or eliminating bus resource conflicts with data memory accesses. The unified bus structure is sufficient to sustain both instruction and data bandwidth requirements without resorting to expensive dual bus structures.

Memory load and store operations are provided for byte, halfword, and word (32bit) data with automatic zero extension of byte and halfword load data These instructions can be pipelined to allow effective single cycle throughput for short sequences. Data dependent operations can complete in two clock cycles. Load and store multiple register instructions allow low overhead context save and restore operations; these instructions can execute in (N+1) clock cycles, where N is the numbers of registers to transfer.

A single condition code/carry (C) bit is provided for condition testing and for use in implementing arithmetic

and logical operations greater than 32bits. Typically, the C bit is set only by explicit test/comparison operations, not as a side-effect of normal instruction operation. Exceptions to this rule occur for specialized operations where it is desirable to combine condition setting with actual computation.

5.3. Programming Model

The RISC Core programming model is defined separately for two privilege modes: supervisor and user. HPROT[1] bit is used to indicate the privilege modes.

Programs access registers based on the indicated mode. User programs can only access registers specific to the user mode; system software executing in the supervisor mode can access all registers, using the control registers to perform supervisory functions. User programs are thus restricted from accessing privileged information, and the operating system performs management and service tasks for the user programs by coordinating their activities.

All instructions execute in either mode. User program can also execute stop, doze, or wait instructions. The trap #n instructions provide controlled access to operating system services for user programs. To prevent a user program from entering the supervisor mode except in a controlled manner, instructions that can alter the S-bit in the program status register (PSR) are privileged.

When the S-bit in the PSR is set, the processor executes instructions in the supervisor mode. Bus cycles associated with an instruction indicate either supervisor or user access depending on the mode.

The processor utilizes the user programming model when it is in normal user mode processing. During exception processing, the processor changes from user to supervisor mode. Exception processing saves the current value of the PSR to stack memory and then sets the S bit in the PSR, forcing the processor into the supervisor mode. To return to the previous operating mode, a system routine may execute the RTE (return from exception) instruction, causing the instruction pipeline to be flushed and refilled from the appropriate address space.

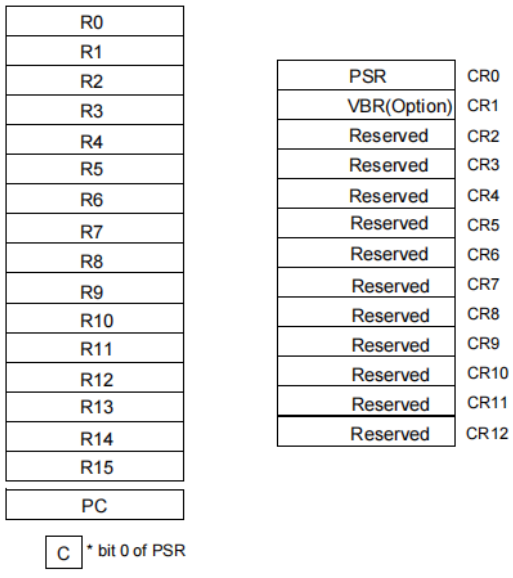


Figure 5-1: Programming Model

The registers depicted in the programming model (see Figure 5-1) provide operand storage and control. The user programming model consists of 16 general-purpose 32bit registers, the 32bit program counter (PC) and the Condition/Carry (C) bit. The C bit is implemented as bit 0 of the PSR. By convention, register R15 serves as the link register for subroutine calls, and register R0 is typically used as the current stack pointer.

5.4. Data Format Summary

The operand data formats supported by the integer unit are standard two's complement data formats. The operand size for each instruction is either explicitly encoded in the instruction (load/store instructions) or implicitly defined by the instruction operation (index operations, byte extraction). Typically, instructions operate on all 32bits of the source operand(s) and generate a 32bit result.

Memory may be viewed from either a Big Endian or Little Endian byte ordering perspective depending on the processor configuration (see Figure 5-2). In Big Endian mode (the default operating mode), the most significant byte (byte 0) of word 0 is located at address 0. For Little Endian mode, the most significant bye of word 0 is located at address Within registers, bits are numbered within a word starting with bit 31 as the most significant bit (see Figure 5-3). By convention, byte 0 of a register is the most significant byte regardless of Endian mode. This is only an issue when executing the xtrb[0-3] instructions.

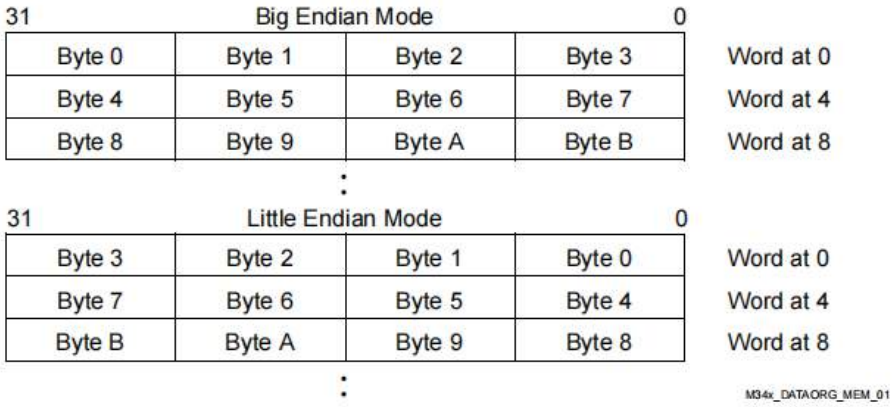


Figure 5-2: Data Organization in Memory

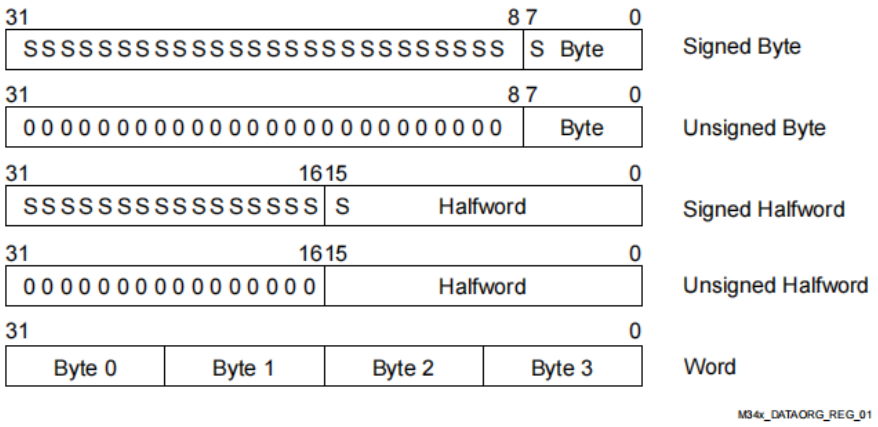


Figure 5-3: Data Organization in Registers

5.5. Operand Addressing Capabilities

RISC Core accesses all memory operands through load and store instructions, transferring data between the general-purpose registers (GPRs) and memory. Register + 4bit scaled displacement addressing mode is used for the load and store instructions to address byte, halfword, or word (32bit) data.

Load and store multiple instructions allow a subset of the 16 GPRs to be transferred to or from a base address pointed to by register R0 (the default stack pointer by convention).

Load and store register quadrant instructions use register indirect addressing to transfer a register quadrant to or from memory.

5.6. Instruction Set Overview

The instruction set is tailored to support high-level languages and is optimized for those instructions most commonly executed. A standard set of arithmetic and logical instructions is provided as well as instruction support for bit operations, byte extraction, data movement, control flow modification, and a small set of conditionally executed instructions which can be useful in eliminating short conditional branches.

Table 5-1 provides an alphabetized listing of the RISC Core instruction set.

Table 5-1: RISC Core Instruction Set

Mnemonic	Description
ABS	Absolute Value
ADDC	Add with C bit
ADDI	Add Immediate
ADDU	Add Unsigned
AND	Logical AND
ANDI	Logical AND Immediate
ANDN	AND NOT
ASR	Arithmetic Shift Right
ASRC	Arithmetic Shift Right, update C bit
BCLRI	Clear Bit
BF	Branch on Condition False
BGENI	Bit Generate Immediate
BGENR	Bit Generate Register
BKPT	Breakpoint
BMASKI	Bit Mask Immediate
BR	Branch
BREV	Bit Reverse
BSETI	Bit Set Immediate
BSR	Branch to Subroutine
BT	Branch on Condition True
CLRF	Clear Register on Condition False
CLRT	Clear Register on Condition True
CMPHS	Compare Higher or Same
CMPLT	Compare Less-Than
CMPLTI	Compare Less-Than Immediate

Mnemonic	Description
CMPNE CMPNEI	Compare Not Equal Compare Not Equal Immediate
DECF DECGT DECLT DECNE DECT DIVS[1] DIVU[1] DOZE	Decrement on Condition False Decrement Register and Set Condition if Result Greater-than Zero Decrement Register and Set Condition if Result Less-than Zero Decrement Register and Set Condition if Result Not Equal to Zero Decrement On Condition True Divide Signed Integers Divide Unsigned Integers Doze
FF1[1] INCF	Find First One Increment on Condition False
INCT IXH IXW	Increment On Condition True Index Halfword Index Word
JAVASW JMP JMPI JSR JSRI	Java interpreter switch Jump Jump Indirect Jump to Subroutine Jump to Subroutine Indirect
LD [BHW] LDM LDQ LRW LSL, LSR LSLC, LSRC LSLI, LSRI	Load Load Multiple Registers Load Register Quadrant Load Relative Word Logical Shift Left and Right Logical Shift Left and Right, update C bit Logical Shift Left and Right by Immediate
MFCR MOV MOVI MOVF MOVT MTCR MULSH MULT	Move from Control Register Move Move Immediate Move on Condition False Move on Condition True Move to Control Register Multiply signed Halfwords Multiply
MVC MVCV	Move C bit to Register Move Inverted C bit to Register
NOT	Logical Complement
OR	Logical Inclusive-OR

Mnemonic	Description
ROTLI RSUB RSUBI	Rotate Left by Immediate Reverse Subtract Reverse Subtract Immediate
RTE RFI	Return from Exception Return from Interrupt
SEXTB SEXTH ST[BHW] STM STQ	Sign-extend Byte Sign-extend Halfword Store Store Multiple Registers Store Register Quadrant
STOP SUBC SUBU SUBI	Stop Subtract with C bit Subtract Subtract Immediate
SYNC	Synchronize
TRAP TST TSTNBZ	Trap Test Operands Test for No Byte Equal Zero
WAIT	Wait
XOR XSR XTRB0 XTRB1 XTRB2 XTRB3	Exclusive OR Extended Shift Right Extract Byte 0 Extract Byte 1 Extract Byte 2 Extract Byte 3
ZEXTB ZEXTH	Zero-extend Byte Zero-extend Halfword

NOTE:

1. Not implemented in the current version.

6. Embedded Programmable Timer (EPT)

6.1. Introduction

Embedded programmable timer(EPT) is a 24bit timer that provides precise interrupts at regular intervals with minimal processor intervention. The timer can either count down from the reload value, or it can be a free-running down-counter.

EPT interrupt can trigger an exception(vector number = 24).

EPT module can be removed by clearing parameter “EPT” to 0 for reducing core gate count.

6.2. Memory Map and Registers

6.2.1. Memory Map

EPT base address is defined as EIC_BASEADDR + 0x1000. The default based address is 0xE000_1000. Table 6-1 shows the offset address of EPT registers. EPT module occupies 4K address area.

Table 6-1: Programmable Timer Module Memory Map

Offset Address	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0	Access
0x0000_0000	EPT Control and Status Register (EPTCSR)				S/U
0x0000_0004	EPT Reload Register (EPTRLD)				S/U
0x0000_0008	EPT Count Register (EPTCNT)				S/U
0x0000_000C	Reserved				S/U

6.2.2. Registers

This subsection contains a description of the EPT module registers.

6.2.2.1. EPT Control Status Register

Offset Address: 0x0000

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	CNTFLAG
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CLKSRC	INTEN	CNTEN
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 6-1: EPT Control Status Register (EPTCSR)

CNTFLAG — Count Down to 0 flag

The read-only bit indicates timer counted to 0. It will be reset by HRESETn.
1 = The timer counted down to 0.
0 = The timer is still counting down.

CLKSRC — Count clock source select

The read/write bit is used to select the count clock source. It will be reset by HRESETn.
1 = Core clock.
0 = External reference clock.

INTEN — EPT Interrupt Request Enable

The read/write bit is used to enable EPT's interrupt when timer counted down to 0, It will be reset by RESET.
1 = EPT exception request occurs when timer counted down to 0.
0 = EPT exception request will not occur when timer counted down to 0.

CNTEN — Counter Enable

The read/write bit is used to enable EPT's counter. It will be reset by RESET.
1 = Counter enabled
0 = Counter disabled

6.2.2.2. EPT Reload Register

Offset Address: 0x0004

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	x	x	x	x	x	x	x	x

	23	22	21	20	19	18	17	16
R	RLD[23]	RLD[22]	RLD[21]	RLD[20]	RLD[19]	RLD[18]	RLD[17]	RLD[16]
W								
RESET:	x	x	x	x	x	x	x	x

	15	14	13	12	11	10	9	8
R	RLD[15]	RLD[14]	RLD[13]	RLD[12]	RLD[11]	RLD[10]	RLD[9]	RLD[8]
W								
RESET:	x	x	x	x	x	x	x	x

	7	6	5	4	3	2	1	0
R	RLD[7]	RLD[6]	RLD[5]	RLD[4]	RLD[3]	RLD[2]	RLD[1]	RLD[0]
W								
RESET:	x	x	x	x	x	x	x	x

= Writes have no effect and the access terminates without a transfer error exception.

Figure 6-2: EPT Reload Register (EPTRLD)

RLD[23:0] — Reload Value

The read/write RLD[23:0] field specifies the reload value when timer counted down to 0. The register has no reset value. The RLD value can be any value in the range 0x0000_0001 ~ 0x00FF_FFFF. Value 0 has no effect. To generate a period timer with N clock cycles, set RLD to N-1.

6.2.2.3. EPT Count Register

Offset Address: 0x0008

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	x	x	x	x	x	x	x	x

	23	22	21	20	19	18	17	16
R	CNT[23]	CNT[22]	CNT[21]	CNT[20]	CNT[19]	CNT[18]	CNT[17]	CNT[16]
W								
RESET:	x	x	x	x	x	x	x	x

	15	14	13	12	11	10	9	8
R	CNT[15]	CNT[14]	CNT[13]	CNT[12]	CNT[11]	CNT[10]	CNT[9]	CNT[8]
W								
RESET:	x	x	x	x	x	x	x	x

	7	6	5	4	3	2	1	0
R	CNT[7]	CNT[6]	CNT[5]	CNT[4]	CNT[3]	CNT[2]	CNT[1]	CNT[0]
W								
RESET:	x	x	x	x	x	x	x	x

= Writes have no effect and the access terminates without a transfer error exception.

Figure 6-3: EPT Counter Register (EPTCNT)

CNT[23:0] — EPT Counter Value

The read-only register indicates the current count value of EPT timer. The register has not reset value.

Reads will return the current value of EPT counter. A write of any value to this register will clear the counter value to 0 and also clears the CNTFLAG to 0.

6.3. Function Description

When Enabled, EPT count down from the value set by RLD to zero, and wrap reloads the value in RLD on the next clock cycle, then down-counts by subsequent clock cycles, writing value of zero to RLD disables the counter on next wrap. When it counted to zero, the CLFAG bit will set to 1, then the ETP will trig the ETP interrupt if INTEN was enabled.

Reading CSR clears the CFLAG bit to 0. Write any value to CNT also clears the CFLAG bit to 0.

6.3.1. Count Timing

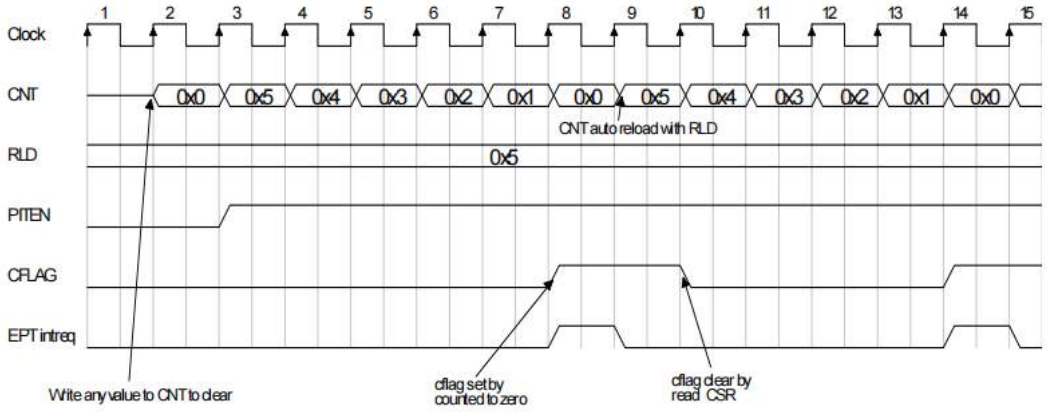


Figure 6-4: EPT Count Timing

7. System Integration Module (SIM)

7.1. Introduction

The System Integration Module (SIM) controls pad configuration, general purpose I/O (GPIO), internal peripheral multiplexing. The pad configuration block controls the static electrical characteristics of I/O pins. The GPIO block provides uniform and discrete input/output control of the MCU I/O pins. The SIM is accessed by the core through the peripheral bus. Figure 7-1 provides a block diagram of the SIM and its interfaces to other system components. Note that Pad Interface/Pad Ring block, and Peripheral I/O Channels are external to the SIM.

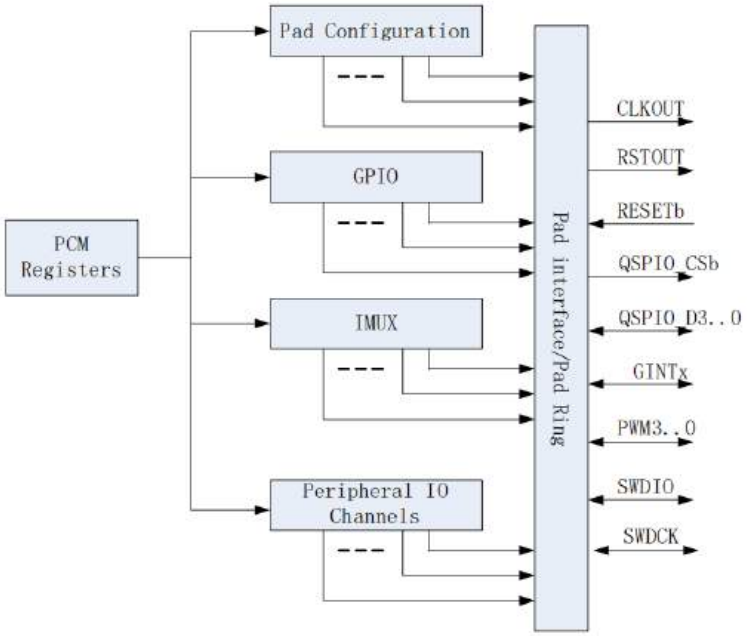


Figure 7-1: SIM Block Diagram

7.2. Features

- System configuration
  - — Pad configuration control
- GPIO
  - — GPIO function on 31 I/O pins
  - — Dedicated input and output registers for each GPIO pin
- External input multiplexing
  - — Allows selection of UART Transmitter/Receiver Pin
  - — Allows selection of CANbus Receiver Pin
  - — Allows selection of I2C Receiver Pin
- Configurate wakeup function
- Configurate encrypt/de-crypt function when read data/instruction from external flash memory

7.3. Modes of Operation

7.3.1. Normal Mode

In normal mode, the SIM provides the register interface and logic that controls port configuration, input multiplexing, and GPIO.

7.3.2. Debug Mode

SIM operation in debug mode is identical to operation in normal mode.

7.4. Memory Map

Table 7-1 is the address map for the SIM registers.

Table 7-1: SIM Address Map

Offset Address	Use	Bits Per Register
0x0000 ~ 0x003F	Pad Configuration Register 0 (SIM_PCR0) — Pad Configuration Register 30 [1]	16
0x0040 ~ 0x03FF	Reserved	—
0x0400 ~ 0x041F	GPIO Pin Data Output Register 0 ~ 3 (SIM_GPDO0_3) — GPIO Pin Data Output Register 28 ~ 31(SIM_GPDO28_31)[1]	8
0x0420 ~ 0x04FF	Reserved	—
0x0600 ~ 0x061F	GPIO Pin Data Input Register 0 ~ 3 (SIM_GPDI0_3) — GPIO Pin Data Input Register 28 ~ 31 (SIM_GPDI28_31)[1]	8
0x0620 ~ 0x06FF	Reserved	—
0x0800	SIM_PGPDO0 — Parallel GPIO Pin Data Output Register 0	32
0x0804 ~ 0x08FF	Reserved	—
0x0900	SIM_PGPDI0 — Parallel GPIO Pin Data input Register 0	32
0x0904 ~ 0x08FF	Reserved	—
0x0A00	SIM_MPGPDO0 — Masked Parallel GPIO Pin Data Output Register 0	32
0x0A04	SIM_MPGPDO1 — Masked Parallel GPIO Pin Data Output Register 1	32
0x0A08 ~ 0x0BFF	Reserved	—
0x0C00	WKUPC — Wakeup Configuration Register	32
0x0C04	QSPIXIPCR — QSPI XIP Mode Configuration Register	32
0x0C08	QSPIKEYR — QSPI 32bit Key Register	32
0x0C0C ~ 0x0FFF	Reserved	—

NOTE: *Gaps exist in this memory space where I/O pins are not available in the specified package.*

7.5. Register Descriptions

7.5.1. Pad Configuration Registers (SIM_PCR)

The following subsections define the Pad Configuration Registers for all device pins that allow configuration of the pin function, direction, and static electrical attributes. The information presented pertains to which bits and fields are active for a given pin or group of pins, and the reset state of the register. Note that the reset state of PCRs given in the following sections is that prior to execution of the boot program. The boot program may change certain PCRs based on the reset configuration.

Figure 7-2 shows a sample PCR map. Please note the following:

- The register bit numbering order follows the most significant bit being bit 15. Field bit ranges are the opposite—the least significant bit is referred to as bit 0.
- Bits 0 through 3 and bits 12 through 13 are examples reserved bits. They are read-only and always return a value of 0.
- The PCRs are 16bit registers but may be read or written as 32bit values aligned on 32bit address boundaries.

Offset Address: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	PA		OBE[1]	IBE[2]	DSC[2:0]			ODE	0	HYS	WPE	WPS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

= Unimplemented or Reserved

NOTE:

1. *OBE bit is significant in GPIO*
2. *IBE bit is significant in GPIO*

Figure 7-2: Sample PCR Map

Table 7-2 lists and describes the fields contained in the PCRs. Not all fields appear in each PCR but each field has identical function in each register where it resides.

Table 7-2: SIM_PCR Field Descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Reserved</td><td>Reserved fields are indicated by shading in the register maps.</td></tr>
<tr><td>PA</td><td>Pin assignment
Selects the function of a multiplexed pad.

Table 0-1
<table><tr><th>PA Value[1]</th><th colspan="2">Pin Function</th></tr><tr><td>0b11</td><td>P</td><td>Primary function</td></tr><tr><td>0b01</td><td>A1</td><td>Alternate function</td></tr><tr><td>0b10</td><td>A2</td><td>Alternate function2</td></tr><tr><td>0b00</td><td>G</td><td>GPIO</td></tr></table></td></tr>
<tr><td>OBE[2,3]</td><td>Output buffer enable
Enables the pad as an output and drives the output buffer enable signal.

0: Disable output buffer for the pad.
1: Enable output buffer for the pad.</td></tr>
<tr><td>IBE[2,3]</td><td>Input buffer enable
Enables the pad as an input and drives the input buffer enable signal.

0: Disable input buffer for the pad.
1: Enable input buffer for the pad is enabled.

The IBE and OBE bit definitions are specific for each PCR. In cases where an I/O function is input or output only the IBE and OBE bits do not need to be set to enable the input or output respectively. In cases where an I/O function can be either an input and output, the IBE and OBE bits must be set accordingly (IBE = 1 for input, and OBE = 1 for output). For I/O functions that change direction dynamically, such as the external data bus, switching between input and output is handled internally and the IBE and OBE bits have no effect.

For all PCRs where GPIO function is available on the pin, if the pin is configured as an output and the IBE bit is set, the actual value of the pin will be reflected in the corresponding GPDIx_x register. Negating the IBE bit when the pin is configured as an output will reduce noise and power consumption.</td></tr>
</table>

Field	Description
DSC[2:0][4]	Drive strength control Controls the pad drive strength. Drive strength control pertains to pins with the fast I/O pad type. 000: 6mA drive strength 001: 13mA drive strength 010: 19mA drive strength 011: 26mA drive strength 100: 32mA drive strength 101: 39mA drive strength 110: 45mA drive strength 111: 52mA drive strength
HYS[5]	Input hysteresis Controls whether hysteresis is enabled for the pad. 0: Disable hysteresis for the pad. 1: Enable hysteresis for the pad.
WPE[6]	Weak pullup/down enable Controls whether the weak pullup/down devices are enabled/disabled for the pad. Pullup/down devices are enabled by default. 0: Disable weak pull device for the pad. 1: Enable weak pull device for the pad.
WPS[6]	Weak pullup/down select Controls whether weak pullup or weak pulldown devices are used for the pad when weak pullup/down devices are enabled. The WKPCFG pin determines whether pullup or pulldown devices are enabled during reset. The WPS bit determines whether weak pullup or pulldown devices are used after reset, or for pads in which the WKPCFG pin does not determine the reset weak pullup/down state. 0: Pulldown is enabled for the pad. 1: Pullup is enabled for the pad.

NOTE:

1. *Depending on the register, the PA field size can vary in length. For PA fields having fewer than three bits, remove the appropriate number of leading zeroes from these values.*
2. *In cases where an I/O function is either input-only or output-only the IBE and OBE bits do not need to be set to enable pin I/O.*
3. *For I/O functions that change direction dynamically, such as the SCL/SDA, switching between input and output is handled peripheral internally and the IBE and OBE bits are ignored.*
4. *If a pin is configured as an input, the ODE, SRC, and DSC bits do not apply. SRC is meaningless in this device.*
5. *If a pin is configured as an output, the HYS bit does not apply. HYS is meaningless in this device.*

6. *When a pin is configured as an output, the weak internal pull up/down is disabled regardless of the WPE or WPS settings in the PCR.*

There are a number of input signals on the device that have multiple external pins as input sources. Only one input source can be active at any time for these pins. To achieve this, there is a pre-defined priority for the PCR registers controlling these pins, which is used to mux the input sources and allow only one active input. The multiple source inputs with PCR priority is given in Table 7-3.

Table 7-3: PCR Priority for Multiple Source Input

Function	PCR Number Highest Priority -> Lowest Priority	
TXD1	2	12
RXD1	3	13
CAN_RX	13	30
SCL	14	29
SDA	15	30

Table 7-4: SIM_PCRn Settings

PCRn	Offset Address	Primary Function	Alt1	Alt2	GPIO	SIM_PCRn Default Settings							
						PA[1:0]	OBE	IBE	DSC	ODE	HYS	WPE	WPS
0	0x02	PWM0	GINT0[0]	COMP0_OUT	GPIO0	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
1	0x00	PWM1	GINT0[1]		GPIO1	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
2	0x06	PWM2	GINT0[2]	TXD1	GPIO2	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
3	0x04	PWM3	GINT0[3]	RXD1	GPIO3	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
4	0x0A	TK0	GINT0[4]		GPIO4	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
5	0x08	TK1	GINT0[5]		GPIO5	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
6	0x0E	RSTOUT	GINT0[6]	TK2	GPIO6	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
7	0x0C	CLKOUT	GINT0[7]	TK3	GPIO7	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
8	0x12	EBI_DB0	SPI0_CS#		GPIO8	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
9	0x10	EBI_DB1	SPI0_MOSI		GPIO9	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
10	0x16	EBI_DB2	SPI0_MISO		GPIO10	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
11	0x14	EBI_DB3	SPI0_SCK		GPIO11	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
12	0x1A	EBI_DB4	TXD1	CAN_TX	GPIO12	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
13	0x18	EBI_DB5	RXD1	CAN_RX	GPIO13	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
14	0x1E	EBI_DB6	SCL		GPIO14	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
15	0x1C	EBI_DB7	SDA		GPIO15	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
16	0x22	SWDIO	GINT1[0]	AIN0	GPIO16	2'b11	1'b0	1'b0	3'h4	1'b0	1'b1	1'b1	1'b1
17	0x20	SWDCK	GINT1[1]	AIN1	GPIO17	2'b11	1'b0	1'b0	3'b0	1'b0	1'b1	1'b1	1'b1
18	0x26	AIN2	GINT1[2]		GPIO18	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
19	0x24	QSPI0_CS#			GPIO19	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
20	0x2A	QSPI0_D0			GPIO20	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
21	0x28	QSPI0_D1			GPIO21	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
22	0x2E	QSPI0_D2			GPIO22	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
23	0x2C	QSPI0_D3			GPIO23	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
24	0x32	QSPI0_SCK			GPIO24	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
25	0x30	EBI_CS#	SPI1_CS#		GPIO25	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0

PCRn	Offset Address	Primary Function	Alt1	Alt2	GPIO	SIM_PCRn Default Settings							
						PA[1:0]	OBE	IBE	DSC	ODE	HYS	WPE	WPS
26	0x36	EBI_WR#	SPI1_MOSI		GPIO26	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
27	0x34	EBI_RD#	SPI1_MISO		GPIO27	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
28	0x3A	EBI_RS	SPI1_SCK		GPIO28	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
29	0x38	TXD0	SCL	CANTX	GPIO29	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
30	0x3E	RXD0	SDA	CANRX	GPIO30	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
31	0x3C	CLKMODE				2'b11	1'b0	1'b0	3'b0	1'b0	1'b1	1'b1	1'b1

7.5.2. GPIO Pin Data Output Registers (SIM_GPDO0_3 ~ SIM_GPDO28_31)

The definition of the SIM_GPDOx_x registers is given in Figure 7-3 to Figure 7-10. Each of the 32 PDO (31 used) bits correspond to the pin with the same GPIO pin number. For example, PDO0 is the pin data output bit for the PWM0_GINT0[0]_GPIO0 pin, and PDO30 is the pin data output bit for the RXD0_SDA_CAN_RX_GPIO30 pin. Gaps exist in this memory space where the pin is not available in the package.

The SIM_GPDOx_x registers are written to by software to drive data out on the external GPIO pin. Each byte of a register drives a single external GPIO pin, which allows the state of the pin to be controlled independently from other GPIO pins. Writes to the SIM_GPDOx_x registers have no effect on pin states if the pins are configured as inputs by the associated Pad Configuration Registers. The SIM_GPDOx_x register values are automatically driven to the GPIO pins without software update if the direction of the GPIO pins is changed from input to output.

Writes to the SIM_GPDOx_x registers have no effect on the state of the corresponding pins when the pins are configured for their primary or alternate function by the corresponding PCR.

Offset Address: 0x0400

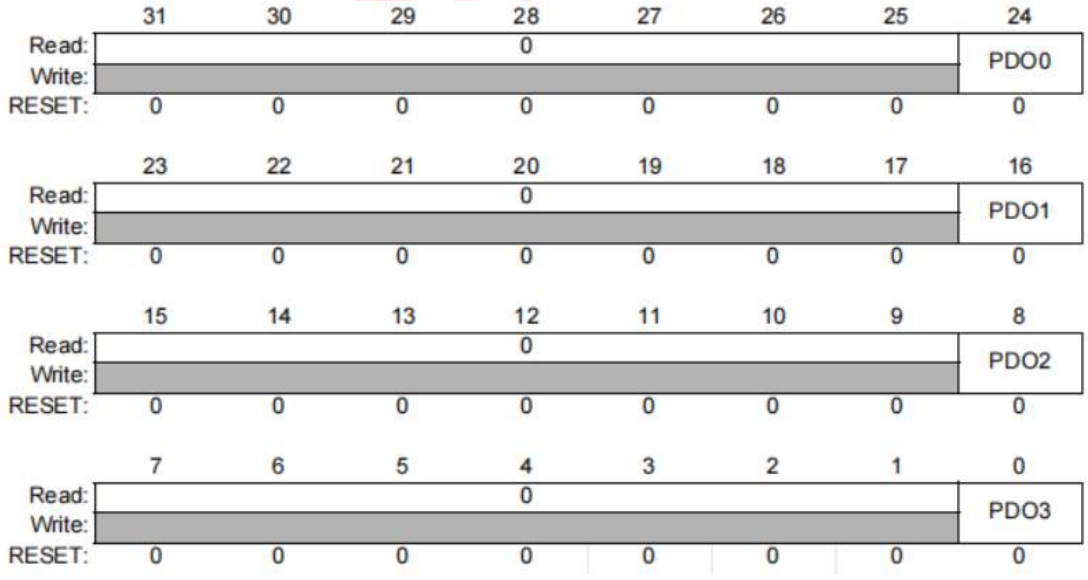


Figure 7-3: GPIO Pin Data Out Register 0 ~ 3 (SIM_GPDO0 ~ SIM_GPDO3)

Offset Address: 0x0404

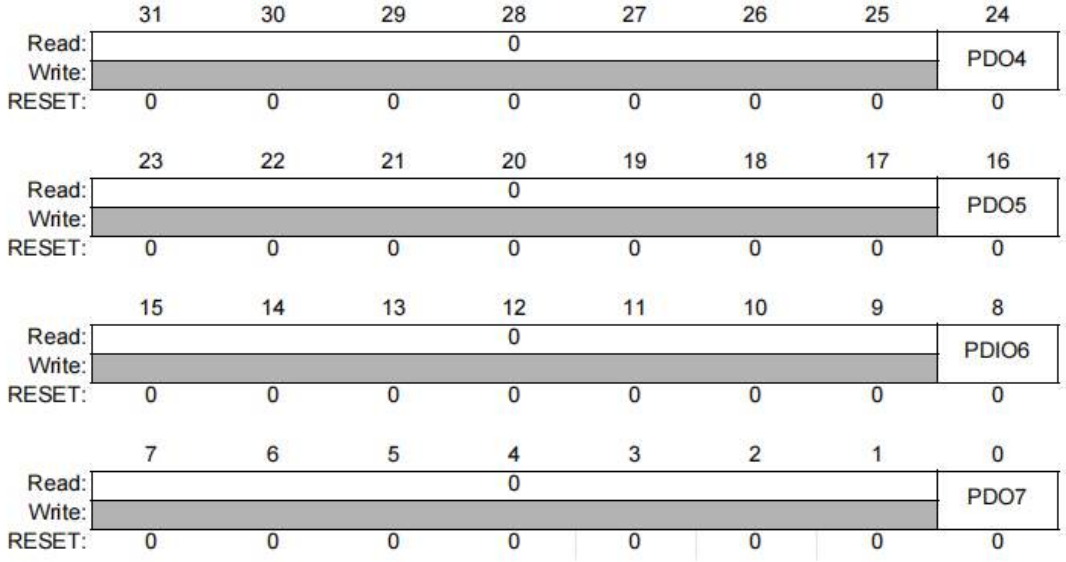


Figure 7-4: GPIO Pin Data Out Register 4 ~ 7 (SIM_GPDO4 ~ SIM_GPDO7)

Offset Address: 0x0408

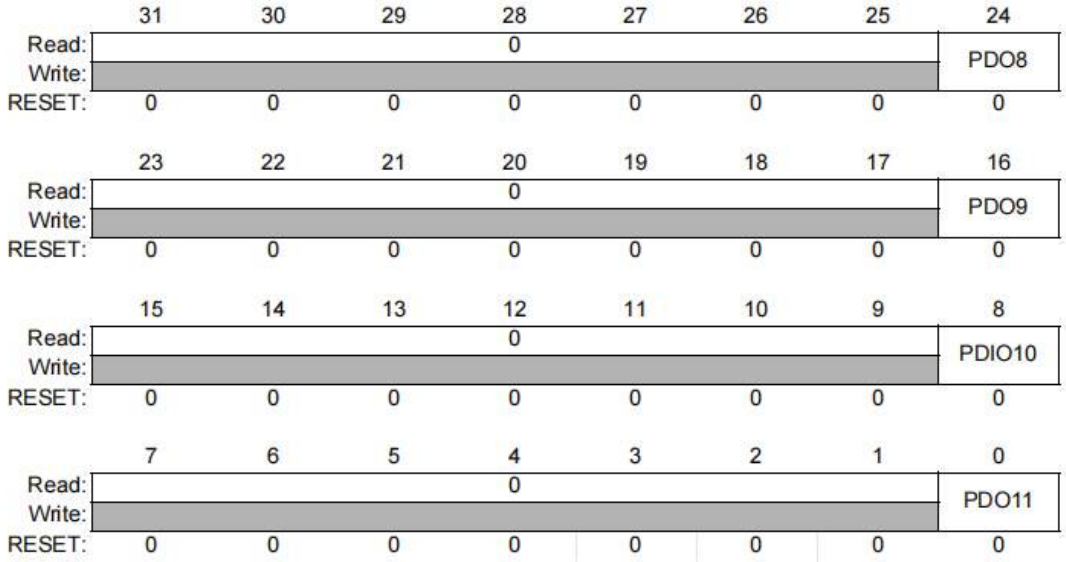


Figure 7-5: GPIO Pin Data Out Register 8 ~ 11 (SIM_GPDO8 ~ SIM_GPDO11)

Offset Address: 0x040C

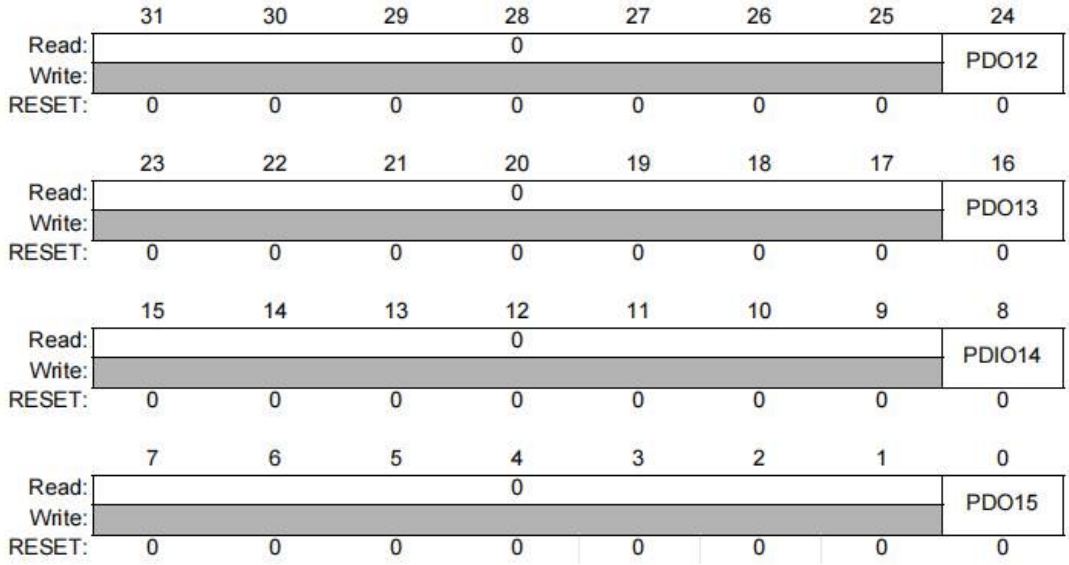


Figure 7-6: GPIO Pin Data Out Register 12 ~ 15 (SIM_GPDO12 ~ SIM_GPDO15)

Offset Address: 0x0410

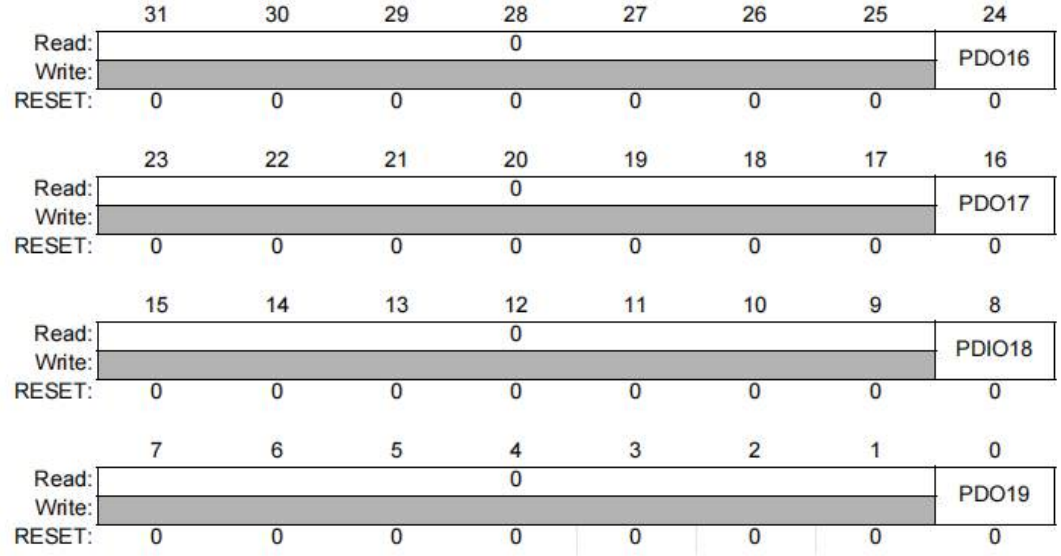


Figure 7-7: GPIO Pin Data Out Register 16 ~ 19 (SIM_GPDO16 ~ SIM_GPDO19)

Offset Address: 0x0414

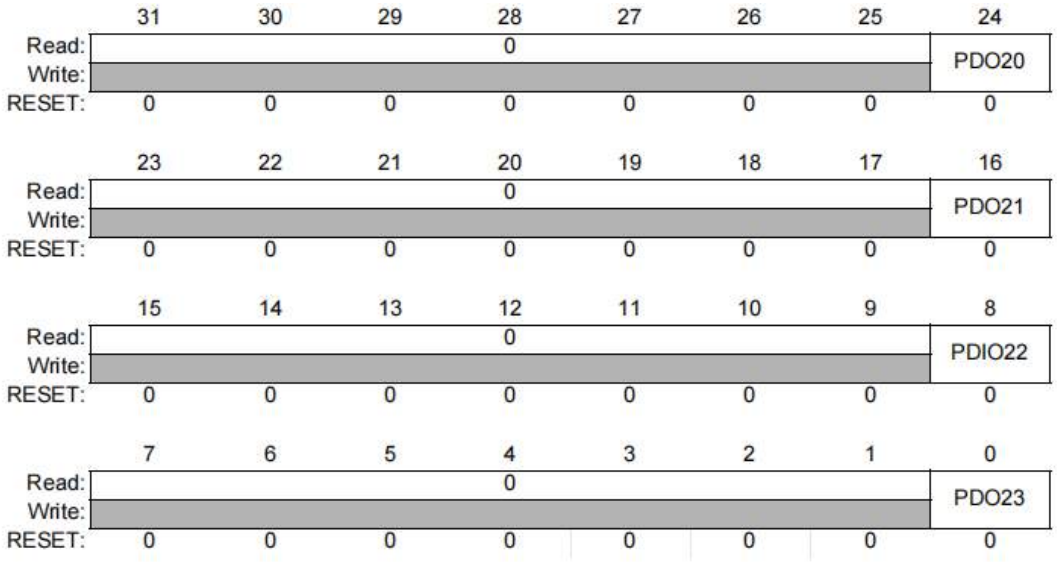


Figure 7-8: GPIO Pin Data Out Register 20 ~ 23 (SIM_GPDO20 ~ SIM_GPDO23)

Offset Address: 0x0418

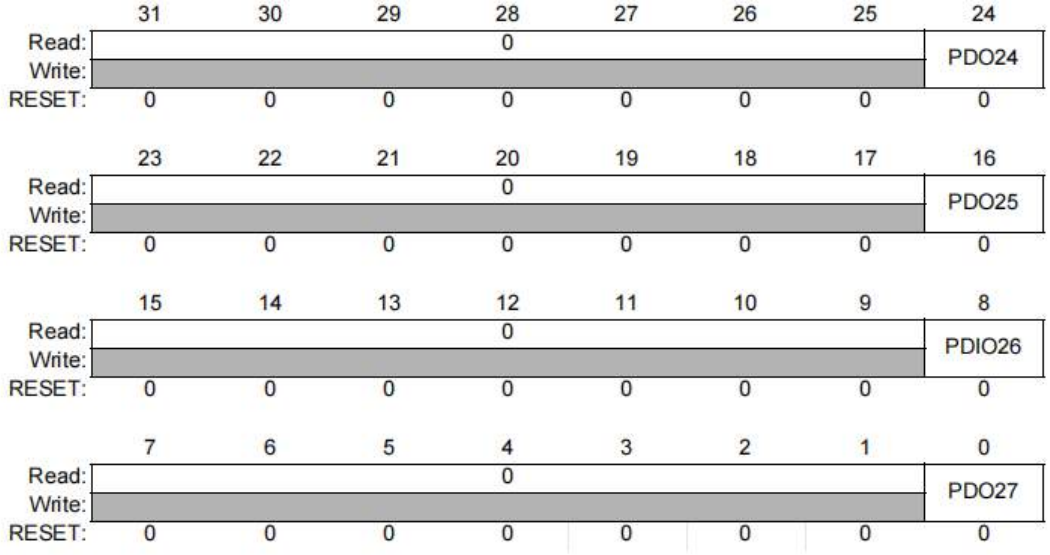


Figure 7-9: GPIO Pin Data Out Register 24 ~ 27 (SIM_GPDO24 ~ SIM_GPDO27)

Offset Address: 0x041C

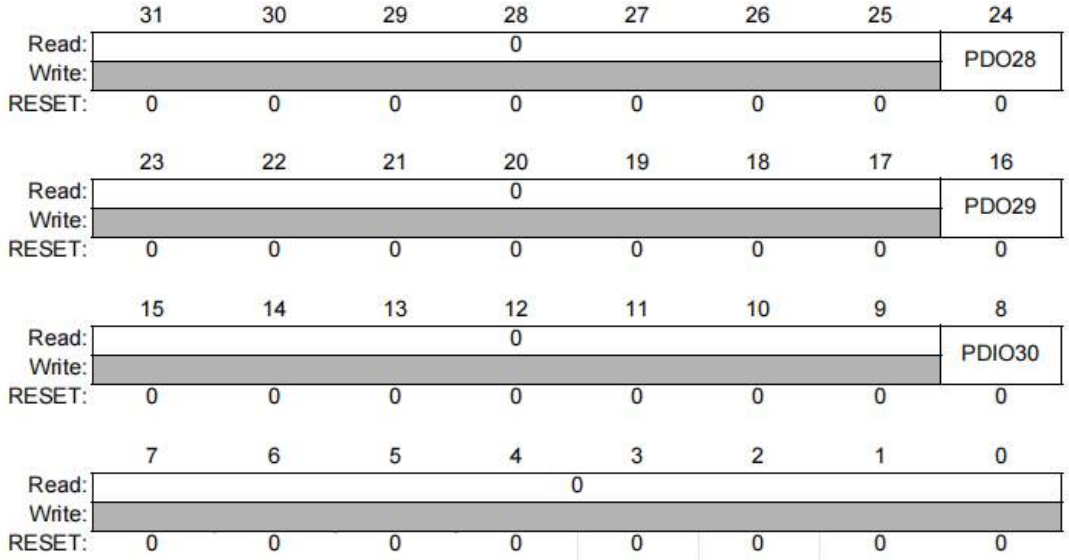


Figure 7-10: GPIO Pin Data Out Register 28 ~ 31 (SIM_GPDO28 ~ SIM_GPDO31)

Table 7-5: SIM_GPDOx Register Field Descriptions

Field	Description
PDOx	Pin Data Out This bit stores the data to be driven out on the external GPIO pin associated with the register. 1: VOH is driven on the external GPIO pin when the pin is configured as an output. 0: VOL is driven on the external GPIO pin when the pin is configured as an output.

7.5.3. GPIO Pin Data Input Registers (SIM_GPDI0_3 ~ SIM_GPDI_28_31)

The definition of the SIM_GPDIx_x registers is given in Figure 7-11 to Figure 7-18.Each of the 32 GPDI(31 used) bits correspond to the pin with the same GPIO pin number. For example, GPDI0 is the pin data input bit for the PWM0_GINT0[0]_GPIO0 pin, and PDI30 is the pin data input bit for the RXD0_SDA_CAN_RX_GPIO30 pin. Gaps exist in this memory space where the pin is not available in the package.

The SIM_GPDIx_x registers are read-only registers that allow software to read the input state of an external GPIO pin. Each byte of a register represents the input state of a single external GPIO pin. If the GPIO pin is configured as an output, and the input buffer enable (IBE) bit is set in the associated Pad Configuration Register, the SIM_GPDIx_x register reflects the actual state of the output pin.

Offset Address: 0x0600

	31	30	29	28	27	26	25	24
Read:				0				PDI0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:				0				PDI1
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:				0				PDI2
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:				0				PDI3
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 7-11: GPIO Pin Data in Register 0 ~ 3 (SIM_GPDI0 ~ SIM_GPDI3)

Offset Address: 0x0604

	31	30	29	28	27	26	25	24
Read:				0				PDI4
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:				0				PDI5
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:				0				PDI6
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:				0				PDI7
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 7-12: GPIO Pin Data in Register 4 ~ 7 (SIM_GPDI4 ~ SIM_GPDI7)

Offset Address: 0x0608

	31	30	29	28	27	26	25	24
Read:	0							PDI8
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI9
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI10
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI11
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 7-13: GPIO Pin Data in Register 8 ~ 11 (SIM_GPDI8 ~ SIM_GPDI11)

Offset Address: 0x060C

	31	30	29	28	27	26	25	24
Read:	0							PDI12
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI13
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI14
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI15
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 7-14: GPIO Pin Data in Register 12 ~ 15 (SIM_GPDI12 ~ SIM_GPDI15)

Offset Address: 0x0610

	31	30	29	28	27	26	25	24
Read:	0							PDI16
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI17
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI18
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI19
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 7-15: GPIO Pin Data in Register 16 ~ 19 (SIM_GPDI16 ~ SIM_GPDI19)

Offset Address: 0x0614

	31	30	29	28	27	26	25	24
Read:	0							PDI20
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI21
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI22
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI23
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 7-16: GPIO Pin Data in Register 20 ~ 23 (SIM_GPDI20 ~ SIM_GPDI23)

Offset Address: 0x0618

	31	30	29	28	27	26	25	24
Read:				0				PDI24
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:				0				PDI25
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:				0				PDI26
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:				0				PDI27
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 7-17: GPIO Pin Data in Register 24 ~ 27 (SIM_GPDI24 ~ SIM_GPDI27)

Offset Address: 0x061C

	31	30	29	28	27	26	25	24
Read:				0				PDI28
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:				0				PDI29
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:				0				PDI30
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:					0			
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 7-18: GPIO Pin Data in Register 28 ~ 31 (SIM_GPDI28 ~ SIM_GPDI31)

Table 7-6: SIM_GPDIx Register Field Descriptions

Field	Description
PDIx	Pin Data In This bit stores the input state on the external GPIO pin associated with the register. 1: Signal on pin is greater than or equal to VIH. 0: Signal on pin is less than or equal to VIL.

7.5.4. Parallel GPIO Pin Data Output Register (SIM_PGPDO0)

The PGPDOx registers are written to by software to drive data out on the external GPIO pin. These registers access the same GPIO pins accessed by SIM_GPDO0 ~ SIM_GPDO31 bit registers. The SIM_GPDO registers should map directly to these registers. For example, SIM_PGPDO0 bit 31 is SIM_GPDO0 bit 7, SIM_PGPDO0 bit 30 is SIM_GPDO1 bit 7 and so on. Figure 7-19 is a lookup table correlating the SIM_GPDOx and SIM_PGPDOn registers.

Offset Address: 0x0800

	31	30	29	28	27	26	25	24
R W	PGPDO 0	PGPDO 1	PGPDO 2	PGPDO 3	PGPDO 4	PGPDO 5	PGPDO 6	PGPDO 7
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R W	PGPDO 8	PGPDO 9	PGPDO 10	PGPDO 11	PGPDO 12	PGPDO 13	PGPDO 14	PGPDO 15
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R W	PGPDO 16	PGPDO 17	PGPDO 18	PGPDO 19	PGPDO 20	PGPDO 21	PGPDO 22	PGPDO 23
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	PGPDO 24	PGPDO 25	PGPDO 26	PGPDO 27	PGPDO 28	PGPDO 29	PGPDO 30	PGPDO 31
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 7-19: Parallel GPIO Pin Data Output Register (SIM_PGPDO0)

Table 7-7: SIM_PGPDO0 Field Descriptions

Field	Description
PGPDO*x*	Pin Data Out This bit stores the data to be driven out on the external GPIO pin associated with the register. 1: VOH is driven on the external GPIO pin when the pin is configured as an output. 0: VOL is driven on the external GPIO pin when the pin is configured as an output.

7.5.5. Parallel GPIO Pin Data Input Register (SIM_PGPDI0)

The GPDIx registers are read-only registers that allow reading of the input state of an external GPIO pin. These registers access the same GPIO pins accessed by SIM_GPDI0 - SIM_GPDI31 bit registers. The SIM_GPDI registers should map directly to these registers. See 7.5.4 for a lookup table and examples.

The GPIO read/write should decode logical addresses to the same physical address of the normal GPIO. This way both the GPDI and corresponding PGPDI register should be updated on a pin state change when the IBE is asserted in the corresponding PCR.

Offset Address: 0x0900

	31	30	29	28	27	26	25	24
R	PGPDI 0	PGPDI 1	PGPDI 2	PGPDI 3	PGPDI 4	PGPDI 5	PGPDI 6	PGPDI 7
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	PGPDI 8	PGPDI 9	PGPDI 10	PGPDI 11	PGPDI 12	PGPDI 13	PGPDI 14	PGPDI 15
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	PGPDI 16	PGPDI 17	PGPDI 18	PGPDI 19	PGPDI 20	PGPDI 21	PGPDI 22	PGPDI 23
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PGPDI 24	PGPDI 25	PGPDI 26	PGPDI 27	PGPDI 28	PGPDI 29	PGPDI 30	PGPDI 31
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 7-20: Parallel GPIO Pin Data Input Register(SIM_PGPDI0)

Table 7-8: SIM_PGPDI0 Field Descriptions

Field	Description
PGPDI*x*	Pin Data In This bit stores the input state on the external GPIO pin associated with the register. 1: Signal on pin is greater than or equal to VIH. 0: Signal on pin is less than or equal to VIL.

7.5.6. Masked Parallel GPIO Pin Data Output Register (SIM_MPGPDO0 - SIM_MPGPDO1)

The MPGPDOx registers are written to by software to drive data out on the external GPIO pin (they are write-only registers and reading them will return 0; reading must be done by accessing the corresponding SIM_GPDO register). These registers access the same GPIO pins accessed by SIM_GPDO0–SIM_GPDO31 bit registers. The most significant 16bits in the SIM_MPGPDO registers should map directly to these registers. For example, SIM_MPGPDO0 bit 0 is SIM_GPDO15 bit 7, SIM_MPGPDO0 bit 1 is SIM_GPDO14 bit 7,...., SIM_MPGPD1 bit 0 is SIM_GPDO31 bit 7. The least significant sixteen bits are the corresponding values to be written at GPIO pins defined by MASK field. The masked parallel GPIO read/write should be decode the logical addresses to the same physical address of the normal GPIO.

Offset Address: 0x0A00

	31	30	29	28	27	26	25	24
R								
W	MASK0	MASK1	MASK2	MASK3	MASK4	MASK5	MASK6	MASK7
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R								
W	MASK8	MASK9	MASK10	MASK11	MASK12	MASK13	MASK14	MASK15
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R								
W	DATA0	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R								
W	DATA8	DATA9	DATA10	DATA11	DATA12	DATA13	DATA14	DATA15
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 7-21: Masked Parallel GPIO Pin Data Output Register (SIM_MPGPDO0)

Offset Address: 0x0A04

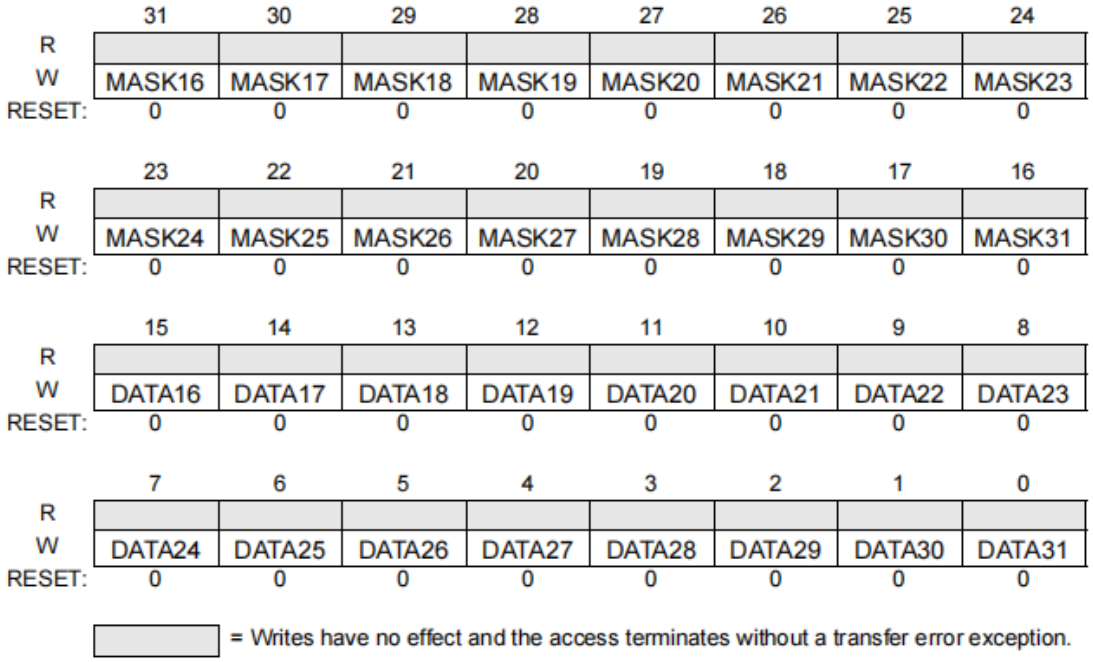


Figure 7-22: Masked Parallel GPIO Pin Data Output Register (SIM_MPGPDO1)

Table 7-9: SIM_MPGPDO0 ~ SIM_MPGPDO31 Field Descriptions

Field	Description
MASK*x*	Pin Data Out. Controls the write access to the corresponding GPDO. 0: Previous value defined by GPDO is maintain. 1: Corresponding GPDO is written with value defined by DATA field.
DATAx	Pin Data Out. Stores the data to be driven out on the external GPIO pin controlled by this register. 0: Logic low value is driven on the pad interface data out signal for the corresponding GPIO pin when the pin is configured as an output. 1: Logic high value is driven on the pad interface data out signal for the corresponding GPIO pin when the pin is configured as an output.

7.5.7. WKUPC — Wakeup Configuration Register

Offset Address: 0x0C00

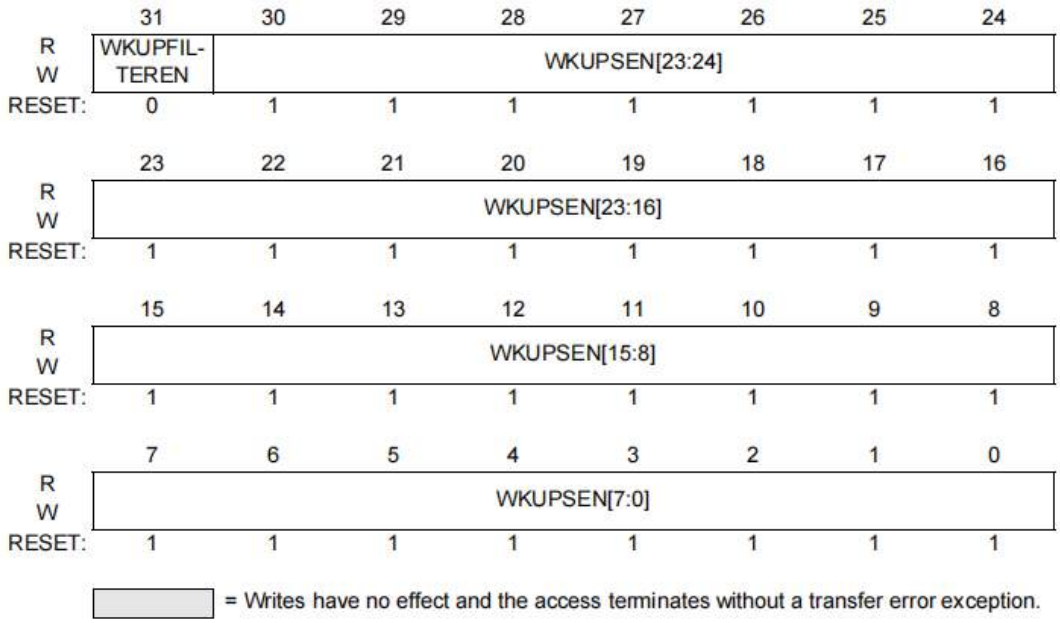


Figure 7-23: WKUPC — Wakeup Configuration Register

WKUPFILTEREN— Wakeup Source Filter Enable

If the WKUPFILTEREN is set, the wakeup source will through a filter to remove glitch and then to wakeup standby mode of the chip.

1 = Wakeup source filter enabled

0 = Wakeup source filter disabled

WKUPSEN— Wakeup Source Enable

This field control whether the corresponding source is used as a source to wakeup standby mode of the chip. If set, the corresponding source is used as wakeup source.

Table 7-10 shows WKUPSEN and the corresponding wakeup source.

Table 7-10: WKUPSEN and the Corresponding Wakeup Source

WKUPSEN	Wakeup Source
WKUPSEN[30]	Reserved
WKUPSEN[29]	Reserved
WKUPSEN[28]	Reserved
WKUPSEN[27]	RTC interrupt
WKUPSEN[26]	PVD interrupt
WKUPSEN[25]	Reserved
WKUPSEN[24]	Reserved
WKUPSEN[23]	COMP0 interrupt
WKUPSEN[22]	Reserved
WKUPSEN[21]	Reserved
WKUPSEN[20]	I2C
WKUPSEN[19]	WDT0 interrupt
WKUPSEN[18]	JTAG POWERON REQEST
WKUPSEN[17]	Reset# pin
WKUPSEN[16]	WDT0 reset
WKUPSEN[15]	Reserved
WKUPSEN[14]	Reserved
WKUPSEN[13]	Reserved
WKUPSEN[12]	Reserved
WKUPSEN[11]	Reserved
WKUPSEN[10]	int1[2]
WKUPSEN[9]	int1[1]
WKUPSEN[8]	int1[0]
WKUPSEN[7]	int0[7]
WKUPSEN[6]	int0[6]
WKUPSEN[5]	int0[5]
WKUPSEN[4]	int0[4]
WKUPSEN[3]	int0[3]
WKUPSEN[2]	int0[2]
WKUPSEN[1]	int0[1]
WKUPSEN[0]	int0[0]

7.5.8. QSPIXIPMCFR — QSPI XIP Mode Configuration Register

The QSPIXIPMCFR register is a read-writable register.

Offset Address: 0x0C04

	31	30	29	28	27	26	25	24
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0	0	0	0	0	0	QSPI0_DATA_ENCR_EN	QSPI0_XIPEN
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 7-24: QSPIXIPMCFR — QSPI XIP Mode Configuration Register

QSPI*_XIPEN— QSPI* XIP Mode Enable Control bit
1 = QSPI* XIP Mode Enabled
0 = QSPI* XIP Mode Disabled

QSPI*_DATA_ENCR_EN— QSPI* XIP Transfer Data Encrypt Function Enable Control bit
1 = QSPI* XIP Transfer Data Encrypt Function Enabled
0 = QSPI* XIP Transfer Data Encrypt Function Disabled

7.5.9. QSPILKEYR — QSPI 32bit Key Register

The QSPIKEYR register is a writable register and read always return 0's.

Offset Address: 0x0C08

	31	30	29	28	27	26	25	24
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[31:24]							
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[23:16]							
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[15:8]							
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[7:0]							
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 7-25: QSPIKEYR — QSPI 32bit Key Register

7.6. Functional Description

The following sections provide an overview of the SIM operation features.

7.6.1. Pad Configuration

The Pad Configuration Registers (PCR) in the SIM allow software control of the static electrical characteristics of external pins. The multiplexed function of a pin, selection of pull up or pull down devices, the slew rate of I/O signals, open drain mode for output pins, hysteresis on input pins, and the drive strength for bus signals can be specified through the PCRs.

7.6.2. GPIO Operation

All GPIO functionality is provided by the SIM for this device. Each device pin that has GPIO functionality has an associated Pin Configuration Register in the SIM where the GPIO function is selected for the pin. In addition, each device pin with GPIO functionality has an input data register (SIM_GPDIx_x) and an output data register (SIM_GPDOx_x).

8. Clock and Power Control Module (CLKM)

8.1. Overview

The clock module contains:

- FIRC: Fast Internal 150MHz RC oscillator
- PLL: Internal Phase Locked Loop Clock
- FXOSC: External Fast Speed crystal oscillator(range from 8MHz to 20MHz)
- SIRC: Internal low speed 128KHz oscillator
- SXOSC: External Low Speed crystal oscillator(32768Hz)
- Status and control registers
- Clock and Power Control logic

8.2. Features

Features of the clock module include:

- Three system clock sources
  - Fast Internal 150MHz RC oscillator
  - Internal PLL clock
  - External Fast Speed crystal oscillator(FXOSC)

- Individual clock divider for IPS, System, TOUCH and ADC clock
- Support for low-power mode
- Modules can be separately stopped by setting MSCR

8.3. Clock Structure

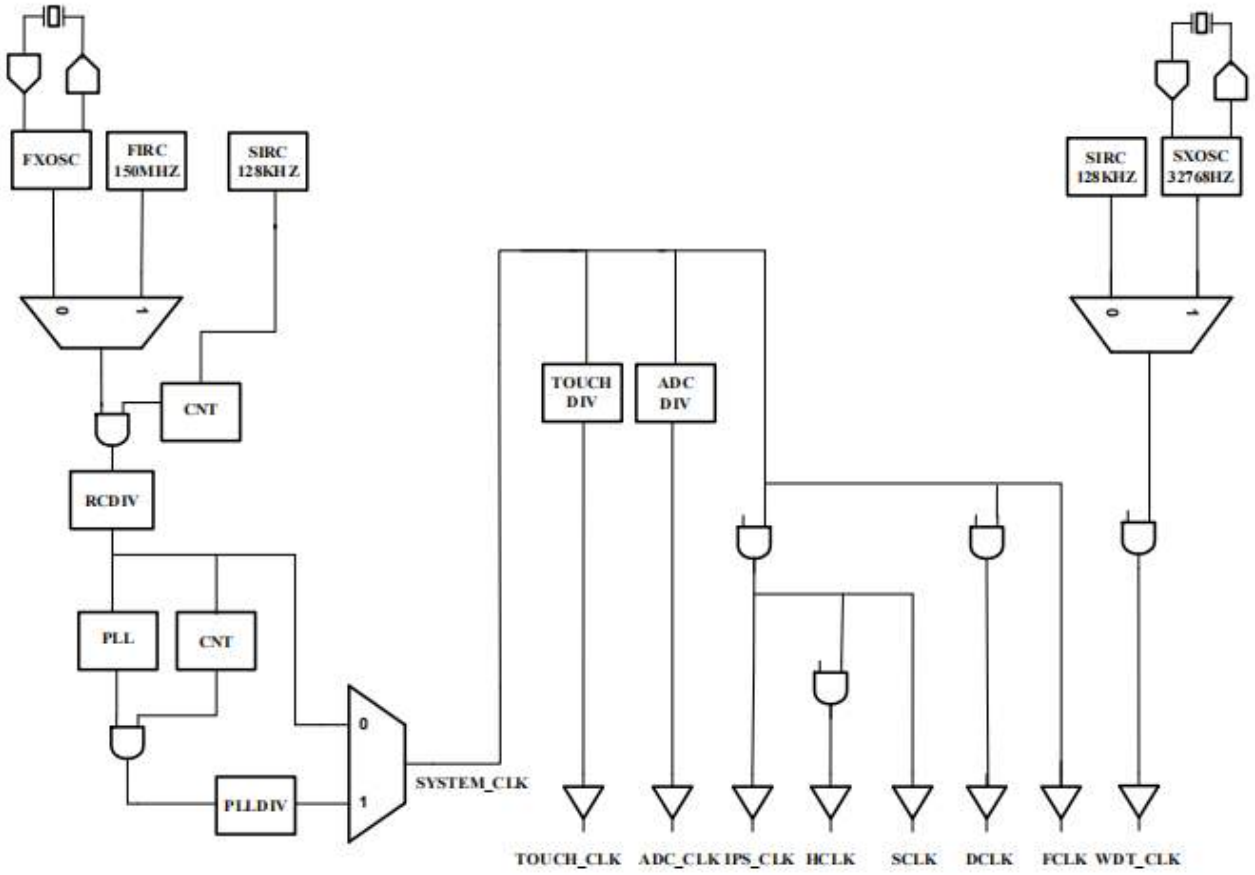


Figure 8-1: Clock Structure

8.4. Clock Source Select

System Clock Source can be Internal PLL clock, Fast Internal 150MHz RC oscillator(FIRC) or external high speed crystal oscillator(FXOSC). Clock source Select is depended on the PLLEN bit of SYNCR register. If it is set then internal PLL is the system clock source, otherwise the system clock source is Fast Internal 150MHz RC oscillator(FIRC) or external high speed crystal oscillator(FXOSC)
which is determined by CCR[CLKMD] bit.

8.4.1. Low-Power Options

8.4.1.1. Wait and Doze Modes

In wait and doze modes, the system clocks to the peripherals and embedded-flash are enabled, the clocks to the CPU, ROM, SRAM are stopped. Each module can disable the module clocks locally at the module level or by setting MSCR.

8.4.1.2. Stop Mode

In stop mode, all system clocks are disabled.

CAUTION: Don't program or erase EFLASH during stop mode.

8.5. Memory Map and Registers

The clock programming model consists of these registers:

- Synthesizer Control Register (SYNCR)
- Low Speed Oscillator Control Register (LOSCCR)
- PLL Configuration and Status Register(PLLCSR)
- Module stop control register (MSCR)
- EPT External and CLKOUT Clock Source Control Register(ECCSCR)
- OSC Bist Test Configuration Register1(OBTCR1)
- OSC Bist Test Configuration Register2(OBTCR2)
- OSC Bist Test Control Register(OBTCR)
- OSC BIST Test Counter Register(OBTCNTR)
- OSC BIST Test Result Register(OBTRR)

8.5.1. Module Memory Map

Table 8-1: Clock Memory Map

Address	31:16	15:0	Access[1]
0x0000	Synthesizer Control Register (SYNCR)		S
0x0004	Low Speed Oscillator Control Register (LOSCCR)		S
0x0008	PLL Configuration and Status Register(PLLCSR)		S
0x000C	Module Stop Control Register (MSCR)		S
0x0010	EPT External and CLKOUT Clock Source Control Register(ECCSCR)		S
0x0014	OSC Bist Test Configuration Register1(OBTCR1)		S
0x0018	OSC Bist Test Configuration Register2(OBTCR2)		S
0x001C	OSC Bist Test Control Register(OBTCR)		S
0x0020	OSC BIST Test Counter Register(OBTCNTR)		S
0x0024	OSC BIST Test Result Register(OBTRR)		S

NOTE: *S = supervisor-only access. User mode accesses to supervisor only address locations have no effect and result in a cycle termination transfer error.*

8.5.2. Register Description

This subsection provides a description of the clock module registers.

8.5.2.1. Synthesizer Control Register

The synthesizer control register (SYNCR) is read/write always.

Offset Address: 0x0000

	31	30	29	28	27	26	25	24
R	SYNCTEST[1:0]		PLLDIV[3:0]				PLLOCKM	ENLOW-POWER
W								
RESET:	0	0	0	0	0	1	0	1

	23	22	21	20	19	18	17	16
R	RCDIV[5:0]						TOUCHDIV[1:0]	
W								
RESET:	0	0	Note1	0	Note1	Note1	1	0

	15	14	13	12	11	10	9	8
R	ADCDIV[3:0]				LOSCEN	PLL-SRCEN	PLLEN	SLEEP
W								
RESET:	0	0	1	0	1	0	0	0

	7	6	5	4	3	2	1	0
R	0	CLKOUT-SEL	STBYMD[1:0]		0	ADCEN	0	LOSCL-PEN
W								
RESET:	0	0	1	1	0	1	0	1

= Writes have no effect and the access terminates without a transfer error exception.

NOTE:*1, Determined by the value of CCR[CLKMD] bit*

Figure 8-2: Synthesizer Control Register (SYNCR)

SYNCTEST[1:0] —SYNCR Write Access Sequence In

The writable bit of SYNCR register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of SYNCR register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

RCDIV[5:0] — FIRC150M or FXOSC Clock Divider

This field sets the divide value for FIRC150M or FXOSC Clock. The default value is 6'b001011 (divide by 12) when *CCR[CLKMD]* bit is set, otherwise is 6'b000000 (divide by 1). The other divide value is referenced to **Table 8-2.**

Table 8-2: FIRC150M or FXOSC Clock Divider

RCDIV[5:0]	Divide Value
000000	Divide-by-1
000001	Divide-by-2
000010	Divide-by-3
000011	Divide-by-4

RCDIV[5:0]	Divide Value
000100	Divide-by-5
000101	Divide-by-6
...	...
...	...
111111	Divide-by-64

NOTE: *RCDIV[5:0] is fixed to 6'h0 (is read only) when CCR[CLKMD] is clear.*

PLLDIV[3:0] — PLL Clock Divider

This field sets the divide value for PLL clock. The default value is 4'b0000 (divide by two). The other divide value is referenced to **Table 8-3**.

Table 8-3: PLL Clock Divider

PLLDIV[3:0]	Divide Value
0000	Divide-by-2
0001	Divide-by-2
0010	Divide-by-4
0011	Divide-by-6
...	...
1101	Divide-by-26
1110	Divide-by-28
1111	Divide-by-30

NOTE: *The frequence of system clock should not be greater than 150MHz.*

TOUCHDIV[1:0] — TOUCH 48MHz Clock Divider

This field sets the divide value for TOUCH 48MHz clock. The default value is 42'b10(divide by 3). The other divide value is referenced to **Table 8-4**

Table 8-4: TOUCH 48MHz Clock Divider

TOUCHDIV[1:0]	Divide Value
00	Divide-by-1
01	Divide-by-2
10	Divide-by-3
11	Divide-by-4

ADCDIV[3:0] — ADC Clock Divider

This field sets the divide value for ADC clock. The default value is 4'b0000(divide by one). The other divide value is referenced to **Table 8-5**.

Table 8-5: ADC Clock Divider

ADCDIV[3:0]	Divide Value
0000	Divide-by-1
0001	Divide-by-2
0010	Divide-by-3
0011	Divide-by-4
0100	Divide-by-5
0101	Divide-by-6
0110	Divide-by-7
0111	Divide-by-8
1000	Divide-by-9
1001	Divide-by-10
1010	Divide-by-11
1011	Divide-by-12
1100	Divide-by-13
1101	Divide-by-14
1110	Divide-by-15
1111	Divide-by-16

LOSCEN — Internal Low Speed 128KHz Oscillator Enable Bit
1 = Internal Low Speed 128KHz Oscillator is enabled
0 = Internal Low Speed 128KHz Oscillator is disabled

PLLSRCEN — System clock will be stopped or not when PLLEN is changed from 0 to 1.
1 = System clock won't be stopped and System clock source is from FXOSC until PLL locked
0 = System clock will be stopped until PLL locked

PLLEN — PLL Enabled control bit.
1 = PLL is enabled and the system clock source is PLL
0 = PLL is disabled and the system clock source is FXOSC SLEEP

— Chip Sleep Mode Control Bit

Set the SLEEP bit, the chip will enter standby mode indicated by STBYMD[1:0]. The operation is the same as "stop" instruction.

NOTE: *SLEEP is valid only when STBYMD[1] = 1'b1. This is not same case as stop instruction because stop instruction is always valid.*

CLKOUTSEL — Clock Out Select Bit

Table 8-6: CLKOUTSEL Mode

CLKOUTSEL	CLKOUT
0	System clock out divider
1	128KHz clock

STBYMD[1:0] — Sleep Operation Control Bits

STBYMD[1:0] control clock source, system clock operation and LDO State in sleep mode as shown in **Table 8-7**.

Table 8-7: Sleep Operation Control Bit in Sleep Mode

	Operation During Sleep Mode				
STBYMD	ADC Clock	TOUCH Clock	System Clocks	Clock Source	LDO
00	Enable	Enable	Disabled	Enable	Normal
01	Disabled	Disabled	Disabled	Enable	Normal
10	Disabled	Disabled	Disabled	Disabled	Normal
11	Disabled	Disabled	Disabled	Disabled	Standby

LOSCLPEN — Internal Low Speed 128KHz Oscillator Low Power Enable

If the LOSCLPE is set, the internal low speed 128KHz oscillator will be stop during the chip enters standby mode.

1 = Low power mode of Internal Low Speed 128KHz Oscillator is enabled
0 = Low power mode of Internal Low Speed 128KHz Oscillator is disabled

ADCEN — Analog-to-digital converter Clock Enable Bit

1 = ADC Clock enable
0 = ADC Clock disable

PLLOCKM — PLL Lock detection flag which is generated by PLL macro.

1 = PLL Lock is generated by PLL macro when PLLEN is set.
0 = PLL Lock is not generated by PLL macro when PLLEN is set.

ENLOWPOWER — Enable Enter Low power mode status Bit

Before system enter standby mode, be sure that this bit is set, otherwise the low power mode won't be entered successfully. This bit is set after recovery from low power mode, and cleared by hardware when enter low power mode

1 = Enable Enter Low power mode status bit
0 = Low power mode is not allowed status bit

8.5.2.2. Low Speed Oscillator Control and Status Register

Offset Address: 0x0004

	31	30	29	28	27	26	25	24
R	LOSCCSTEST[1:0]		0	SIRCST[4:0]				
W								
RESET:	0	0	0	1	1	1	1	1

	23	22	21	20	19	18	17	16
R	SXOSCST[15:8]							
W								
RESET:	0	1	0	1	0	0	0	0

	15	14	13	12	11	10	9	8
R	SXOSCST[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	SIRCRDY	SXOSCRDY	WDTCLKCHGDONE	0	0	0	WDTCLKSEL	SXOSCEN
W								
RESET:	1	0	1	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 8-3: Low Speed Oscillator Control and Status Register (LOSCCSR)

LOSCCSTEST[1:0] —LOSCCSR Write Access Sequence In

The writable bit of IOSCCSR register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of LOSCCSR register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

LOSCST[4:0] — Internal Low Speed Oscillator Stable Time Value

The internal low speed oscillator(SIRC) will wait SIRCST[4:0] cycles of 32KHz(SIRC divided by 4) oscillator and then ready for output clock after switch on.

SXOSCST[15:0] — External Low Speed Oscillator Stable Time Value

The external low speed oscillator(SXOSC) will wait SXOSCST[15:0] cycles of 128KHz(source of SIRC) oscillator and then ready for output clock after switch on.

SXOSCEN — SXOSC is enabled for WDT application.

1 = SXOSC is enabled for WDT

0 = SXOSC crystal is disabled for WDT, then WDTCLKSEL should be clear, otherwise WDT clock will be lost.

WDTCLKSEL — WDT clock selection control bit.

It is recommended that you should clear the bit first then turn off SXOSC when change WDT clock source from SXOSC to SIRC.

1 = WDT clock source is SXOSC(32.768KHz)

0 = WDT clock source is SIRC128KHz

SIRCRDY —Internal low speed oscillator (SIRC) ready flag.
1 = Internal low speed oscillator (SIRC) is ready
0 = Internal low speed oscillator (SIRC) is not ready

SXOSCRDY —External low speed oscillator (SXOSC) is ready for WDT application.
1 = External low speed oscillator (SXOSC) is ready
0 = External low speed oscillator (SXOSC) is not ready

WDTCLKCHGDONE — WDT clock switch done flag. This bit will be change to low when WDTCLKSEL is changed, when WDTCLK change done, then this bit will be set.
1 = WDT clock switch done and WDT can work normally
0 = WDT clock is switching and WDT cannot be work normally

8.5.2.3. PLL Configuration and Status Register

Offset Address: 0x0008

	31	30	29	28	27	26	25	24
R	PLLCSRTEST[1:0]		0	0	0	0	0	PLLOCK
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	PLLN[3:0]			
W								
RESET:	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8
R	PLLOD[1:0]		0	0	0	0	0	0
W								
RESET:	1	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PLLM[6:0]							
W								
RESET:	0	0	1	1	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 8-4: PLL Configuration and Status Register (PLLCSR)

PLLCSRTEST[1:0] —PLLCSR Write Access Sequence In

The writable bit of PLLCSR register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of PLLCSR register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

$$\text{PLL Output Frequency} = XIN * \frac{M}{N} * \frac{1}{NO}$$

CAUTION: *Please keep these conditions during usage:*

1. 3.5MHz <= XIN/N <= 35MHz
2. 200MHz <= XIN*M/N <= 400MHz

3. M≥4

4. N≥1

PLLM[6:0] — Feedback 7bit divider control bits

This field sets the PLL feedback divide value. The default value is 7'h32. The detailed divide value is referenced to **Table 8-8**.

Table 8-8: PLL Feedback Divider Value

PLLM[7:0]	M
7'b000_0000	0
7'b000_0001	1
7'b000_0010	2
7'b000_0011	3
7'b000_0100	4
7'b000_0101	5
...	...
...	...
7'b111_1111	127

PLLOD[1:0] — Output divider control bits

This field sets the output divide value for PLL VCO clock. The default value is 2'b10. The detail divider value is referenced to **Table 8-9**.

Table 8-9: PLL VCO Output Clock Divider

PLLOD[1:0]	NO
00	Divide-by-1
01	Divide-by-2
10	Divide-by-4
11	Divide-by-8

PLLN[3:0] — Input 4bit divider control bits

This field sets the input divide value for PLL clock. The default value is 4'b0010. The detail input divider value is referenced to **Table 8-10**.

Table 8-10: PLL Input Divider

PLLN[3:0]	N
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

PLLN[3:0]	N
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

PLLOCK— PLL Lock flag

This flag will be set after the cycles of PLLOCKCR[PLLST] once PLL is enabled.

1 = PLL is locked

0 = PLL is not locked

8.5.2.4. Module Stop Control Register

The Module Stop Control Register(MSCR) is read/write always.

Offset Address: 0x000C

	31	30	29	28	27	26	25	24
R W	MSCRTEST[1:0]		MS[29]	MS[28]	MS[27]	MS[26]	MS[25]	MS[24]
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	MS[23]	MS[22]	MS[21]	MS[20]	MS[19]	MS[18]	MS[17]	MS[16]
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	MS[15]	MS[14]	MS[13]	MS[12]	MS[11]	MS[10]	MS[9]	MS[8]
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	MS[7]	MS[6]	MS[5]	MS[4]	MS[3]	MS[2]	MS[1]	MS[0]
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 8-5: Module Stop Control Register (MSCR)

MSCRTEST[1:0] —MSCR Write Access Sequence In

The writable bit of MSCR register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of MSCR register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

MS[29:0] — Module Stop Bits

The MS[25:0] bits disable the modules' clocks in the top level. (see **Table 8-11 MS[29:0] Bits Corresponding Modules**).

1 = Module Clock Disabled

0 = Module Clock Enabled

Table 8-11: MS[29:0] Bits Corresponding Modules

MS Bit	Corresponding Module
0	Reserved
1	COMP0
2	Reserved
3	ADC
4	PIT0
5	PIT1
6	Reserved
7	Reserved
8	RTC
9	DMA
10	PWM0
11	Reserved
12	EPORT0
13	EPORT1
14	XBAR
15	OPTION
16	RESET
17	WDT
18	SCI0
19	SIM
20	I2C
21	SCI1
22	SCI2
23	CAN
24	Reserved
25	QSPI0
26	SPI0
27	SPI1
28	Reserved
29	TOUCH

8.5.2.5. EPT External and CLKOUT Clock Source Control Register

Offset Address: 0x0010

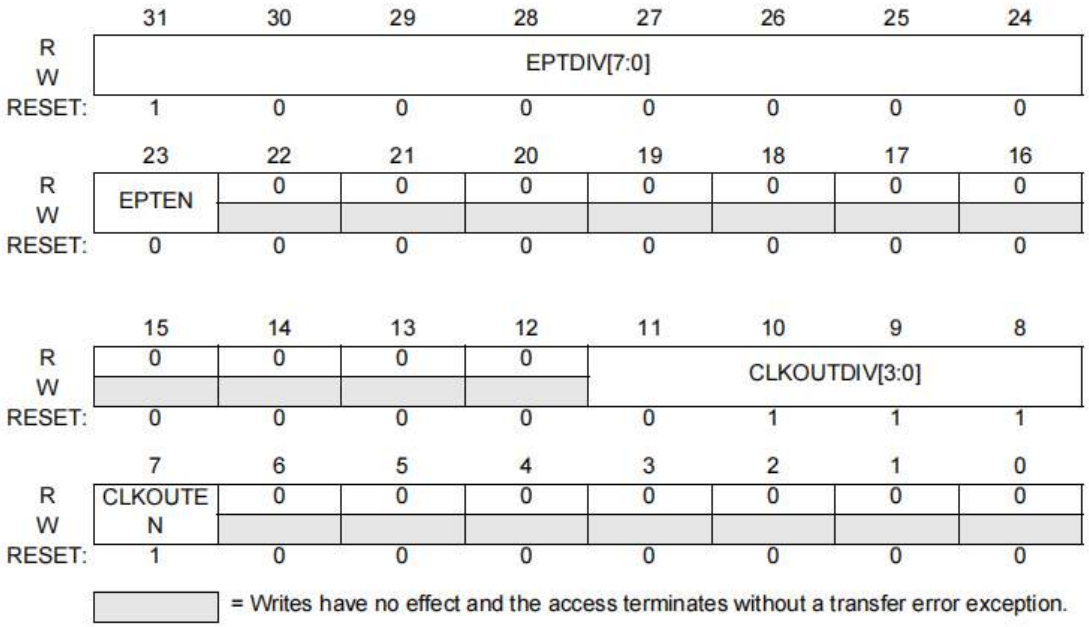


Figure 8-6: EPT External and CLKOUT Clock Source Control Register (ECCSCR)

EPTDIV[5:0] — EPT Clock Divider

This field sets the divide value for EPT clock. The default value is 8'h80. The other divide value is referenced to **Table 8-12**

Table 8-12: EPT Clock Divider

EPTDIV[8:0]	Divide Value
00000000	Divide-by-1
00000001	Divide-by-2
00000010	Divide-by-3
00000011	Divide-by-4
00000100	Divide-by-5
00000101	Divide-by-6
...	...
...	...
...	...
11111110	Divide-by-255
11111111	Divide-by-256

EPTEN — EPT Clock Enable Bit

If the EPTEN bit is set, EPT clock will be divided from system clock.

1 = EPT clock enable

0 = EPT clock disable

CLKOUTDIV[3:0] — CLKOUT Divider

This field sets the system divide value for CLKOUT. The default value is 4'h8. The other divide value is referenced to **Table 8-13**

Table 8-13: CLKOUT Divider

CLKOUTDIV[3:0]	Divide Value
0000	Divide-by-2
0001	Divide-by-4
0010	Divide-by-6
...	...
1111	Divide-by-32

CLKOUTEN — CLKOUT Enable Bit

If the CLKOUTEN bit is set, System clock divider will be enabled for CLKOUT.

1 = System clock divider will be enabled for CLKOUT

0 = System clock divider will be disabled for CLKOUT

8.5.2.6. OSC Bist Test Configuration Register1

Offset Address: 0x0014

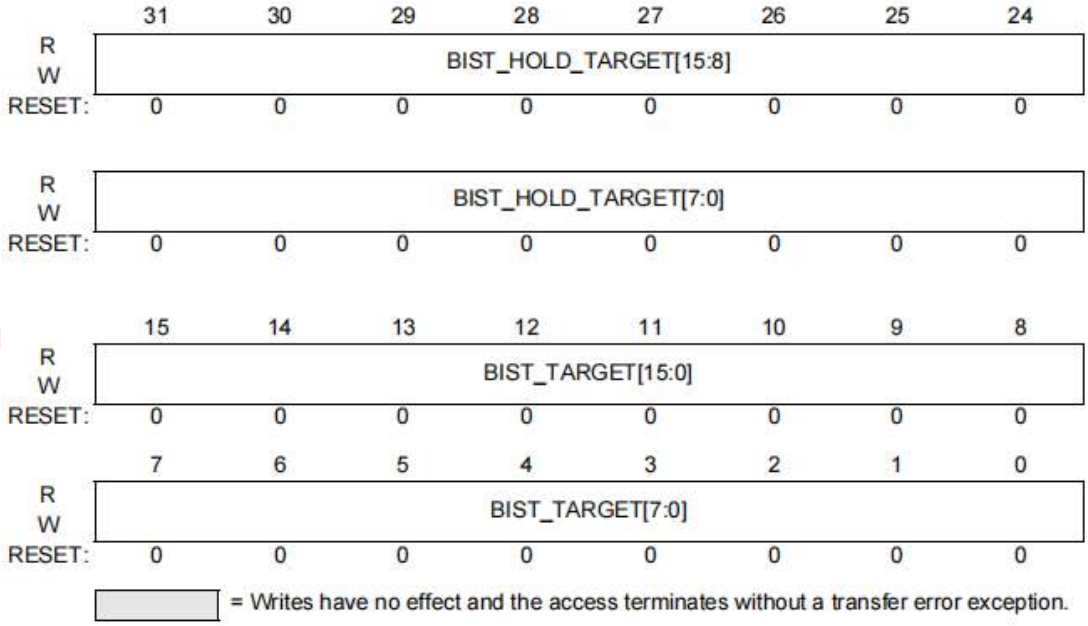


Figure 8-7: OSC Bist Test Configuration Register1(OBTCR1)

BIST_HOLD_TARGET — BIST clk hold count target value

Wait clk under test stable after trim value change

BIST_TARGET — BIST clk count target value

8.5.2.7. OSC Bist Test Configuration Register2

Offset Address: 0x0018

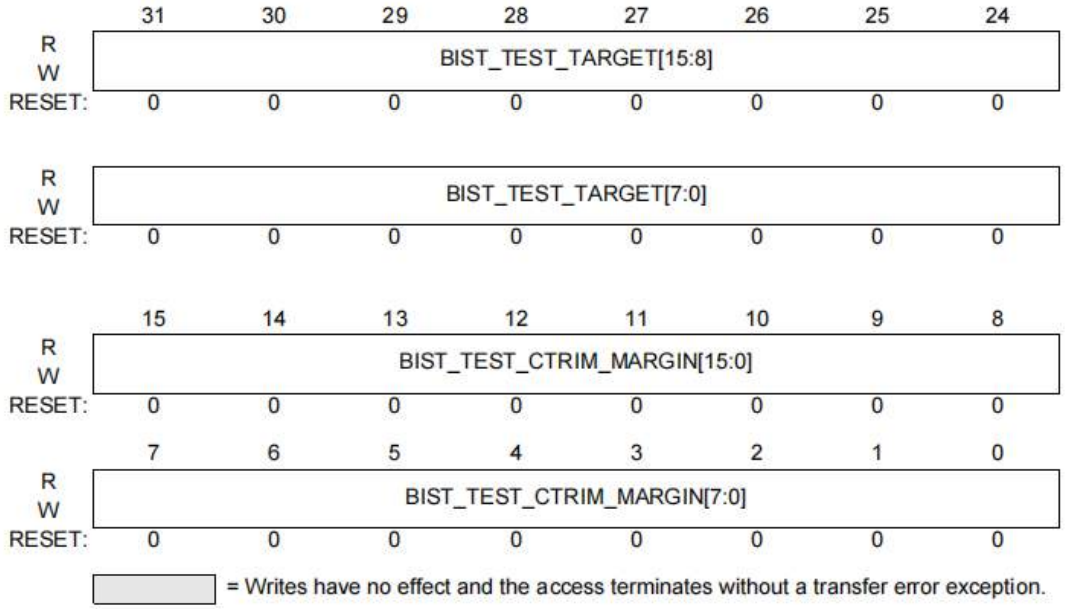


Figure 8-8: OSC Bist Test Configuration Register2(OBTCR2)

BIST_TEST_TARGET — Under test clk count target value

BIST_TEST_CTRIM_MARGIN — Under test clk count error margin for coarse trim

8.5.2.8. OSC Bist Test Control Register

Offset Address: 0x001C

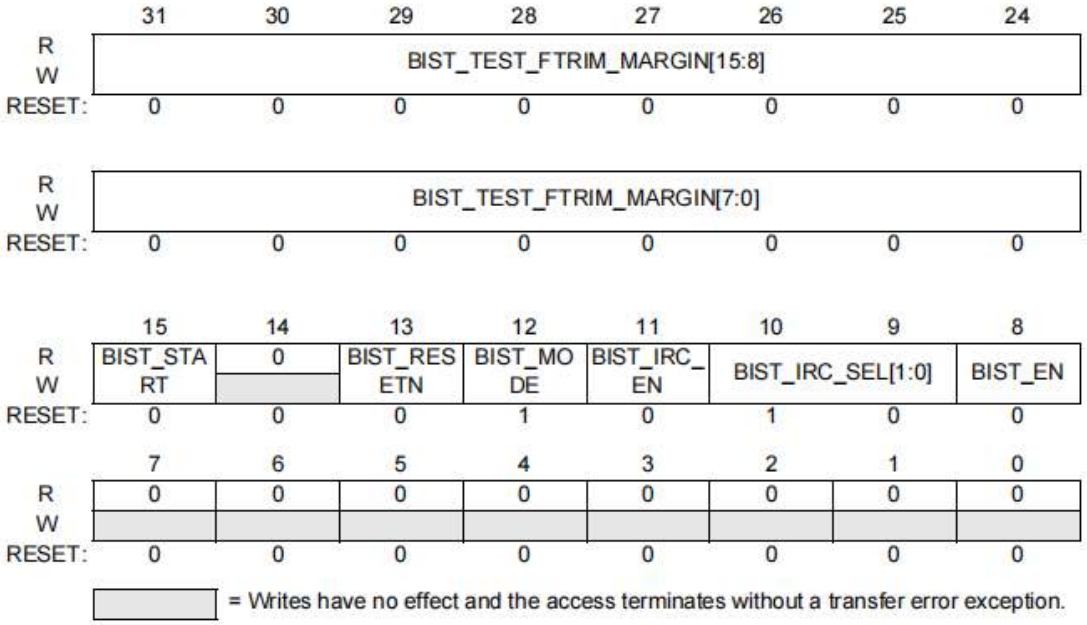


Figure 8-9: OSC Bist Test Control Register(OBTCR)

BIST_TEST_FTRIM_MARGIN — Under test clk count error margin for fine trim

BIST_MODE — BIST Mode

1 = Trace mode (for measure the frequency of PLL,FIRC or 128K clock)
0 = Trim mode

BIST_START — BIST Start

1 = Start
0 = Stop

BIST_RESETN — Bist Reset Negate

1 = Negate Bist reset
0 = Assert Bist Reset

BIST_IRC_EN — FIRC control bit when bist_en is set

1 = FIRC Clock is enabled
0 = FIRC Clock is disabled

BIST_IRC_SEL — FIRC,PLL or 128KHz clock selection

BIST_IRC_SEL[1:0]	Description
2'b00	128KHz
2'b01	128KHz
2'b10	FIRC150MHz
2'b11	PLL

BIST_EN — Reference Clock Enable

1 = Enable
0 = Disable

8.5.2.9. OSC BIST Test Counter Register

Offset Address: 0x0020

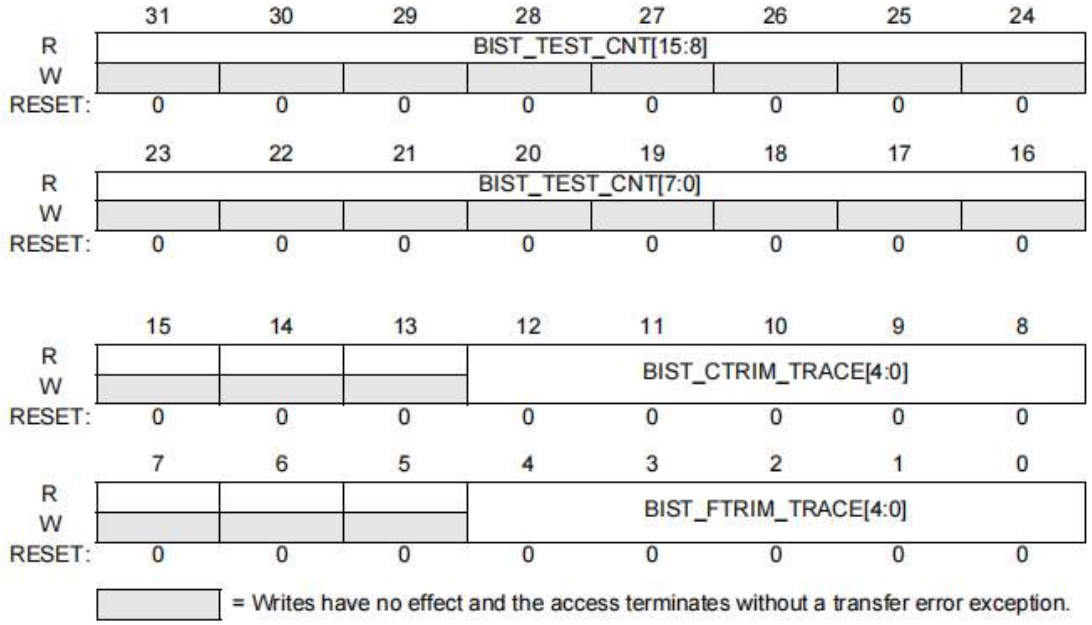


Figure 8-10: OSC BIST Test Counter Register(OBTCNTR)

BIST_CTRIM_TRACE — BIST Ctrim Trace Value

BIST_FTRIM_TRACE — BIST Ftrim Trace Value

BIST_TEST_CNT — BIST Test Counter Value

8.5.2.10. OSC BIST Test Result Register

Offset Address: 0x0024

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	BIST_DONE	BIST_PASS
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	BISTRE-SULT[16]
W								
RESET:	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8
R	BISTRESULT[15:8]							
W								
RESET:	1	0	0	0	0	1	1	1

	7	6	5	4	3	2	1	0
R	BISTRESULT[7:0]							
W								
RESET:	0	1	1	1	1	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 8-11: OSC BIST Test Result Register(OBTRR)

BIST_DONE — BIST Done

1 = Done
0 = Not Done

BIST_PASS — BIST Pass

1 = Pass
0 = Fail

BISTRESULT — BIST trim result valid

WHEN bist_done=1 & bist_pass=1

8.6. Functional Description

8.6.1. Turning On PLL

The following step is recommended:

1. Wait SYNCR[ENLOWPOWER]=1'b1;
2. Configuring SYNCR[PLLEN] =1'b0 and close PLL first.
3. Configuring SYNCR[PLLSRCEN] =1'b0;
4. Configuring PLLCSR[PLLOD]/PLLCSR[PLLN]/PLLCSR[PLLM] according to the target VCO frequency.
5. Configuring SYNCR[PLLEN] and open PLL.
6. Waiting for SYNCR[PLLOCK] status and system clock will be changed to PLL clock.
7. Configuring SYNCR[PLLDIV] according to the target system frequency.

8.6.2. The Frequency of PLL Measurement

The following step is recommended:

1. Setting BIST_RESETN bit with 0 for assert reset.
2. Setting BIST_TARGET counter value.
3. Setting BIST_MODE bit with 1.
4. Setting IRC_BIST_SEL bit with 1 for PLL measurement.
5. Setting BIST _START bit with 1.
6. Setting BIST_RESETN bit with 1 for negated reset.
7. Waiting for BIST_DONE bit.
8. Reading BIST_TEST_CNT for measuring the frequency of PLL clock.
9. Calculating the frequency of PLL clock according to BIST_TEST_CNT and the frequency of fxosc clock.

The Frequency of PLL clock is calculated as follow:

$$f_{PLL} = 2 * f_{fxosc} * BIST\_TEST\_CNT[15:0] / BIST\_TARGET[15:0]$$

8.6.3. The Frequency of 128KHz Measurement

The following step is recommended:

1. Setting BIST_RESETN bit with 0 for assert reset.
2. Setting BIST_TARGET counter value.
3. Setting BIST_MODE bit with 1.
4. Setting IRC_BIST_SEL bit with 0 for 128KHz clock measurement.
5. Setting BIST _START bit with 1.
6. Setting BIST_RESETN bit with 1 for negated reset.
7. Waiting for BIST_DONE bit.
8. Reading BIST_TEST_CNT for measuring the frequency of 128KHz clock.
9. Calculating the frequency of 128KHz clock according to BIST_TEST_CNT and the frequency of fxosc clock.

The Frequency of 128KHz is calculated as follow:

$$f_{128KHz} = f_{fxosc} * BIST\_TEST\_CNT[15:0] / BIST\_TARGET[15:0]$$

9. Reset Controller Module (RCM)

9.1. Overview

The reset controller is provided to determine the cause of reset, assert the appropriate reset signals to the system, and then to keep a history of what caused the reset.

9.2. Features

Module features include:

- Four sources of reset:
  - Power on reset
  - External reset pin
  - Software reset
  - Watchdog Timer Reset
  - Programmable Voltage Detect Reset
- Software-readable status flags indicating the cause of the last reset

9.3. Block Diagram

Figure 9-1 illustrates the reset controller.

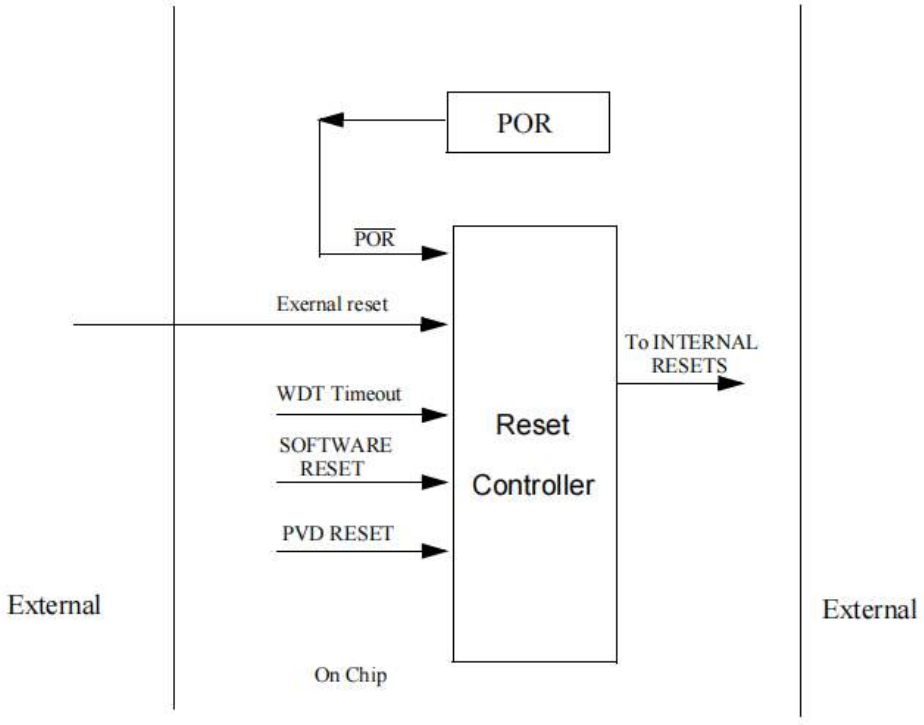


Figure 9-1: Reset Controller Block Diagram

9.4. Memory Map and Registers

The reset controller programming model consists of these registers:

- Reset Control Register (RCR) -- Selects reset controller functions
- Reset Status Register (RSR) -- Reflects the state of the last reset source

See Table 9-1 for the address map and the following paragraphs for a description of the registers.

Table 9-1: Reset Controller Address Map

Address	Bit 7:0	Access [1]
0x0000	Reversed	S/U
0x0001	RTR—Reset Test Register	S/U
0x0002	RSR—Reset Status Register	S/U
0x0003	RCR—Reset Control Register	S/U

NOTE: S/U = supervisor or user mode access.

9.4.1. Reset Test Register

The Reset Test Register(RTR) is only for factory testing.

Offset Address: 0x0001

	7	6	5	4	3	2	1	0
R	TEST[1:0]		0	0	0	0	0	FPOR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and terminate without transfer error exception

Figure 9-2: Reset Test Register(RTR)

TEST[1:0] —RTR Write Access Sequence In

The writable bit of FPOR register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of FPOR bit can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

FPOR — Force Power On Reset

Write 1 to the FPOR bit will result in system power on reset. The reset will result in the chip trimming again.

9.4.2. Reset Status Register

The Reset Status Register(RSR) contains a status bit for every reset source. When reset is entered, the cause of the reset condition is latched along with a value of 0 for the other reset sources that were not pending at the time of the reset condition. These values are then reflected in RSR. One or more status bits may be set at the same time. The cause of any subsequent reset is also recorded in the register, overwriting status from the previous reset condition.

RSR can be read at any time. Writing to RSR has no effect.

Offset Address: 0x0002

	7	6	5	4	3	2	1	0
R	ERST	PVD	SOFT	0	POR	0	0	0
W								
RESET:	0	Reset Dependant						

= Writes have no effect and terminate without transfer error exception

Figure 9-3: Reset Status Register(RSR)

ERST — External Reset

This bit indicates that the last reset state was caused by an external reset.
1 = Last reset state was caused by an external reset
0 = Last reset state was not caused by an external reset

PVD — Programmable Voltage Detect

This bit indicates that the last reset state was caused by an PVD reset.
1 = Last reset state was caused by an PVD reset
0 = Last reset state was not caused by an PVD reset

SOFT — Software Reset Flag

This bit indicates that the last reset state was caused by software.
1 = Last reset state was caused by software.
0 = Last reset state was not caused by software.

POR — Power-On Reset Flag

This bit indicates that the last reset state was caused by power-on or WDT reset
1 = Last reset state was caused by power-on reset.
0 = Last reset state was not caused by power-on reset.

9.4.3. Reset Control Register

The Reset Control Register (RCR) allows software control for requesting a reset.

Offset Address: 0x0003

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	FRCR-STOUT
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and terminate without transfer error exception

Figure 9-4: Reset Control Register(RCR)

FRCRSTOUT — Force RSTOUT Pin

The FRCRSTOUT bit allows software to assert or negate the external RSTOUT pin.

1 = Assert RSTOUT pin

0 = Negate RSTOUT pin

9.5. Functional Description

9.5.1. Reset Sources

Table 9-2 defines the sources of reset and the signals driven by the reset controller.

Table 9-2: Reset Source Summary

Source	Type
POR	Asynchronous
ERST	Asynchronous
Watchdog timer	Asynchronous
Software	Synchronous
PVD	Asynchronous

To protect data integrity, a synchronous reset source is not acted upon by the reset control logic until the end of the current bus cycle. Reset is then asserted on the next rising edge of the system clock after the cycle is terminated. Whenever the reset control logic must synchronize reset to the end of the bus cycle, the internal bus monitor is automatically enabled.

Asynchronous reset sources usually indicate a catastrophic failure. Therefore, the reset control logic does not wait for the current bus cycle to complete. Reset is asserted immediately to the system.

9.5.1.1. Power-On Reset (POR)

At power up, the reset controller asserts system reset. system reset continues to be asserted until POR has reached a minimum acceptable level.

9.5.1.2. Watchdog Timer Reset

A watchdog timer timeout causes timer reset request to be recognized and latched.

9.5.1.3. Software Reset

If the SYSRESTEQ bit in RISC Core Embedded Interrupt Controller (EIC) is set, the software reset will be generated. The reset controller asserts system reset for approximately 2048 cycles. Then the part exits reset and resumes operation.

9.5.1.4. Programmable Voltage Detect Reset

When the PVDRE bit of the CCR register in Embedded Flash Module (EFM) is set, PVD will generate reset when the VDD exceeds the PVD threshold.

9.5.2. Reset Control Flow

The reset logic control flow is shown in Figure 9-5. All cycle counts given are approximate.

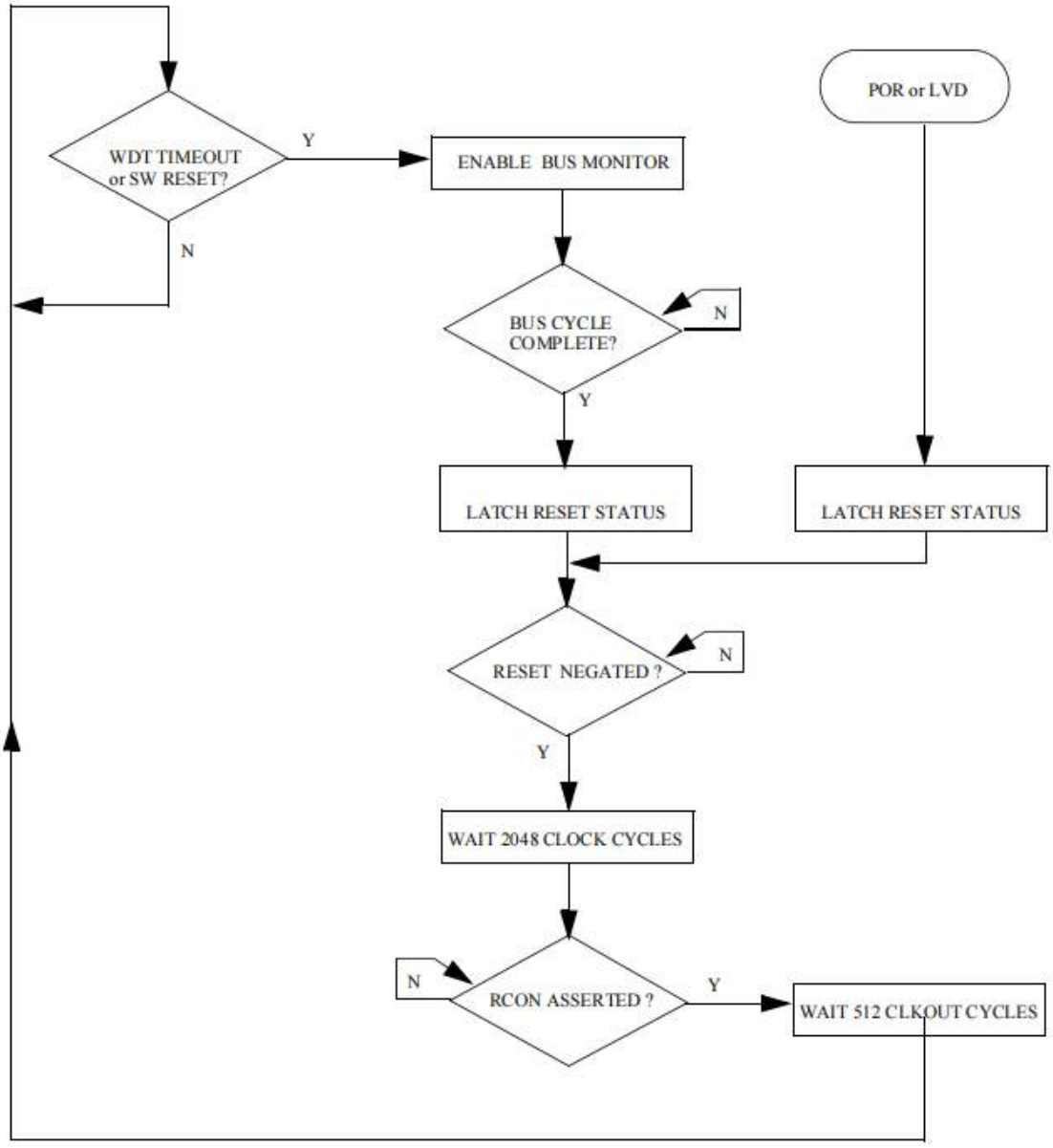


Figure 9-5: Reset Control Flow

10. Static Random Access Memory (SRAM)

10.1. Introduction

Features of the static random access memory (SRAM) include:

- On-chip 32-Kbyte SRAM
- Fixed address space
- Byte, half-word (16bit), or word (32bit) read/write accesses
- One clock per access (including bytes, half-words, and words)
- Supervisor or user mode access

10.2. Modes of Operation

Access to the SRAM is not restricted in any way. The array can be accessed in supervisor and user modes.

10.3. Low-Power Modes

In wait, doze and stop mode, clocks to the SRAM are disabled. No recovery time is required when exiting these modes.

10.4. Reset Operation

The SRAM contents are undefined immediately following a power-on reset. SRAM contents are unaffected by system reset. If a synchronous reset occurs during a read or write access, then the access completes normally and any pipelined access in progress is stopped without corruption of the SRAM contents.

10.5. Interrupts

The SRAM module does not generate interrupt requests.

11. Cache Module (CACHEM)

11.1. Introduction

The Cache module provides the processor with tightly-coupled processor-local memories and bus paths to EBI and QSPI0 memory spaces.

A cache is a block of high-speed memory locations containing address information (commonly known as a tag) and the associated data. The purpose is to decrease the average time of a memory access. Caches operate on two principles of locality:

- Spatial locality — An access to one location is likely to be followed by accesses from adjacent locations (for example, sequential instruction execution or usage of a data structure).
- Temporal locality — An access to an area of memory is likely to be repeated within a short time period (for example, execution of a code loop).

To minimize the quantity of control information stored, the spatial locality property is used to group several locations together under the same tag. This logical block is commonly known as a cache line.

When data is loaded into a cache, access times for subsequent loads and stores are reduced, resulting in overall performance benefits. An access to information already in a cache is known as a cache hit, and other accesses are called cache misses.

Normally, caches are self-managing, with the updates occurring automatically. Whenever the processor wants to access a cacheable location, the cache hit is checked. If the access is a cache hit, the access occurs immediately. Otherwise, a location is allocated and the cache line is loaded from memory. Different cache topologies and access policies are possible.

However, they must comply with the memory coherency model of the underlying architecture. Caches introduce a number of potential problems, mainly because of:

- Memory accesses occurring at times other than when the programmer would normally expect them;
- The existence of multiple physical locations where a data item can be held.

The Cache Controller is targeted for use with any 32bit AHB-bus based application that desires a cache function. This cache function can increase performance by providing fast access to recently used code or data.

The local memory controller supports three modes of operation:

1. Write-through — access to address spaces with this cache mode are cacheable.
   - A write-through read miss on the input bus causes a line read on the output bus of a 16-byte-aligned memory address containing the desired address. This miss data is loaded into the cache and is marked as valid and not modified.
   - A write-through read hit to a valid cache location returns data from the cache with no output bus access.
   - A write-through write miss bypasses the cache and writes to the output bus (no allocate on write miss policy for write-through mode spaces).
   - A write-through write hit updates the cache hit data and writes to the output bus.
2. Write-back — access to address spaces with this cache mode are cacheable.
   - A write-back read miss on the input bus will cause a line read on the output bus of a 16-byte-aligned memory address containing the desired address. This miss data is loaded into the cache

and marked as valid and not modified.

- A write-back read hit to a valid cache location will return data from the cache with no output bus access.
- A write-back write miss will do a "read-to-write" (allocate on write miss policy for write-back mode spaces). A line read on the output bus of a 16 byte aligned memory address containing the desired write address is performed. This miss data is loaded into the cache and marked as valid and modified; and the write data will then update the appropriate cache data locations.

3. Non-cacheable — access to address spaces with this cache mode are not cacheable.
   - These accesses bypass the cache and access the output bus.

The Cache Controller has two AHB-bus interfaces, a master and a slave interface. The master interface has decoded logic to determine the cache mode of valid address phase accesses. Master accesses will then access or bypass the cache depending on their cache mode. The slave interface is used for cache misses as well as accesses that bypass the cache.

The Cache Controller has a 2-way set-associative organization. The cache has
32bit wide address and data paths and a 16-byte line size. The cache tags and data storage use single port, synchronous RAMs. The Controller has a variety of read and write data buffers to improve performance. Cache misses and line pushes generate 4-beat 32bit wrapping burst accesses with the critical word accessed first for maximum performance.

11.2. Block Diagram

This Cache module provides zero wait state access to RAM and cacheable address spaces.

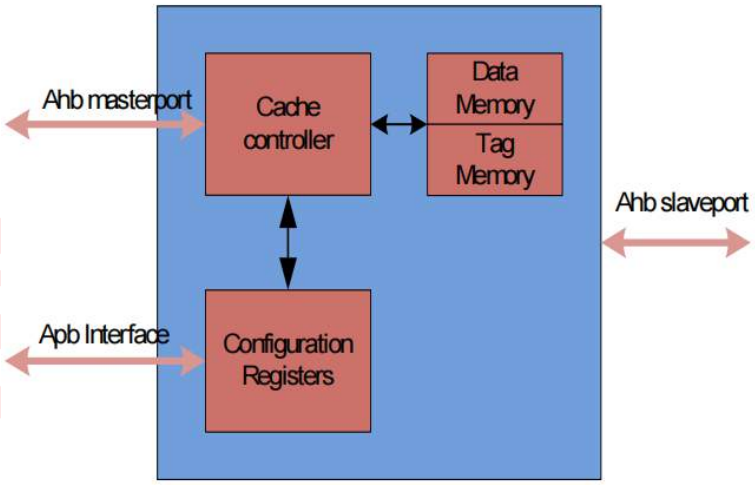


Figure 11-1: Cache Module Block Diagram

11.3. Memory Map/Register Definition

Table 11-1: Cache Module Memory Map

Address Offset	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0	Access
0x0000	Cache Control Register (LMEM_CCR)				S/U
0x0004	Cache Line Control Register (LMEM_CLCR)				S/U
0x0008	Cache Search Address Register (LMEM_CSAR)				S/U
0x000C	Cache Read/Write Value Register (LMEM_CCVR)				S/U
0x0020	Cache Access Register(LMEM_ACR)				S/U
0x0180	Cache Page Invalidation Start Address				S/U
0x0184	Cache Page Invalidate Size				S/U
0x0188	Cache Clock Gate				S/U

NOTE:

1. S = CPU supervisor mode access only, U = CPU user mode access only

2. User mode accessed to supervisor-only address locations have no effect and result in a cycle termination transfer error.

11.4. Register Description

This subsection provides a description of the Cache module.

11.4.1. Cache Control Register (LMEM_CCR)

Offset Address: 0x0000

	31	30	29	28	27	26	25	24
R	GO	0	0	0	PUSHW1	INVW1	PUSHW0	INVW0
W								
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	ENCACHE
W								
RESET:	0	1	1	1	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 11-2: Cache Control Register (LMEM_CCR)

ENCACHE—Cache enable.
1 = cache enabled
0 = cache disabled

INVW0— Invalidate way 0. If the PUSHW0 and INVW0 bits are set, then after setting the GO bit, push all modified lines in way 0 and invalidate all lines in way 0 (clear way 0).
1 = When setting the GO bit, invalidate all lines in way 0.
0 = no operation

PUSHW0 — Push way 0
1 = When setting the GO bit, push all modified lines in way 0
0 = no operation

INVW1— Invalidate way 1. If the PUSHW0 and INVW0 bits are set, then after setting the GO bit, push all modified lines in way 1 and invalidate all lines in way 1 (clear way 1).
1 = When setting the GO bit, invalidate all lines in way 1.
0 = no operation

PUSHW1 — Push way 1
1 = When setting the GO bit, push all modified lines in way 1
0 = no operation

GO— Initiate Cache Command Setting this bit initiates the cache command indicated by bits 27-24. Reading this bit indicates if a command is active. This bit stays set until the command completes. Writing zero has no effect.
1 = Write: initiate command indicated by bits 27-24. Read: cache command active.
0 = Write: no effect. Read: no cache command active.

11.4.2. Cache Line Control Register (LMEM_CLCR)

Offset Address: 0x0004

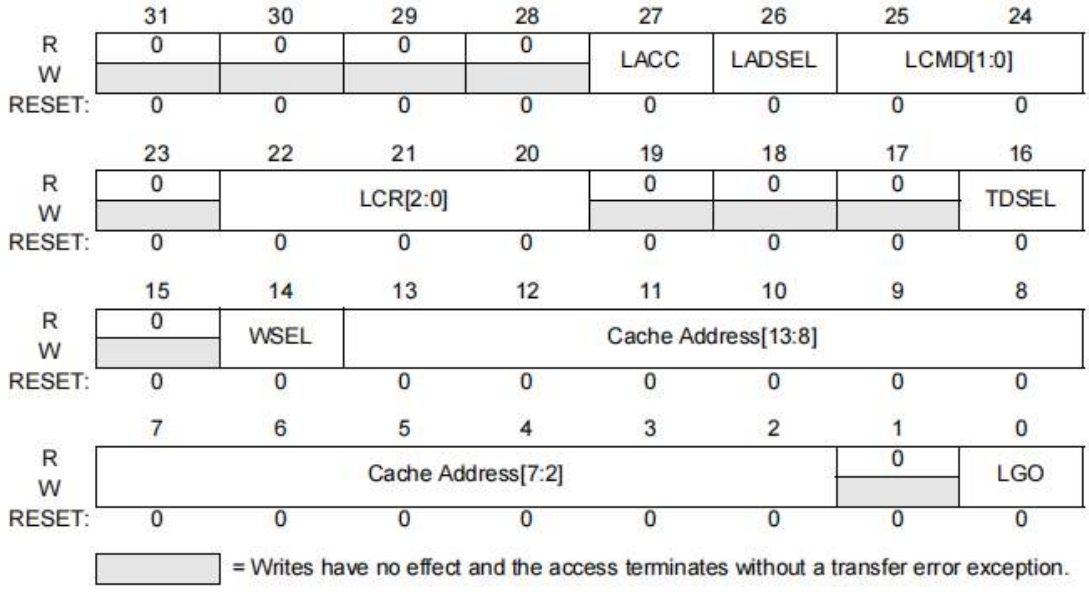


Figure 11-3: Cache Line Control Register (LMEM_CLCR)

LGO— Initiate Cache Line Command. Setting this bit initiates the cache line command indicated by **LMEM_CLCR[**27-24]. Reading this bit indicates if a line command is active. This bit stays set until the command completes. Writing zero has no effect. This bit is shared with CSAR[LGO].

1 = Write: initiate line command indicated by bits 27-24. Read: line command active.
0 = Write: no effect. Read: no line command active.

CACHE_ADDRESS— Cache address.

CLCR[13:4] bits are used to access the tag arrays; CLCR[13:2] bits are used to access the data arrays.

WSEL— Way select. Select the way of the line commands.

1 = way 1.
0 = way 0.

TDSEL— Tag/Data select. Select tag or data for search and read or write commands.

1 = Tag.
0 = Data.

LCR[2:0] — Line command current line status.

LCMD[1:0] — Line command.

00 = Search and read or write.
01 = Invalidate
10 = Push
11 = Clear

LADSEL — Line Address Select. When using the cache address, the way must also be specified in CLCR[WSEL]. When using the physical address, both ways are searched and the command is performed only if a hit.

0 = Cache address
1 = Physical address

LACC — Line access type.

0 = Read
1 = Write

11.4.3. Cache Search Address Register (LMEM_CSAR)

The CSAR register is used to define the explicit cache address or the physical address for line-sized commands specified in the CLCR[LADSEL] bit.

Offset Address: 0x0008

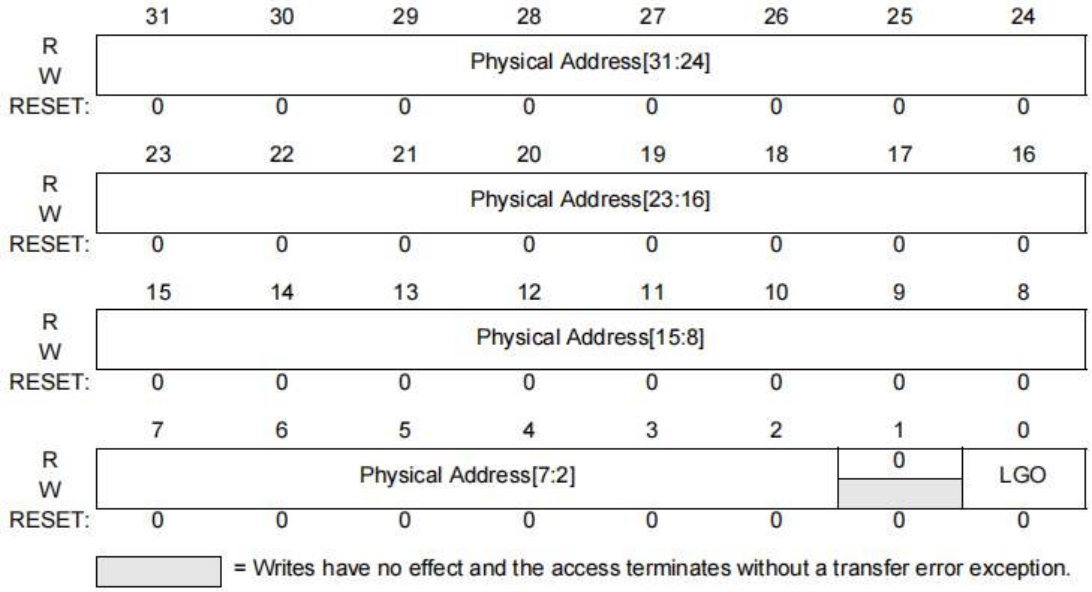


Figure 11-4: Cache Search Address Register (LMEM_CSAR)

Physical_Address[31:2] — Physical Address. This represents bits [31:2] of the system address. Physical_Address[31:14] bits are used for tag compare Physical_Address[13:4] bits are used to access the tag arrays Physical_Address[13:2] bits are used to access the data arrays.

LGO— Initiate Cache Line Command. Setting this bit initiates the cache line command indicated by **LMEM_CLCR[**27-24]. Reading this bit indicates if a line command is active. This bit stays set until the command completes. Writing zero has no effect.

This bit is shared with CLCR[LGO]

- Write: initiate line command indicated by **LMEM_CLCR[**27-24]. Read: line command active.
- Write: no effect. Read: no line command active.

11.4.4. Cache Read/Write Value Register (LMEM_CCVR)

The CCVR register is used to store the write data or read data for the commands specified in the CLCR register.

Offset Address: 0x000C

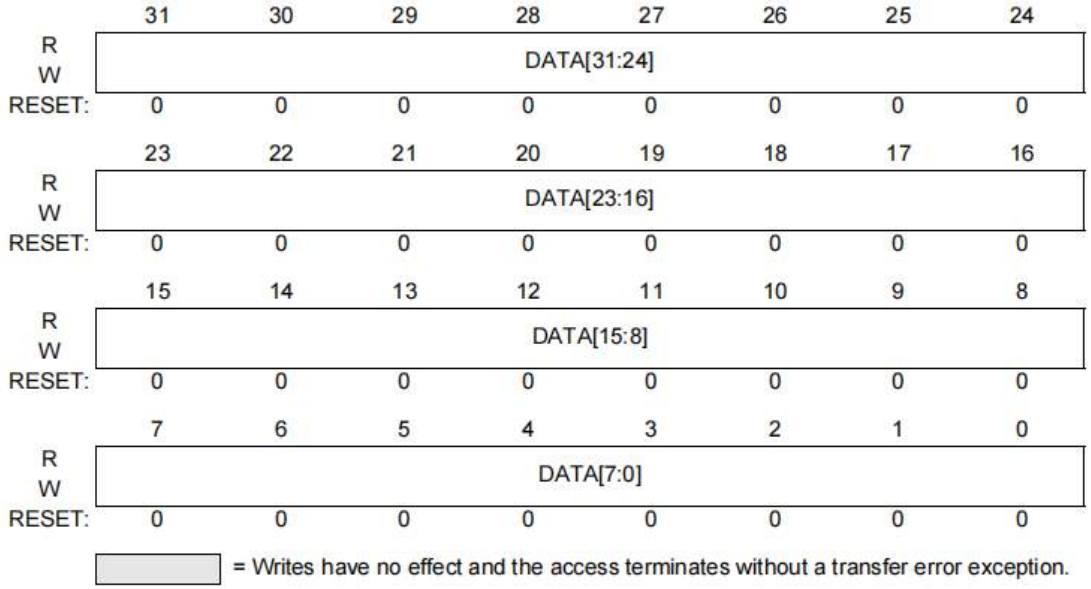


Figure 11-5: Cache Search Address Register (LMEM_CSAR)

DATA— Cache read/write Data

For tag search, read or write:

- DATA[31:14] bits are used for tag array R/W value
- DATA[13:4] bits are used for tag set address on reads; unused on writes
- DATA[3:2] bits are reserved

For data search, read or write:

- DATA[31:0] bits are used for data array R/W value

11.4.5. Cache Access Register(LMEM_ACRG)

The ACRG is used to control the attributes of 10 different memory regions such as write-through, write-back or non-cacheable.

Offset Address: 0x0020

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	R3_CACHEABLE	R3_WT_WB	R2_CACHEABLE	R2_WT_WB	R1_CACHEABLE	R1_WT_WB	R0_CACHEABLE	R0_WT_WB
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 11-6: Cache Access Register(LMEM_ACRG)

Region0: 0x2000_0000 ~ 0x2FFF_FFFF

Region1: 0x6000_0000 ~ 0x6FFF_FFFF

Region2: 0x7000_0000 ~ 0x7FFF_FFFF

Region3: 0x8000_0000 ~ 0x8FFF_FFFF

Rx_CACHEABLE— Region x is cacheable.
1 = cacheable.
0 = non-cacheable

Rx_WT_WB— Region x is write-through if cacheable.
1 = write-back
0 = write-through

11.4.6. Cache Page Invalidation Base Address Register (LMEM_PAGE_INV_BADDR)

Offset Address: 0x0180

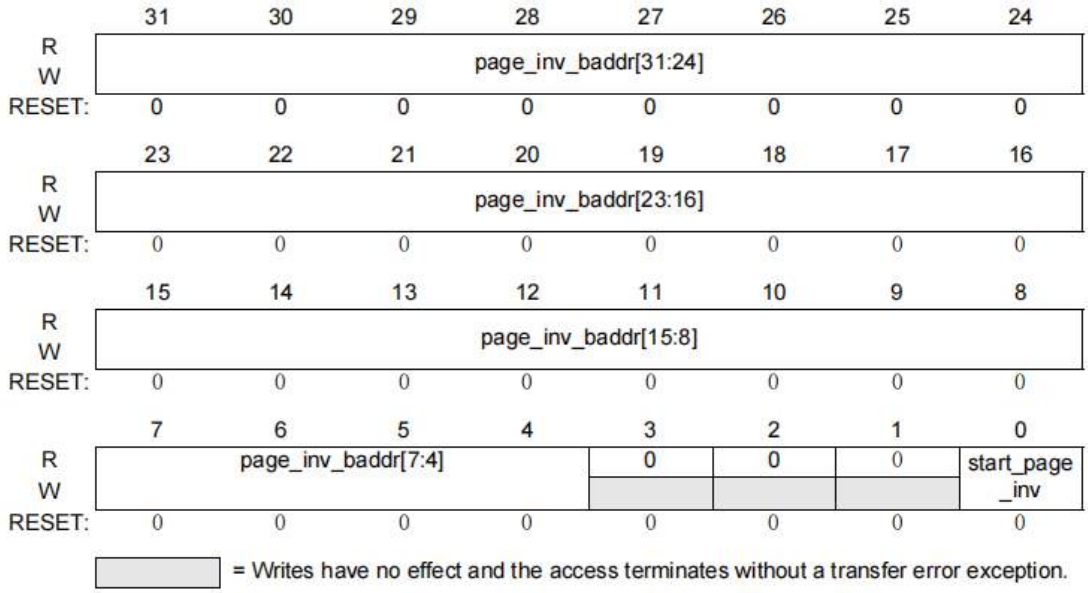


Figure 11-7: Cache Page Invalidate Base Address Register (LMEM_PAGE_INV_BADDR)

page_inv_baddr[31:4] — cache invalidate start address for system memory. Lower 4bits is ignored because of line alignment.

start_page_inv — start to page invalidate. Cleared if the invalidation is completed.
This bit is shared in bit 0 of LMEM_PAGE_INV_SIZE_REG.

11.4.7. Cache Page Invalidation Base Size Register (LMEM_PAGE_INV_SIZE)

Offset Address: 0x0184

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	page_inv_size[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	page_inv_size[7:4]				0	0	0	start_page_inv
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 11-8: Cache Page Invalidate Size Register (LMEM_PAGE_INV_SIZE)

page_inv_size[15:4] — cache invalidate size for system memory. Lower 4bits is ignored because of line alignment.

start_page_inv — start to page invalidate. Cleared if the invalidation is completed. This bit is shared in bit 0 of LMEM_PAGE_INV_BADDR_REG.

11.4.8. Cache Clock Enable Register (LMEM_CACHE_CLK_EN)

Offset Address: 0x0188

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	cache_clk_ enable
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 11-9: Cache Clock Enable Register (LMEM_CACHE_CLK_EN)

cache_clk_enable — cache clock is enabled. Must be '1' if cache is enabled.

11.5. Cache Function

The caches on this device are structured as follows. Both caches have a 2-way set-associative cache structure with a total size of 32 KBytes. The caches have 32bit address and data paths and a 16-byte line size. The cache tags and data storage use external single-port, synchronous RAMs.

For these 32-KByte caches, each cache TAG function uses two 1024 x 20bit RAM arrays and the cache DATA function uses two 4096 x 32bit RAM arrays. The cache TAG entries store 18bits of upper address as well as a modified and valid bit per cache line. The cache DATA entries store four bytes of code or data.

All normal cache accesses use physical addresses. This leads to the following cache address use:

CACHE - 32 KByte size = (1024 sets) x (16-byte lines) x (2-way set-associative)

TAG:
- address[31:14] used in tag for compare (hit) logic
- address[13:4] used to select 1 of 1024 sets
- address[3:0] not used

DATA
- address[31:14] not used
- address[13:4] used to select one of 1024 sets
- address[3:2] used to select one of four 32bit words within a set
- address[1:0] used to select the byte within the 32bit word

The caches use a one-cycle parallel TAG and DATA access structure. This structure is aligned in the two stage AHB bus pipeline from the negative edge of the address phase to the negative edge of the data phase:

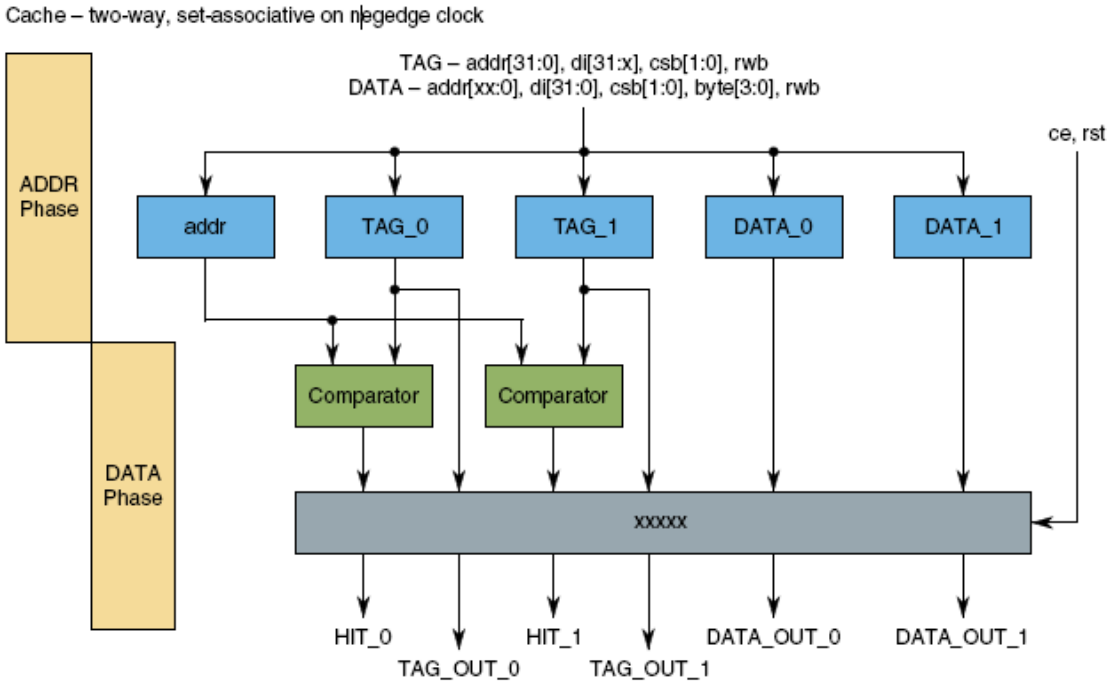


Figure 11-10: Cache Tag and Data Access Structure

On a normal cacheable operation, the full address is registered on the negedge of the address phase in the cache control logic while address bits[12:4] are registered in the TAG arrays and address bits[12:2] are registered in the DATA arrays. During the full cycle from the negedge of the address phase to the negative edge of the data phase, the TAG and DATA arrays are accessed and cache hit analyzed. For hits, the desired read data is returned in the second half of the data phase and writes are done starting on the negedge of the data phase cycle. For misses, the data phase is held as needed while the cache uses its slave bus (CCM, CSM) to access the desired cache line via a line-sized data transfer sent to the crossbar switch.

11.6. Cache Control

The Code and System Caches are disabled at reset. Cache tag and data arrays are not cleared at reset. Therefore, to enable the caches, cache commands must be done to clear and initialize the required tag array bits and to configure and enable the caches.

11.6.1. Cache Set Commands

The cache set commands may operate on:

- all of way 0,
- all of way 1, or
- all of both ways (complete cache).

Cache set commands are initiated using the upper bits in the CCR register. Cache set commands perform their operation on the cache independent of the cache enable bit, CCR[ENCACHE].

A cache set command is initiated by setting the CCR[GO] bit. This bit also acts as a busy bit for set commands. It stays set while the command is active and is cleared by the hardware when the set command completes.

Supported cache set commands are given in Figure 11-11. Set commands work as follows:

- Invalidate − Unconditionally clear valid and modify bits of a cache entry.
- Push − Push a cache entry if it is valid and modified, then clear the modify bit. If entry not valid or not modified, leave as is.
- Clear − Push a cache entry if it is valid and modified, then clear the valid and modify bits. If entry not valid or not modified, clear the valid bit.

CCR[27:24]				Command
PUSH W1	INVW1	PUSH W0	INVW0	
0	0	0	0	NOP
0	0	0	1	Invalidate all way 0
0	0	1	0	Push all way 0
0	0	1	1	Clear all way 0
0	1	0	0	Invalidate all way 1
0	1	0	1	Invalidate all way 1; invalidate all way 0 (invalidate cache)
0	1	1	0	Invalidate all way 1; push all way 0
0	1	1	1	Invalidate all way 1; clear all way 0
1	0	0	0	Push all way 1
1	0	0	1	Push all way 1; invalidate all way 0
1	0	1	0	Push all way 1; push all way 0 (push cache)
1	0	1	1	Push all way 1; clear all way 0
1	1	0	0	Clear all way 1
1	1	0	1	Clear all way 1; invalidate all way 0
1	1	1	0	Clear all way 1; push all way 0
1	1	1	1	Clear all way 1; clear all way 0 (clear cache)

Figure 11-11: Cache Set Commands

After a reset, complete an invalidate cache command before using the cache.

11.6.2. Cache Line Commands

Cache line commands operate on a single line in the cache at a time. Cache line commands can be performed using a physical or cache address.

- A cache address consists of a set address and a way select. The line command acts on the specified cache line.
- Cache line commands with physical addresses first search both ways of the cache set specified by bits [13:4] of the physical address. If they hit, the commands perform their action on the hit way.

Cache line commands are specified using the upper bits in the CLCR register. Cache line commands perform their operation on the cache independent of the cache enable bit (CCR[ENCACHE]). Using a cache address, the command can be completely specified using the CLCR register. Using a physical address, the command must also use the CSAR register to specify the physical address.

A line cache command is initiated by setting the line command go bit (CLCR[LGO] or CSAR[LGO]). This bit also acts a busy bit for line commands. It stays set while the command is active and is cleared by the hardware when the command completes.

The CLCR[27:24] bits select the line command as follows:

CLCR[27:24]			Command
LACC	LADSEL	LCMD	
0	0	00	Search by cache address and way
0	0	01	Invalidate by cache address and way
0	0	10	Push by cache address and way
0	0	11	Clear by cache address and way
0	1	00	Search by physical address
0	1	01	Invalidate by physical address
0	1	10	Push by physical address
0	1	11	Clear by physical address
1	0	00	Write by cache address and way
1	0	01	Reserved, NOP
1	0	10	Reserved, NOP
1	0	11	Reserved, NOP
1	1	*xx*	Reserved, NOP

Figure 11-12: Cache Line Commands

11.6.2.1. Executing a Series of Line Commands Using Cache Addresses

A series of line commands with incremental cache addresses can be performed by just writing to the CLCR.

- Place the command in CLCR[27:24],
- Set the way (CLCR[WSEL]) and tag/data (CLCR[TDSEL]) controls as needed,
- Place the cache address in CLCR[CACHEADDR], and
- Set the line command go bit (CLCR[LGO]).

When one line command completes, initiate the next command by following these steps:

- Increment the cache address (at bit 2 to step through data or at bit 4 to step through lines), and
- Set the line command go bit (CLCR[LGO]).

11.6.2.2. Executing a Series of Line Commands Using Physical Addresses

Perform a series of line commands with incremental physical addresses using the following steps:

- Place the command in CLCR[27:24]
- Set the tag/data (CLCR[TDSEL]) control
- Place the physical address in CSAR[PHYADDR] and set the line command go bit(CSAR[LGO]).

When one line command completes, initiate the next command by following these steps:

- Increment the physical address (at bit 2 to step through data or at bit 4 to step through lines), and
- Set the line command go bit (CSAR[LGO]).

The line command go bit is shared between the CLCR and CSAR registers, so that the above steps can be completed in a single write to the CSAR register.

11.6.2.3. Line Command Results

At completion of a line command, the CLCR register contains information on the initial state of the line targeted by the command. For line commands with cache addresses, this information is read before the line command action is performed from the targeted cache line. For line commands with physical addresses, this information is read on a hit before the line command action is performed from the hit cache line or has initial valid bit cleared if the command misses. In general, if the valid indicator CLCR[LCIVB] is cleared, the targeted line was invalid at the start of the line command and no line operation was performed.

CLCR[22:20]			For cache address commands	For physical address commands
LCWAY	LCIMB	LCIVB		
0	0	0	Way 0 line was invalid	No hit
0	0	1	Way 0 valid, not modified	Way 0 valid, not modified
0	1	0	Way 0 line was invalid	No hit
0	1	1	Way 0 valid and modified	Way 0 valid and modified
1	0	0	Way 1 line was invalid	No hit
1	0	1	Way 1 valid, not modified	Way 1 valid, not modified
1	1	0	Way 1 line was invalid	No hit
1	1	1	Way 1 valid and modified	Way 1 valid and modified
4'b1xxx			Bit 23 is available for future use, giving 8 more options if necessary.	

Figure 11-13: Line Command Results

At completion of a line command other than a write, the CCVR (Cache R/W Value Register) contains information on the initial state of the line tag or data targeted by the command. For line commands, CLCR[TDSEL] selects between tag and data. If the line command used a physical address and missed, the data is don't care. For write commands, the CCVR holds the write data.

12. Crossbar Switch (XBAR)

12.1. Introduction

12.1.1. Overview

This chapter provides information on the layout, configuration, and programming of the crossbar switch. The crossbar switch connects bus masters and bus slaves using a hardware interconnect matrix. This structure allows all bus masters to access different bus slaves simultaneously with no interference while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

The XBAR has three master ports and eight slave ports. Figure 12-1 shows a block diagram of the XBAR.

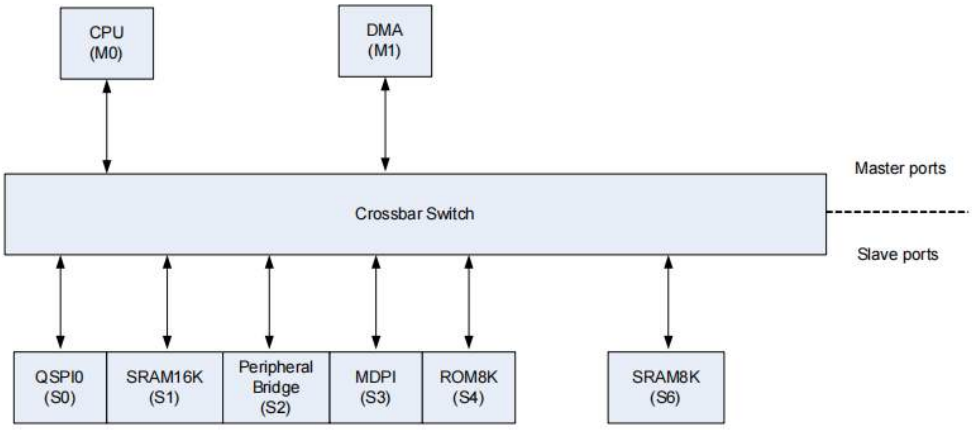


Figure 12-1: XBAR Device-Specific Block Diagram

12.1.2. Features

The crossbar switch includes these distinctive features:

- 2 master ports:
  - CPU
  - DMA
- 6 slave ports
  - ROM memory controller
  - Two Internal SRAM memory controller
  - One QSPI interface
  - EBI interface
  - Peripheral bridges
- Symmetric crossbar bus switch implementation
  - Concurrent accesses from different masters to different slaves
  - Configurable slave arbitration attributes on a slave-by-slave basis
- 32bit address, 32bit data paths
- Fixed priority scheme and fixed parking strategy
- Support for 8, 16, and 32bit single transfers
- Support for low-power park mode

12.2. Memory Map and Registers

This section provides information on XBAR registers.

12.2.1. Register Summary

Each slave port of the crossbar switch contains configuration registers. Read and write transfers of the configuration registers require two bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The slave registers also feature a bit that, when set, prevents the registers from being written. The registers remain readable, but future write attempts have no effect on the registers and are terminated with a bus error response to the master initiating the write.

NOTE:

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the Chip Configuration details for the exact master/slave assignments for your device. Additionally, all references to the crossbar switch registers are based on the physical port connections, not the logical port numbers.

The memory map for the XBAR registers is shown in Table 12-1.

Table 12-1: XBAR Register Configuration Summary

Offset Address	Register	Access(1)
0x0000	Master Priority Register for Slave port 0(MPR0)	S
0x0010	General Purpose Control Register for Slave port 0(SGPCR0)	S
0x0100	Master Priority Register for Slave port 1(MPR1)	S
0x0110	General Purpose Control Register for Slave port 1(SGPCR1)	S
0x0100	Master Priority Register for Slave port 2(MPR2)	S
0x0210	General Purpose Control Register for Slave port 2(SGPCR2)	S
0x0300	Master Priority Register for Slave port 3(MPR3)	S
0x0310	General Purpose Control Register for Slave port 3(SGPCR3)	S
0x0400	Master Priority Register for Slave port 4(MPR4)	S
0x0410	General Purpose Control Register for Slave port 4(SGPCR4)	S
0x0500	Master Priority Register for Slave port 5(MPR5)	S
0x0510	General Purpose Control Register for Slave port 5(SGPCR5)	S
0x0600	Master Priority Register for Slave port 6(MPR6)	S
0x0610	General Purpose Control Register for Slave port 6(SGPCR6)	S
0x0700	Master Priority Register for Slave port 7(MPR7)	S
0x0710	General Purpose Control Register for Slave port 7(SGPCR7)	S

NOTE: *S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.*

12.2.2. XBAR Register Descriptions

12.2.2.1. Master Priority Register (XBAR_MPRn)

The Master Priority Register (MPR) resides in each slave port and sets the priority of each master port on a per slave port basis, e.g., MPR0 sets priority for each master port for slave port 0.

The Master Priority Register can only be accessed in supervisor mode with 32bit accesses. Once the RO (Read Only) bit has been set in the slave General Purpose Control Register the Master Priority Register can only be read from, attempts to write to it will have no effect on the MPR and result in an error response.

Offset Address: 0x0000 + n*0x100

	31	30	29	28	27	26	25	24
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0	0	0	0	0	MSTR2		
Write:								
RESET:	0	0	0	0	0	0	1	0

	7	6	5	4	3	2	1	0
Read:	0	MSTR1				MSTR0		
Write:								
RESET:	0	0	0	1	0	0	0	0

= Writes have no effect and without a transfer error exception.

Figure 12-2: Master Priority Register (XBAR_MPRn)

Table 12-2: XBAR Master Priority Register Field Descriptions

Field	Description
31–11	Reserved
10–8 MSTR2	Master 2 Priority These bits set the arbitration priority for master port 2 (USBC) on the associated slave port. These bits are initialized by hardware reset. The reset value is 010. 000: This master has the level 1(highest) priority when accessing the slave port. 001: This master has the level 2 priority when accessing the slave port. ... 111: This master has the level 3 (lowest) priority when accessing the slave port.

Field	Description
7	Reserved This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.
6–4 MSTR1	Master 1 Priority These bits set the arbitration priority for master port 1 (DMA) on the associated slave port. These bits are initialized by hardware reset. The reset value is 001. 000: This master has the level 1(highest) priority when accessing the slave port. 001: This master has the level 2 priority when accessing the slave port. ... 111: This master has the level 3 (lowest) priority when accessing the slave port.
3	Reserved This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.
2–0 MSTR0	Master 0 Priority These bits set the arbitration priority for master port 0 (CPU) on the associated slave port. These bits are initialized by hardware reset. The reset value is 000. 000: This master has the level 1(highest) priority when accessing the slave port. 001: This master has the level 2 priority when accessing the slave port. ... 111: This master has the level 3 (lowest) priority when accessing the slave port.

NOTE: *No two available master ports can be programmed with the same priority level. Attempts to program two or more available masters with the same priority level will result in an error response and the MPR will not be updated.*

12.2.2.2. Slave General Purpose Control Register (XBAR_SGPCRn)

The Slave General Purpose Control Register (SGPCR) controls several features of each slave port. The Read Only (RO) bit will prevent any registers associated with this slave port from being written to once set. This bit may be written with '0' as many times as the user desires, but once it is written to a '1' only a reset condition will allow it to be written again.

The PCTL bits determine how the slave port will park when no master is actively making a request. The available options are to park on the master defined by the PARK bits, park on the last master to use the slave port, or go into a low power park mode which will force all the outputs of the slave port to inactive states when no master is requesting an access. The low power park feature can result in an overall power savings if the slave port is not saturated; however, it will force an extra clock of latency whenever any master tries to access it when it is not in use because it will not be parked on any master.

The PARK bits determine which master the slave will park on when no master is making an active request. Please use caution to only select master ports that are actually present in the design. If the user programs the PARK bits to a master not present in the current design implementation undefined behavior will result.

NOTE: *The SGPCR can only be accessed in supervisor mode with 32bit accesses. Once XBAR_SGPCR[RO] has been set, the SGPCR can only be read; attempts to write to it will have no effect on the SGPCR and result in an error response.*

Offset Address: 0x0010 + n*0x100

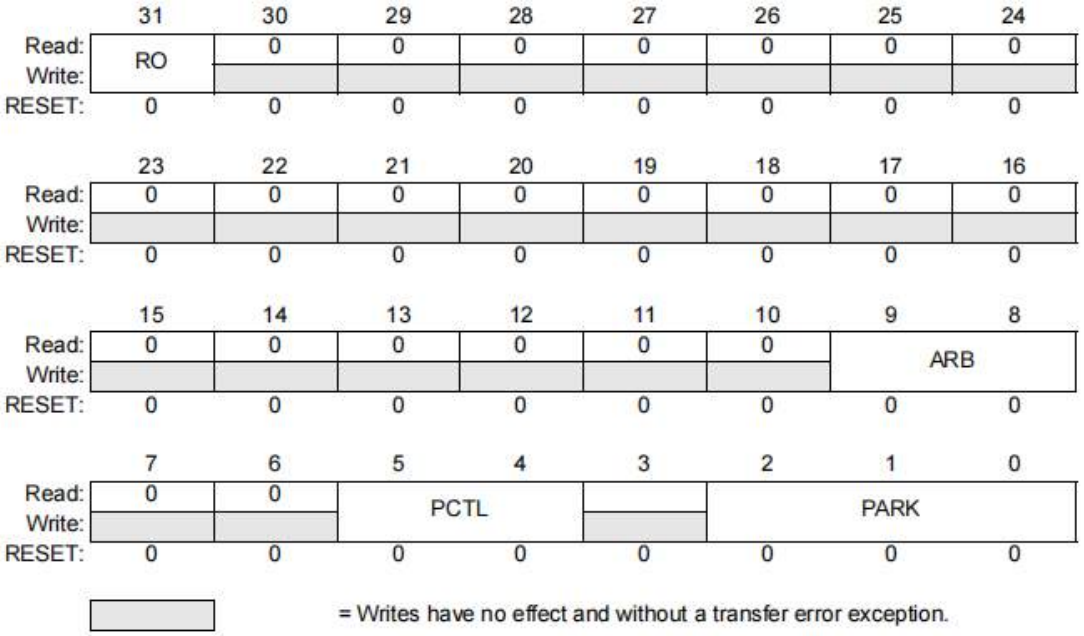


Figure 12-3: Slave General Purpose Control Register (XBAR_SGPCRn)

Table 12-3: XBAR Slave General Purpose Control Register Field Descriptions

Field	Description
31 RO	Read Only This bit is used to force all of a slave port's registers to be read only. Once written to '1' it can only be cleared by hardware reset. This bit is initialized by hardware reset. The reset value is 0. 0: All this slave port's registers can be written. 1: All this slave port's registers are read only and cannot be written (attempted writes have no effect and result in an error response).
30– 10	Reserved These bits are reserved for future expansion. They read as zero and should be written with zero for upward compatibility.

Field	Description
9–8 ARB	Arbitration Mode These bits are used to select the arbitration policy for the slave port. These bits are initialized by hardware reset. The reset value is 00. 00: Fixed Priority 01: Round-Robin (rotating) Priority 10: Reserved 11: Reserved
7–6	Reserved These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.
5–4 PCTL	Parking Control Determines the slave port's parking control. The low-power park feature results in overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. These bits are initialized by hardware reset. The reset value is 00. 00: When no master is making a request the arbiter will park the slave port on the master port defined by the PARK bit field. 01: When no master is making a request the arbiter will park the slave port on the last master to be in control of the slave port. 10: When no master is making a request the arbiter will park the slave port on no master and will drive all outputs to a constant safe state. 11: Reserved
3	Reserved This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.

Field	Description
2–0 PARK	PARK These bits are used to determine which master port this slave port parks on when no masters are actively making requests. These bits are initialized by hardware reset. The reset value is 000. ***NOTE:*** *Only select master ports that are actually present on the device. If not, undefined behavior may occur.* 000: Park on Master Port 0 (CPU) 001: Park on Master Port 1 (DMA) 010: Park on Master Port 2 (USBC) 011: Reserved 100: Reserved 101: Reserved 110: Reserved 111: Reserved

12.3. Function

This section describes in more detail the functionality of the XBAR.

12.3.1. General Operation

When a master sends an access to the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. It is possible to make single-clock (zero wait state) accesses through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Since the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply enters a wait state.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
- An outstanding request to one slave port that has a long response time and
- A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a fixed-length burst transfer it retains control

of the slave port until that transfer completes. When a master has control of a given slave port, the crossbar returns all response information from the slave back to the requesting master.

The crossbar terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave buses). Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus even though a default master may be granted access to the slave port.

When a slave bus is being IDLEd by the crossbar, it can park the slave port on the master port indicated by the XBARx_CRSn[PARK]. This is done in an attempt to save the initial clock of arbitration delay that otherwise would be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low-power park mode in attempt to save power by using XBARx_CRSn[PCTL].

12.3.2. Register Coherency

Since the content of the registers has a real-time effect on the operation of the crossbar, it is important to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave-port-related master accesses; instead, they track only with slave accesses.

12.3.3. Arbitration

The XBAR supports two arbitration schemes: a simple fixed-priority comparison algorithm and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

12.3.3.1. Fixed Priority Operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the XBARx_PRSn (priority registers). If two masters request access to a slave port, the master with the higher priority in the selected priority register gains control over the slave port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port performs an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port, the new requesting master is granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed-length burst transfer or a locked transfer. In this case, the new requesting master must wait until the end of the burst transfer or locked transfer before it is granted control of the slave port.

If the new requesting master's priority level is lower than the master that currently has control of the slave port, the new requesting master is forced to wait until the current master runs one of the following cycles:

- An IDLE cycle
- A non-IDLE cycle to a location other than the current slave port

12.3.3.2. Round-robin Priority Operation

When operating in round-robin mode, each master is assigned a relative priority based on the physical master port number. This relative priority is compared to the ID (master port number) of the last master to perform a transfer on the slave bus. The highest priority requesting master

Crossbar Switch (XBAR)

becomes owner of the slave bus at the next transfer boundary (accounting for locked and

fixed-length burst transfers). Priority is based on how far the ID of the requesting master is ahead of the ID of the last master.

After access is granted to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and

5 make simultaneous requests, they are serviced in the order 4, 5, and then 0.

Parking may continue to be used in a round-robin mode, but it does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

12.3.4. Priority Assignment

Each master port needs to be assigned a unique 3bit priority level. If an attempt is made to program multiple master ports with the same priority level within a register (MPR) the XBAR will respond with an error and the registers will not be updated.

12.4. Initialization/Application information

No initialization is required by or for the crossbar switch. Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. Settings and priorities should be programmed to achieve maximum system performance.

13. Direct Memory Access (DMA)

13.1. Information Specific to This Device

This section presents device-specific parameterization, customization, and feature availability information not specifically referenced in remainder of this chapter.

13.1.1. Device-Specific Features

- 6 programmable channels to support independent 8, 16 or 32bit single value or block transfers
- Support of variable sized queues and circular queues
- Source and destination address registers independently configured to post-incrementor remain constant
- Each transfer initiated by peripheral, CPU, periodic timer interrupt or DMA channel request
- Peripheral DMA request sources possible from QSPI, QADC
- Each DMA channel able to optionally send interrupt request to CPU on completion of single value or block transfer
- DMA transfers possible between system memories and all accessible memory mapped locations including peripheral and registers
- DMA supports the following functionality:
  - Scatter Gather
  - Channel Linking
  - Inner Loop Offset
  - Arbitration

    Fixed Group, fixed channel

    Round Robin Group, fixed channel

    Round Robin Group, Round Robin Channel

    Fixed Group, Round Robin Channel
  - Channel preemption
  - Cancel channel transfer
- Interrupts – The DMA has a single interrupt request for each implemented channel and a combined DMA Error interrupt to flag transfer errors to the system. Each channel DMA interrupt can be enabled or disabled and provides notification of a completed transfer. Refer to the Interrupt Vector in the EIC chapter of the reference manual for the allocation of these interrupts.

13.1.2. Channel Assignments

The assignments between the DMA requests from the blocks to the channels of the DMA are shown in Table 13-1.

Table 13-1: DMA Channel Assignment

DMA Request	Channel	Description
ADC_ISR[EMPTY]	0	qadc_dma_req when the data number in the FIFO is not empty and bit ADC_CFGR1 [DMAEN] is set,
QSPI0 DMA Receive Request	1	QSPI0 dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above DMARDLR[DMARDL] + 1, and DMACR[RDMAE]=1
QSPI0 DMA Transmit Request	2	QSPI0 dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below DMATDLR[DMATDL], and DMACR[TDMAE] = 1
SPI0 DMA Receive Request	3	SPI0 dma_rx_req signal is generated when the number of valid data entries in the receive FIFO is more than SPIDMATHR[RXDMATH] or RX FIFO TimeOut, and SPIDMACR[RXDMAE] = 1
SPI0 DMA Transmit Request	4	SPI0 dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is less than SPIDMATHR[TXDMATH] or TX FIFO TimeOut, and SPIDMACR[TXDMAE] = 1
SPI1 DMA Receive Request	5	SPI1 dma_rx_req signal is generated when the number of valid data entries in the receive FIFO is more than SPIDMATHR[RXDMATH] or RX FIFO TimeOut, and SPIDMACR[RXDMAE] = 1
SPI1 DMA Transmit Request	6	SPI1 dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is less than SPIDMATHR[TXDMATH] or TX FIFO TimeOut, and SPIDMACR[TXDMAE] = 1
PIT1 DMA Request	7	pit1_dma_req signal is generated when the PIT0 counter reaches 0x0000 and PCSR[PDMAE] = 1
PIT0 DMA Request	8	pit0_dma_req signal is generated when the PIT0 counter reaches 0x0000 and PCSR[PDMAE] = 1
Reserved	9	Reserved
SCI0 TX DMA Request	10	sci0_tx_req signal is generated when the number of datawords in the transmit FIFOis equal to or less than the number indicated by SCI_WATER[TXWATER])
SCI0 RX DMA Request	11	sci0_rx_req signal is generated when the number of datawords in the receive buffer is greater than the number indicated by SCI_WATER[RXWATER]
SCI1 TX DMA Request	12	sci1_tx_req signal is generated when the number of datawords in the transmit FIFO is equal to or less than the number indicated by SCI_WATER[TXWATER])
SCI1 RX DMA Request	13	sci1_rx_req signal is generated when the number of datawords in the receive buffer is greater than the number indicated by SCI_WATER[RXWATER]
Reserved	14	Reserved

DMA Request	Channel	Description
Reserved	15	Reserved

13.2. Introduction

The direct memory access controller (DMA) is capable of performing complex data movements through 16 programmable channels, with minimal intervention from the host processor. The hardware microarchitecture includes a DMA engine that performs source and destination address calculations, and the actual data movement operations, along with an SRAM-based memory containing the transfer control descriptors (TCD) for the channels. This implementation minimizes the overall block size.

Figure 13-1 is a block diagram of the DMA module.

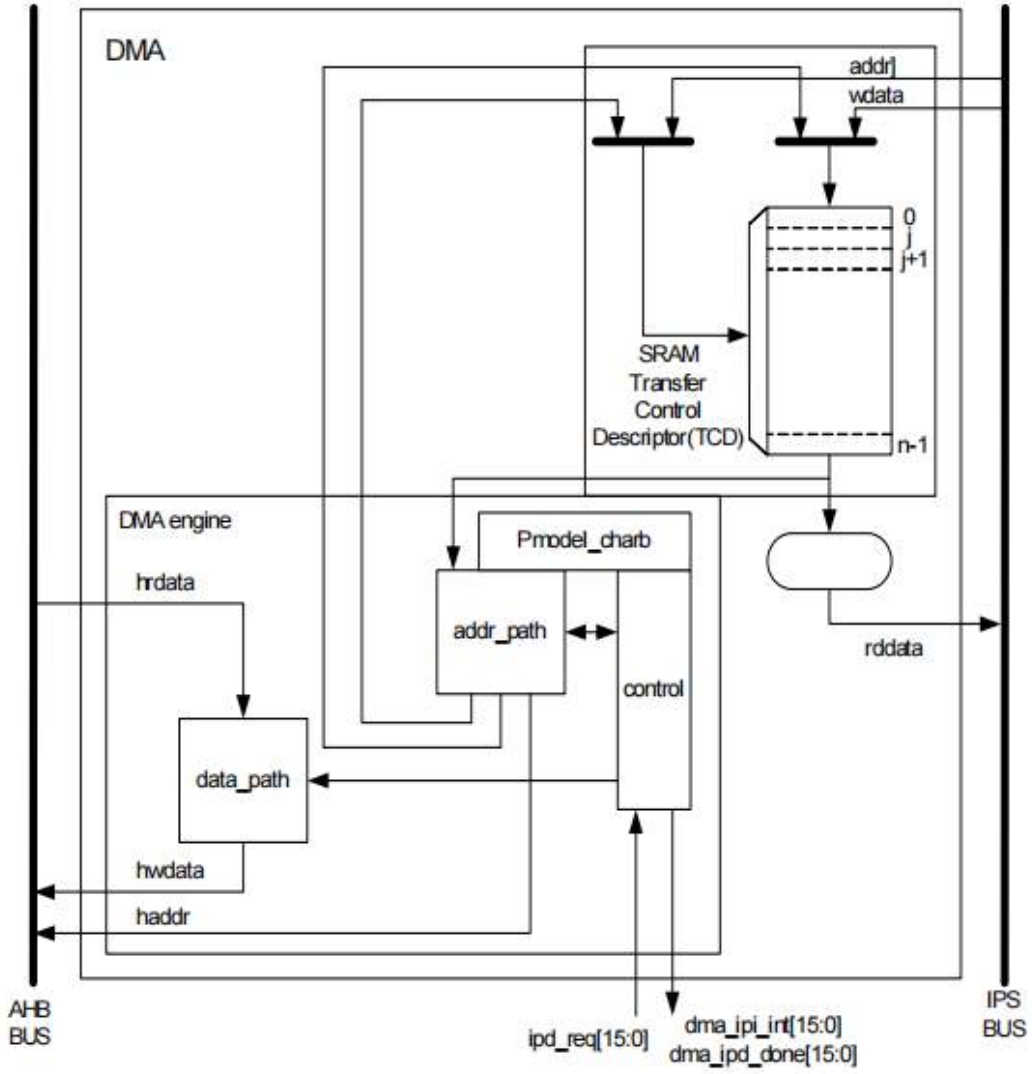


Figure 13-1: DMA Block Diagram

13.2.1. Features

The DMA module supports the following features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source, destination addresses, transfer size, plus support for enhanced addressing modes

- Transfer control descriptor organized to support two-deep, nested transfer operations
  - An inner data transfer loop defined by a “minor” byte transfer count
  - An outer data transfer loop defined by a “major” iteration count

- Channel service request via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers

  Independent channel linking at end of minor loop and/or major loop
  - Peripheral-paced hardware requests (one per channel)
  - For all three methods, one service request per execution of the minor loop is required

- Support for fixed-priority and round-robin channel arbitration

- Channel completion reported via optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Error terminations are optionally enabled per channel, and logically summed together to form a small number of error interrupt outputs

- Support for scatter/gather DMA processing

- Support for complex data structures

- Support to cancel transfers via software or hardware

14. Option Byte (OPB)

14.1. Register Memory Map

Table 14-1: Register Memory Map

Address Offset	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0	Access[1,2]
0x001C	CCR		PVDC		S
0x0020	PLLOCKCR		EIOSCST		S
0x0024	PVDFEVR		RFEVR		S
0x002C	Reserved[3]				S
0x0030	IOSTC				S
0x0034	LDOTCR	VREFTCR	Reserved		S
0x0038	RTCTCR		EOSCTCR		S

NOTE:

1. *S = CPU supervisor mode access only.*
2. *User mode accesses to supervisor-only address locations have no effect and result in a cycle termination transfer error.*
3. *Writes to reserved address locations have no effect and reads return 0s.*

14.1.1. Register Descriptions

14.1.1.1. PVDC — Programmable Voltage Detector Configuration Register

Offset Address: 0x001C

	15	14	13	12	11	10	9	8
Read:	PVDF	PVD-PORRE	PVDTEST[1:0]		PVDOE	PVDRE	PVDIE	PVDE
Write:								
RESET:	0	0	0	0	1	0	0	1

	7	6	5	4	3	2	1	0
Read:	0	0	0	0	0	0	PVDC[1:0]	
Write:								
RESET:	0	0	0	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 14-1: PVDC — Programmable Voltage Detector Configuration Register

PVDTEST[1:0] — Write Access Enable Sequence Input

The PVDC register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the PVDC register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

PVDF — Programmable Voltage Detector Flag

The PVDF indicates VDD is lower than PVD threshold. POR can clear it.

1 = VDD33 is lower than PVD threshold

0 = VDD33 is not lower than PVD threshold

PVDOE — Programmable Voltage Detector Output Enable

1 = PVD Output Enable

0 = PVD Output Disable

PVDE — Programmable Voltage Detector Enable

The PVDE Config if PVD is enabled. POR can clear it.

1 = Enable PVD

0 = Disable PVD

PVDPORRE — Program Voltage Detector (PVD) Power-On Reset Enable

The PVDPORRE shows whether Program Voltage Detector Power-On reset is enabled or not. When enabled, RSR[POR] flag will be set after PVD reset.

1 = PVD will generate Power-On reset when VDD33 is lower than PVD threshold

0 = PVD won't generate Power-On reset when VDD33 is lower than PVD threshold

PVDRE — Program Voltage Detector (PVD) Reset Enable

The PVDRE shows whether Program Voltage Detector reset is enabled or not. When enabled and PVDPORRE is disabled status, RSR[PVDF] flag will be set after PVD reset.

1 = PVD reset enabled

0 = PVD reset disabled

PVDIE — Programmable Voltage Detector Interrupt Enable

The PVDRE config if VCC is lower than PVD threshold can generate an interrupt

1 = VCC is lower than PVD threshold generates an interrupt

0 = VCC is lower than PVD threshold not generates an interrupt

PVDC[1:0] — Programmable Voltage Detector Configuration Value.

Table 14-2: PVDC Setting Table

PVDC	Detect Voltage
2'b00	2.16V
2'b01	2.32V
2'b10	2.48V
2'b11	2.64V

14.1.1.2. CCR — Customer Configuration Register

Offset Address: 0x001E

	31	30	29	28	27	26	25	24
Read:	0	0	RTC_INTERFACE_EN	0	0	0	RTC_INTERFACE_LVD_ISOEN	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	CLKMD	0	0	0	0	0	CCRTEST[1:0]	
Write:								
RESET:	Note1	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of CCR[CLKMD]*

Figure 14-2: CCR — Customer Configuration Register

CCRTEST[1:0] — Write Access Enable Sequence Input

The writable bit of CCR register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of CCR register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

RTC_INTERFACE_EN — This bit Determine whether the RTC analog module can be reconfigured or not when PRCSR[Dir] or PRENR[RTC_EN_Dir] is set.

1 = RTC analog module can be reconfigured.

0 = RTC analog module cannot be reconfigured.

RTC_INTERFACE_LVD_ISOEN — This bit Determine whether the RTC analog interface will be isolated or not when Program Voltage Detector event happens.

1 = RTC analog interface won't be isolated when Low voltage happens.

0 = RTC analog interface will be isolated when Low voltage happens.

CLKMD — Clock Mode Control Bit

The CLKMD reflects the clock source of the chip and is determined by external CLKMODE pin.

Table 14-3: Clock Mode Setting Table

CLKMD	Description
1'b1	FIRC150MHz
1'b0	FXOSC

14.1.1.3. EIOSCST — External or Internal High Speed Oscillator Stable Time Configuration Register

Offset Address: 0x0020

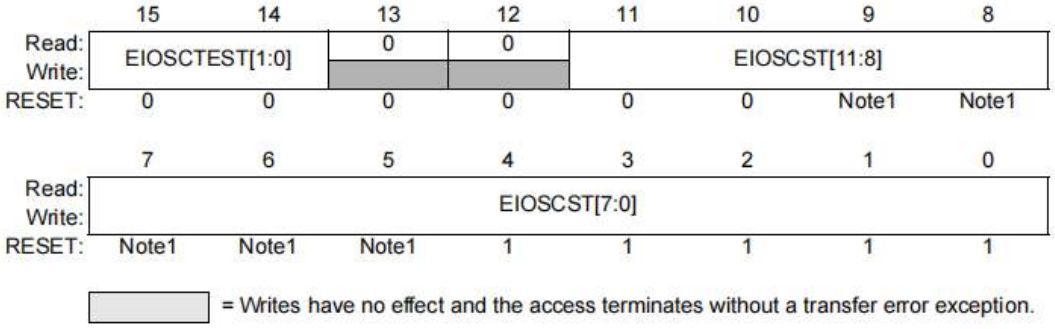


Figure 14-3: EIOSCST — External or Internal High Speed Oscillator Stable Time Configuration Register

EIOSCTEST[1:0] — Write Access Enable Sequence Input

The writable bit of EIOSCST register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of EIOSCST register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

EIOSCST[11:0] — External or Internal High Speed Oscillator Stable Time Value.

After External or Internal Oscillator is enabled, it will wait EIOSCST[11:0] cycles of 128KHz oscillator and then the gate of the clock will be turn-on. The default value is 0x1F when CLKMD is set, otherwise is 0x3FF.

14.1.1.4. PLLOCKCR — PLL Lock Time Configuration Register

Offset Address: 0x0022

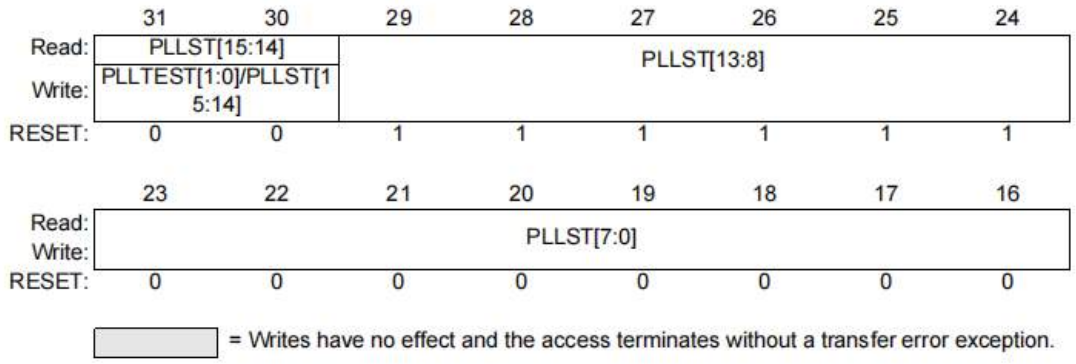


Figure 14-4: PLLOCKCR — PLL Lock Time Configuration Register

PLLTEST[1:0] — Write Access Enable Sequence Input

The PLLST register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these

two bits' value == 2'b11,then the PLLST register can be changed at will. Any writes can clear these two bits when the value equals to 2'b11.

PLLST[15:0] — PLL Lock Time Value.

After enabled, the PLL will wait PLLST[15:0] cycles of External High speed oscillator (FXOSC) and then the gate of the clock will be turn-on.

NOTE: *PLLST[15:14] is writable only when the value of PLLTEST[1:0] equals to 2'b11.*

14.1.1.5. RFEVR — RESET Pin Filter Enable and Value Register

Offset Address: 0x0024

	15	14	13	12	11	10	9	8
Read:	0	0	RFEVRTEST[1:0]		0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	RFV[7:0]							
Write:								
RESET:	0	0	0	0	1	1	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 14-5: RFEVR — RESET pin Filter Enable and Value Register

PVDFTEST[1:0] — Write Access Enable Sequence Input

The writable bit of RFEVR register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of RFEVR register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

RFV[7:0] — RESET Pin Filter Value

These bits determine the minimum pulse width of external RESET pin.

14.1.1.6. PVDFEVR — Programmable Voltage Detector Filter Enable and Value Register

Offset Address: 0x0026

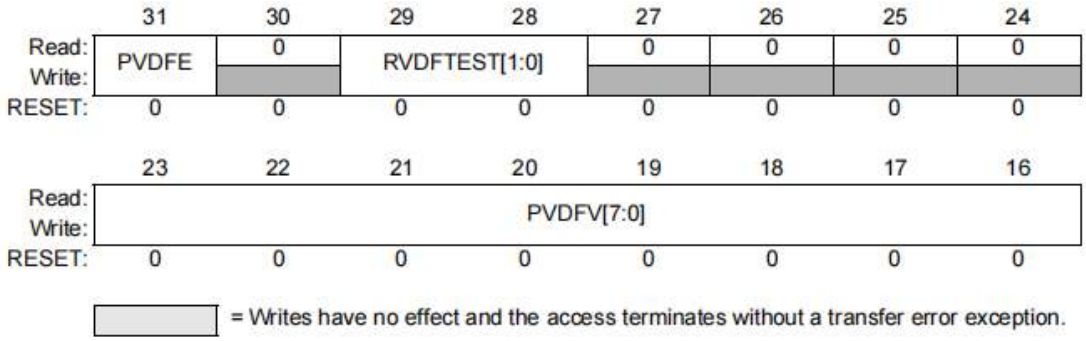


Figure 14-6: PVDFEVR — Programmable Voltage Detector Filter Enable and Value Register

PVDFTEST[1:0] — Write Access Enable Sequence Input

The writable bit of PVDFEVR register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of PVDFEVR register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

PVDFE — Programmable Voltage Detector Filter Enable.

The PVDFE shows whether Program Voltage Detector Filter is enabled.

1 = PVD filter is enabled

0 = PVD filter is not enabled

PVDFV[7:0] — Programmable Voltage Detector Filter Value

These bits determine the minimum pulse width of internal Programmable Voltage Detector.

14.1.1.7. IOSCTC — Internal High Speed Oscillator Trimming Configuration Register

Offset Address: 0x0030

	31	30	29	28	27	26	25	24
Read:	IOSCTCTEST[1:0]		0	0	0	0		
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	CTRIM_LV [1]
Write:								
RESET:	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8
Read:	CTRIM_LV [0]	CTRIM_TD_LV[6:0]						
Write:								
RESET:	1	0	0	0	0	1	1	1

	7	6	5	4	3	2	1	0
Read:	FTRIM_LV[4:0]					TEMPTRIM_LV[2:0]		
Write:								
RESET:	0	1	1	1	1	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 14-7: IOSCTC — Internal High Speed Oscillator Trimming Configuration Register

IOSCTCTEST[1:0] — IOSCTC Write Access Sequence In

The writable bit of IOSCTC register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of IOSCTC register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

CTRIM_LV[1:0] — Coarse Trimming Value for internal high speed oscillator

CTRIM_TD_LV[4:0] — Coarse Trimming Value for internal high speed oscillator

FTRIM_LV[6:0] — Fine Trimming Value for internal high speed oscillator

TEMPTRIM_LV[2:0] — Temperature correction for internal high speed oscillator.

14.1.1.8. VREFTCR — VREF1P2 Trimming Configuration Register

Offset Address: 0x0036

	23	22	21	20	19	18	17	16
Read: Write:	VREFTCTEST[1:0]		VREFPD	VREFTRIM[4:0]				
RESET:	0	0	1	1	1	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of FUSE Area*

Figure 14-8: VREFTCR — VREF Trimming Configuration Register

VREFTCTEST[1:0] — WSFTCR Write Access Sequence In

The writable bit of VREFTCTEST register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of VREFTCTEST register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value
has no effect and returns 2'b11.

VREFPD — Internal VREF1P2 Module Power Down. The setup time is 10us after Internal VREF1P2 Module is turn on.

1 = Internal VREF1P2 Module Power Down

0 = Internal VREF1P2 Module Power On

VREFTRIM[4:0] — VREF1P2 Module Trimming value

14.1.1.9. LDO1P2TC — LDO1P2 Trimming Configuration Register

Offset Address: 0x0037

	31	30	29	28	27	26	25	24
Read:	LDO1P5TC[7:6]		LDO1P5TC[5:0]					
Write:	LDO1P2TEST[1:0]/ LDO1P5TC[7:6]							
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of FUSE Area*

Figure 14-9: LDO1P5TC — LDO1P2 Trimming Configuration Register

LDO1P2TEST[1:0] — LDO1P2TC Write Access Sequence In

The writable bit of LDO1P2TC register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of LDO1P2TC register can be changed at will. Any writes can clear these two bits when the value equals to 2'b11.

LDO1P2TC[5:0] — LDO1P2 Trimming Value

LDO1P2TC[6] — LDO1P2 power down control bit

1 = LDO1P2 will be power down.
0 = LDO1P2 will be power on.

LDO1P2TC[7] — LDO over-current limit function control bit
1 = LDO over-current limit function will be turned off.
0 = LDO over-current limit function will be turned on.

NOTE: *LDO1P2TC[7:6] is writable only when the value of LDO1P2TEST[1:0] equals to 2'b11.*

14.1.1.10. RTCTCR — RTC Trimming Configuration Register

Offset Address: 0x003A

	31	30	29	28	27	26	25	24
Read:	RTCTCTEST[1:0]		0	0	0	0	RTC_TEST_MODE	Trim_load
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	RTC_BIAS_TRIM[3:0]				RTC_CAP_TRIM[3:0]			
Write:								
RESET:	0	1	1	1	1	1	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of FUSE Area*

Figure 14-10: RTCTCR — RTC Trimming Configuration Register

RTCTCTEST[1:0] — RTCTC Write Access Sequence In

The writable bit of RTCTC register cannot be changed, unless the correct sequence write in. The right sequence is :2'b01->2'b10->2'b11.After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of RTCTC register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

RTC_BIAS_TRIM[3:0] — RTC Oscillator Bias Trimming configuration bits.

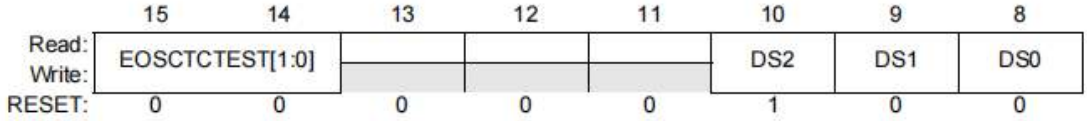
RTC_CAP_TRIM[3:0] — RTC Oscillator Load Capacitance C1 and C2 Trimming configuration bits.

Trim_load — Low to high edge will latch RTC_BIAS_TRIM and RTC_CAP_TRIM value into RTC analog module.

RTC_TEST_MODE — RTC Test Mode enable.
1 = RTC Test Mode is enabled.
0 = RTC Test Mode is disabled.

14.1.1.11. EOSCTCR — External Oscillator rimming Configuration Register

Offset Address: 0x0039



- *Note:1, Determined by the value of FUSE Area*

Figure 14-11: EOSCTCR — External Oscillator Trimming Configuration Register

EOSCTCTEST[1:0] — EOSCTCR Write Access Sequence In

The writable bit of EOSCTCR register cannot be changed, unless the correct sequence write in. The right sequence is : 2'b01->2'b10->2'b11. After write these two bits following this sequence, these two bits' value == 2'b11,then the writable bit of EOSCTCR register can be changed at will. Only writes 2'b00 can clear these two bits when the value equals to 2'b11.Writes other value has no effect and returns 2'b11.

EOSCTRIM[2:0] — *External Oscillator* Trimming value

Table 14-3: GM of Oscillator Cell With Different Drive Strength

Driving Select			GM(mA/V)
DS2	DS1	DS0	
0	0	0	3.6
0	0	1	6.9
0	1	0	10.3
0	1	1	13.2
1	0	0	16.1
1	0	1	18.9
1	1	0	21.5
1	1	1	24.3

15. External Bus Interface (EBI)

15.1. Introduction

LT165 has an embedded External Bus Interface (EBI) for controlling the transfer of the information between internal bus and external 8bit MCU Display Panel. One asynchronous chip select channel are available.

15.2. Features

Features of the MCU Display Panel Interface include:

- Reduced system complexity
  - No external glue logic required for typical systems if chip selects are used.
- One programmable asynchronous active-low chip selects.
- Programmable chip selects wait cycle
  - To interface with various panels, up to 64 chip selects asserted cycle can be programmed.
- Programmable read/write asserted cycle
- Programmable read/write negated cycle

15.3. Signal Description

Table 15-1 provides an overview of the signals described below.

Table 15-1: Signal Properties

Name	Function	Pullup
EBI_D[7:0]	Data bus	—
EBI_WR#	Write enable and low active	Active
EBI_RD#	Read enable and low active	Active
EBI_RS	Command or Data indication	Active
EBI_CS#	Chip select and low active	Active

15.4. Module Memory Map

Table 15-2 shows the register memory map of EBI Module:.

Table 15-2: Register Memory Map

Offset Address	Bits 31-16	Bits 15-0	Access[1,2]
0x0000	EBICR0 — EBI Control Register 0		S
0x0004	EBICR1 — EBI Control Register 1		S

NOTE:

1. S = CPU supervisor mode access only.

2. User mode accesses to supervisor-only address locations have no effect and result in a cycle termination transfer error.

15.5. Register Descriptions

15.5.1. EBI Control Registers 0 (EBICR0)

Offset Address: 0x0000

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	WS[7:0]							
W								
RESET:	0	0	0	0	0	1	1	1

	7	6	5	4	3	2	1	0
R	PS[1:0]		0	0	0	0	EBI_LIT_ENDIAN	CSEN
W								
RESET:	0	1	0	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 15-1: EBI Control Register 0 (EBICR0)

PS[1:0] — Port Size Bits

The PS bit defines the width of the external data port supported by the chip select as either 8bit or 16bit. This field is read only in this device.

00 = Not supported, will lead to an unpredictable error
10 = 16bit port
01 = 8bit port
11 = Not supported, will lead to an unpredictable error

EBI_LIT_ENDIAN — EBI ports is little endian

1 = EBI ports is little endian.
0 = EBI ports is big endian.

WS[7:0] — Wait States Field

The WS field determines the number of wait states for the chip select logic to insert before asserting the internal cycle termination signal. One wait state is equal to one system clock cycle. If WS is configured for zero wait states, then the internal cycle termination signal is asserted in the clock cycle following the start of the cycle access, resulting in one-clock transfers. A WS configured for one wait state means that the internal cycle termination signal is asserted two clock cycles after the start of the cycle access.

Since the internal cycle termination signal is asserted internally after the programmed number of wait states, software can adjust the bus timing to accommodate the access speed of the external device. With up to 256 possible wait states, even slow devices can be interfaced with the MCU.

Table 15-3: EBI_CS# Chip Select Wait States Encoding

WS[7:0]	Number of EBI_CS# Cycles	
	Read Access	Write Access
00000000	1	1
00000001	2	2
00000010	3	3
00000011	4	4
00000100	5	5
00000101	6	6
00000110	7	7
00000111	8	8
00001000	9	9
00001001	10	10
00001010	11	11
00001011	12	12
00001100	13	13
00001101	14	14
..	...	...
11111111	256	256

CSEN — Chip Select Enable Bit

The CSEN bit enables the chip select logic. When the chip select function is disabled, the EBI_CS signal is negated high.

1 = Chip select function enabled.

0 = Chip select function disabled.

15.5.2. EBI Control Registers 1 (EBICR1)

Offset Address: 0x0004

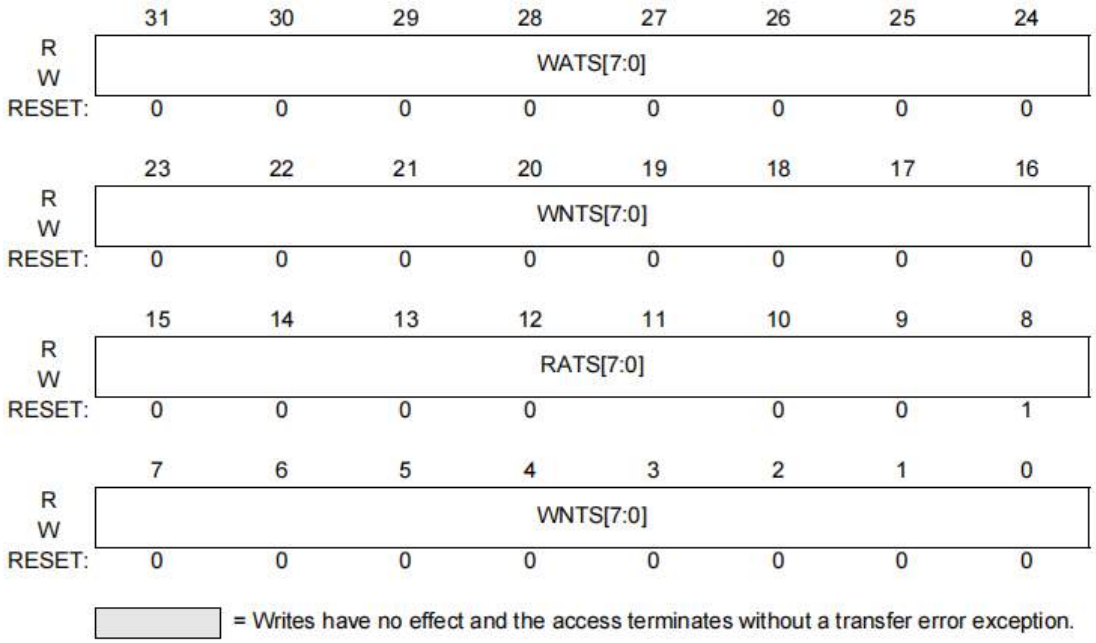


Figure 15-2: EBI Control Register 1 (EBICR1)

WATS[7:0] — Write Signal Assert Timing Select

These bits select the assertion timing of EBI_WR#, when the operation is written. See functional description for details.

WNTS[7:0] — Write Signal Negate Timing Select

These bits select the negation timing of EBI_WR#, when the operation is written. See functional description for details.

RATS[7:0] — Read Assert Timing Select

These bits select the assertion timing of EBI_RD#, when the operation is read. See functional description for details.

RNTS[7:0] — Read Assert Timing Select

These bits select the negation timing of EBI_RD#, when the operation is read. See functional description for details.

15.6. Functional Description

15.6.1. EBI_WR#, EBI_RD# Signal Timing

For write and read operation, EBI_WR# and EBI_RD# assert and negate timing can be configured.25

Table 15-4: EBI_WR# and EBI_RD# Assert and Negate Timing (Unit: System Cycle)

RATS[7:0]/WATS[7	Assert Time (T_{ass})	RNTS[7:0]/WNTS[7	Negate Time
00000000	1/2	00000000	1/2
00000001	2/2	00000001	2/2
00000010	3/2	00000010	3/2
00000011	4/2	00000011	4/2
00000100	5/2	00000100	5/2
...	...	...	...
11111110	2555/2	11111110	255/2
11111111	256/2	11111111	256/2

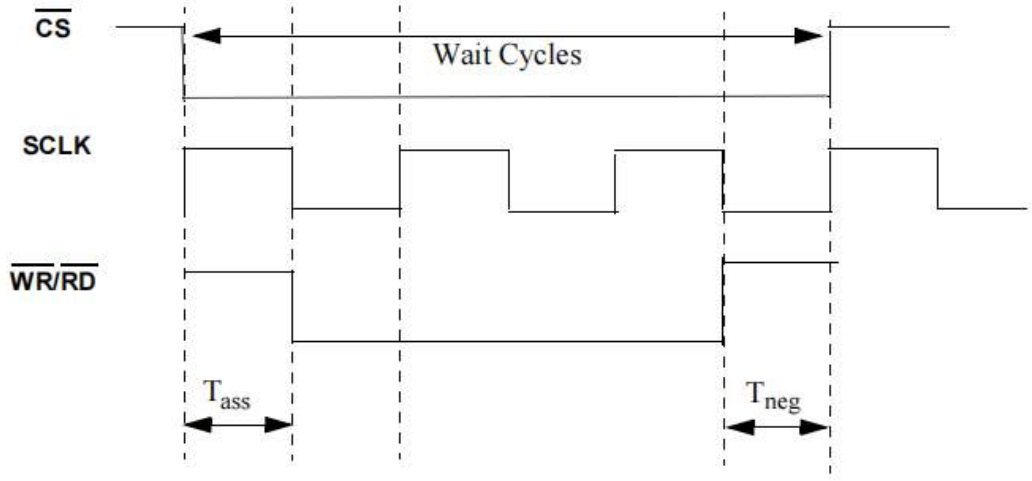


Figure 15-3: EBI_WR#, EBI_RD# Assert/Negate Timing

15.6.2. EBI Functional Description

15.6.2.1. Chip Selects

Chip select can provide a chip enable signal for MCU Panel and assert the internal bus cycle termination signal.

Setting the CSEN bit in EBICR enables the chip select to provide a panel chip enable signal.

The configuration of the active chip select(CSEN), the wait state (WS) field, the port size (PS) field is used for the access.

Table 15-5: Chip Select Address Range Encoding

Chip Select	Block Size	Normal Mode Address Range
EBI_CS#	1MB	0x2000_0000 - 0x2FFF_FFFF

15.6.2.2. Operand Transfer

The possible operand accesses for the internal AHB bus are:

- Byte
- Aligned upper half-word
- Aligned lower half-word
- Aligned word

No misaligned transfers are supported. The EBI controls the byte, half-word, or word operand transfers between the AHB bus and an 8bit or16bit port. "Port" refers to the width of the data path that an external device uses during a data transfer. Each port is assigned to particular bits of the data bus. For each chip select, an 8bit port is assigned to pins D[31:24], a 16bit port is assigned to pins D[31:16].

Table 15-6 to Table 15-7 shows each possible transfer size, alignment, and port width of EBI_CS#. The data bytes shown in the table represent external data pins. This data is multiplexed and driven to the external data bus as shown. The bytes labeled with a dash are not required; the bus will ignore them on read transfers, and driven them with undefined data on write transfers.

Table 15-6: 8bit Port Data Transfer of EBI_CS# (big endian)

Transfer Size	Port Width	Internal Addr		External Pins	Data Bus Transfer			
		Bit1	Bit0	EBI_RS				
Byte	8	0	0	0	D[15:8]	-	-	-
		0	1	1	-	D[15:8]	-	-
		1	0	0	-	-	D[15:8]	-
		1	1	1	-	-	-	D[15:8]
Half-word	8	0	0	0	D[15:8]	-	-	-
				1	-	D[15:8]	-	-
		1	0	0	-	-	D[15:8]	-
				1	-	-	-	D[15:8]
Word	8	0	0	0	D[15:8]	-	-	-
				1	-	D[15:8]	-	-
				0	-	-	D[15:8]	-
				1	-	-	-	D[15:8]

Table 15-7: 16bit Port Data Transfer of EBI_CS# (big endian)

Transfer Size	Port Width	Internal Addr		External Pins	Data Bus Transfer			
		Bit1	Bit0	EBI_RS				
Byte	16	0	0	0	D[15:8]	-	-	-
		0	1	0	-	D[7:0]	-	-
		1	0	1	-	-	D[15:8]	-
		1	1	1	-	--	-	D[7:0]
Half-word	16	0	0	0	D[15:8]	D[7:0]	-	-
		1	0	1	-	-	D[15:8]	D[7:0]
Word	16[1]	0	0	0	D[15:8]	D[7:0]	-	-
				1	-	-	D[15:8]	D[7:0]

NOTE:

1. The EBI runs two cycles for word accesses to 16bit ports. The table shows the data placement for both bus cycles.

16. Programmable Interrupt Timer Modules (PIT)

16.1. Introduction

The programmable interrupt timer (PIT) is a 16bit timer that provides precise interrupts at regular intervals with minimal processor intervention. The timer can either count down from the value written in the modulus latch, or it can be a free-running down-counter.

16.2. Block Diagram

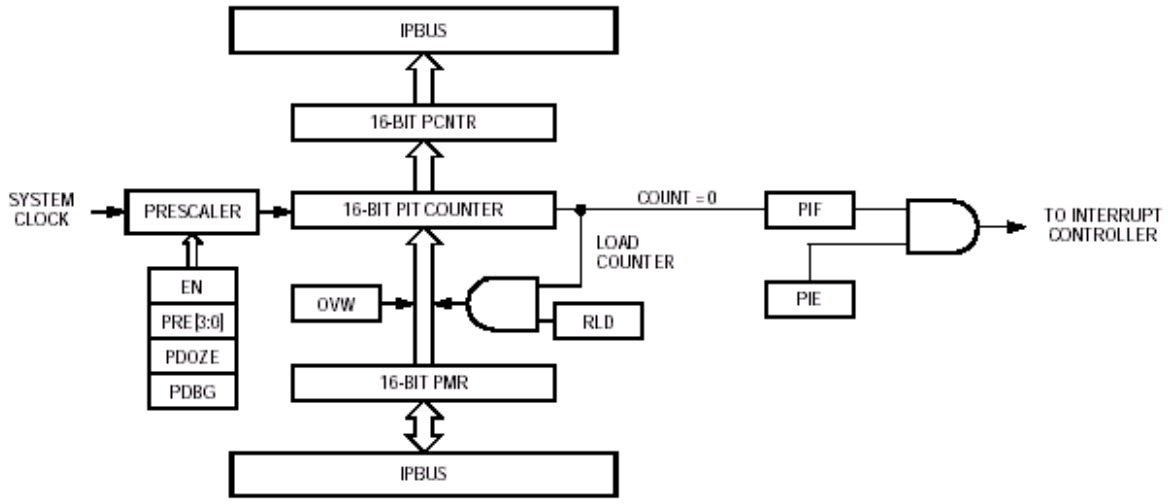


Figure 16-1: PIT Block Diagram

16.3. Modes of Operation

This subsection describes the three low-power modes and the debug mode.

16.3.1. Wait Mode

In wait mode, the PIT module continues to operate normally and can be configured to exit the low-power mode by generating an interrupt request.

16.3.2. Doze Mode

In doze mode with the PDOZE bit set in the PIT Control and Status Register (PCSR), PIT module operation stops. In doze mode with the PDOZE bit clear, doze mode does not affect PIT operation. When doze mode is exited, PIT operation continues from the state it was in before entering doze mode.

16.3.3. Stop Mode

In stop mode, the system clock is absent, and PIT module operation stops.

16.3.4. Debug Mode

In debug mode with the PDBG bit set in PCSR, PIT module operation stops. In debug mode with the PDBG bit clear, debug mode does not affect PIT operation. When debug mode is exited, PIT operation continues from the state it was in before entering debug mode, but any updates made in debug mode remain.

16.4. Signals

The PIT module has no off-chip signals.

16.5. Memory Map and Registers

This subsection describes the memory map and register structure for PIT.

16.5.1. Memory Map

Refer to Table 16-1 for a description of the memory map. This device has four programmable interrupt timers.

Table 16-1: Programmable Interrupt Timer Module Memory Map

PIT1 Address	Bits 15-8	Bits 7-0	Access(1)
0x0000	PIT Modulus Register (PMR)		S
0x0002	PIT Control and Status Register (PCSR)		S
0x0004	Unimplemented (2)		—
0x0006	PIT Count Register (PCNTR)		S/U

NOTE:

1. *S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.*
2. *Accesses to unimplemented address locations have no effect and result in a cycle termination transfer error.*

16.5.2. Registers

The PIT programming model consists of these registers:

- The PIT Control and Status Register (PCSR) configures the timer's operation. See 16.5.2.1 PIT Modulus Register.
- The PIT Modulus Register (PMR) determines the timer modulus reload value. See 16.5.2.3 PIT Count Register.
- The PIT Count Register (PCNTR) provides visibility to the counter value. See 16.5.2.3 PIT Count Register.

16.5.2.1. PIT Modulus Register

The 16bit read/write PIT Modulus Register (PMR) contains the timer modulus value for loading into the PIT counter when the count reaches 0x0000 and the RLD bit is set.

When the OVW bit is set, PMR is transparent, and the value written to PMR is immediately loaded into the PIT counter. The prescaler counter is reset anytime a new value is loaded into the PIT counter and also during reset. Reading the PMR returns the value written in the modulus latch. Reset initializes PMR to 0xFFFF.

Offset Address: 0x0000

	15	14	13	12	11	10	9	8
R W	PM15	PM14	PM13	PM12	PM11	PM10	PM9	PM8
RESET:	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
R W	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
RESET:	1	1	1	1	1	1	1	1

Figure 16-2: PIT Modulus Register (PMR)

16.5.2.2. PIT Control and Status Register

Offset Address: 0x0002

	15	14	13	12	11	10	9	8
R W	0	0	0	0	PRE3	PRE2	PRE1	PRE0
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	PDMAE	PDOZE	PDBG	OVW	PIE	PIF	RLD	EN
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 16-3: PIT Control and Status Register (PCSR)

PRE[3:0] — Prescaler Bits

The read/write PRE[3:0] bits select the system clock divisor to generate the PIT clock as **Table 16-2** shows.

To accurately predict the timing of the next count, change the PRE[3:0] bits only when the enable bit (EN) is clear. Changing the PRE[3:0] resets the prescaler counter. System reset and the loading of a new value into the counter also reset the prescaler counter. Setting the EN bit and writing to PRE[3:0] can be done in this same write cycle. Clearing the EN bit stops the prescaler counter.

Table 16-2: Prescaler Select Encoding

PRE[3:0]	IPS Clock Divisor
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1,024
1011	2,048
1100	4,096
1101	8,192
1110	16,384
1111	32,768

PDMAE — DMA Enable ControlBit

The read/write PDMAE bit control whether a dma request will be generated or not when the PIT counter reaches 0x0000.

1 = Dma request will be generated when the PIT counter reaches 0x0000.
0 = Dma request won't be generated when the PIT counter reaches 0x0000.

PDOZE — Doze Mode Bit

The read/write PDOZE bit controls the function of the PIT in doze mode. Reset clears PDOZE.

1 = PIT function stopped in doze mode
0 = PIT function not affected in doze mode

When doze mode is exited, timer operation continues from the state it was in before entering doze mode.

PDBG — Debug Mode Bit

The read/write PDBG bit controls the function of the PIT in debug mode. Reset clears PDBG.

1 = PIT function stopped in debug mode
0 = PIT function not affected in debug mode

During debug mode, register read and write accesses function normally. When debug mode is exited, timer operation continues from the state it was in before entering debug mode, but any updates made in debug mode remain.

NOTE: *Changing the PDBG bit from 1 to 0 during debug mode starts the PIT timer. Likewise, changing the PDBG bit from 0 to 1 during debug mode stops the PIT timer.*

OVW — Overwrite Bit

The read/write OVW bit enables writing to PMR to immediately overwrite the value in the

PIT counter.
1 = Writing PMR immediately replaces value in PIT counter.
0 = Value in PMR replaces value in PIT counter when count reaches 0x0000.

PIE — PIT Interrupt Enable Bit

The read/write PIE bit enables the PIF flag to generate interrupt requests.
1 = PIF interrupt requests enabled
0 = PIF interrupt requests disabled

PIF — PIT Interrupt Flag

The read/write PIF flag is set when the PIT counter reaches 0x0000. Clear PIF by writing a 1 to it or by writing to PMR or acknowledged by dma request. Writing
0 has no effect. Reset clears PIF.
1 = PIT count has reached 0x0000.
0 = PIT count has not reached 0x0000.

RLD — Reload Bit

The read/write RLD bit enables loading the value of PMR into the PIT counter when the count reaches 0x0000.
1 = Counter reloaded from PMR on count of 0x0000
0 = Counter rolls over to 0xFFFF on count of 0x0000

EN — PIT Enable Bit

The read/write EN bit enables PIT operation. When the PIT is disabled, the counter and prescaler are held in a stopped state.
1 = PIT enabled
0 = PIT disabled

16.5.2.3. PIT Count Register

The 16bit, read-only PIT Control Register (PCNTR) contains the counter value. Reading the 16bit counter with two 8bit reads is not guaranteed to be coherent. Writing to PCNTR has no effect, and write cycles are terminated normally.

Offset Address: 0x0006

	15	14	13	12	11	10	9	8
R	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8
W								
RESET:	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
R	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
W								
RESET:	1	1	1	1	1	1	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 16-4: PIT Count Register (PCNTR)

16.6. Functional Description

This subsection describes the PIT functional operation.

16.6.1. Set-and-Forget Timer Operation

This mode of operation is selected when the RLD bit in the PCSR register is set. When the PIT counter reaches a count of 0x0000, the PIF flag is set in PCSR. The value in the modulus latch is loaded into the counter, and the counter begins decrementing toward 0x0000. If the PIE bit is set in PCSR, the PIF flag issues an interrupt request to the CPU.

When the OVW bit is set in PCSR, the counter can be directly initialized by writing to PMR without having to wait for the count to reach 0x0000.

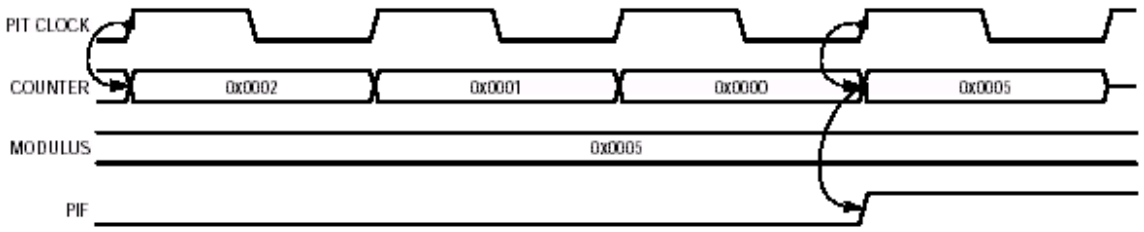


Figure 16-5: Counter Reloading from the Modulus latch

16.6.2. Free-Running Timer Operation

This mode of operation is selected when the RLD bit in PCSR is clear. In this mode, the counter rolls over from 0x0000 to 0xFFFF without reloading from the modulus latch and continues to decrement.

When the counter reaches a count of 0x0000, the PIF flag is set in PCSR. If the PIE bit is set in PCSR, the PIF flag issues an interrupt request to the CPU.

When the OVW bit is set in PCSR, the counter can be directly initialized by writing to PMR without having to wait for the count to reach 0x0000.

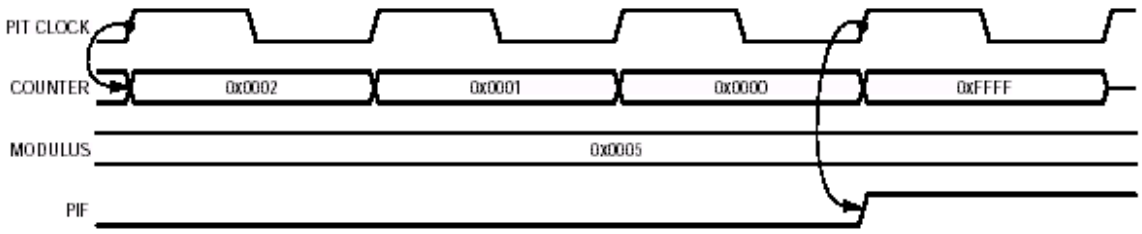


Figure 16-6: Counter in Free-Running Mode

16.6.3. Timeout Specifications

The 16bit PIT counter and prescaler supports different timeout periods. The prescaler divides the system clock as selected by the PRE[3:0] bits in PCSR. The PM[15:0] bits in PMR select the timeout period.

$$\text{timeout period} = \text{PRE}[3:0] \times (\text{PM}[15:0] + 1) \text{ clocks}$$

16.7. Interrupt Operation

Table 16-3 lists the interrupt requests generated by the PIT.

Table 16-3: PIT Interrupt Requests

Interrupt Request	Flag	Enable Bit
Timeout	PIF	PIE

The PIF flag is set when the PIT counter reaches 0x0000. The PIE bit enables the PIF flag to generate interrupt requests. Clear PIF by writing a 1 to it or by writing to the PMR.

17. Watchdog Timer Module (WDT)

17.1. Introduction

The watchdog timer is a 16bit timer used to help software recover from runaway code or give an interrupt when the operation has run longer than expected. The watchdog timer has a free-running down-counter (watchdog counter) that generates a reset or interrupt on underflow. To prevent a reset, software must periodically restart the countdown by servicing the watchdog.

17.2. Modes of Operation

This subsection describes the operation of the watchdog timer in low-power modes and debug mode of operation.

17.2.1. Wait Mode

In wait mode with the WAIT bit set in the Watchdog Control Register (WCR), watchdog timer operation stops. In wait mode with the WAIT bit clear, the watchdog timer continues to operate normally.

17.2.2. Doze Mode

In doze mode with the DOZE bit set in WCR, watchdog timer module operation stops. In doze mode with the DOZE bit clear, the watchdog timer continues to operate normally.

17.2.3. Stop Mode

In stop mode with the STOP bit set in WCR, watchdog operation stops in stop mode. When stop mode is exited, the watchdog operation continues operation from the state it was in prior to entering stop mode. In stop mode with the STOP bit clear, the watchdog timer continues to operate normally.

17.2.4. Debug Mode

In debug mode with the DBG bit set in WCR, watchdog timer module operation stops. In debug mode with the DBG bit clear, the watchdog timer continues to operate normally. When debug mode is exited, watchdog timer operation continues from the state it was in before entering debug mode, but any updates made in debug mode remain.

17.3. 17.3 Block Diagram

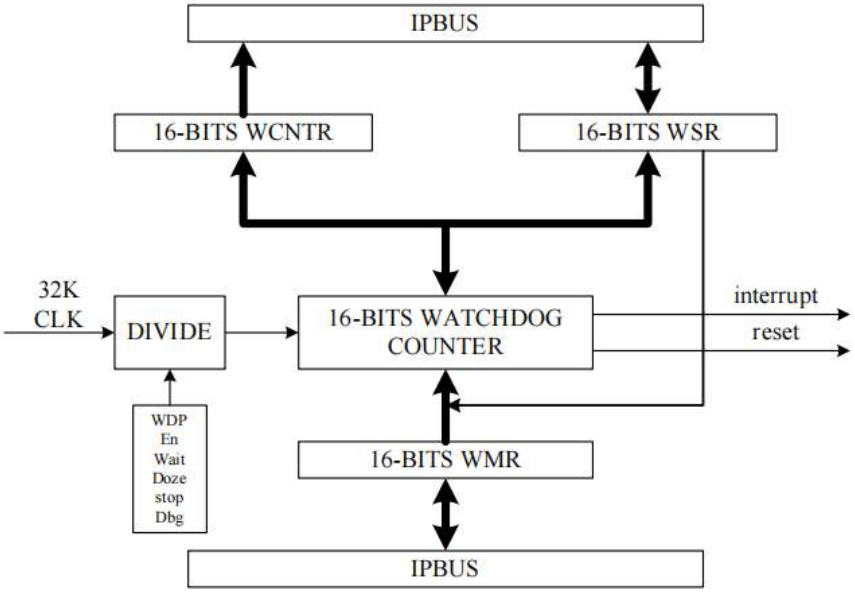


Figure 17-1: Watchdog Timer Block Diagram

17.4. Signals

The watchdog timer module has no off-chip signals.

17.5. Memory Map and Registers

This subsection describes the memory map and registers for the watchdog timer.

17.5.1. Memory Map

Refer Table 17-1 to for an overview of the watchdog memory map.

Table 17-1: Watchdog Timer Module Memory Map

Offset Address	Bits 15-8	Bits 7-0	Access(1)
0x0000	Watchdog Modulus Register (WMR)		S
0x0002	Watchdog Control Register (WCR)		S
0x0004	Watchdog Service Register (WSR)		S/U
0x0006	Watchdog Count Register (WCNTR)		S/U

NOTE: *S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.*

17.5.2. Registers

The watchdog timer programming model consists of these registers:

- The Watchdog Control Register (WCR) configures watchdog timer operation. See 17.5.2.2 Watchdog Control Register.
- The Watchdog Modulus Register (WMR) determines the timer modulus reload value. See 17.5.2.4 Watchdog Count Register.
- The Watchdog Count Register (WCNTR) provides visibility to the watchdog counter value. See 17.5.2.4 Watchdog Count Register.
- The Watchdog Service Register (WSR) requires a service sequence to prevent reset. See The read-only WC[15:0] field reflects the current value in the watchdog counter. Reading the 16bit WCNTR with two 8bit reads is not guaranteed to return a coherent value. Writing to WCNTR has no effect, and write cycles are terminated normally. This register is for watchdog work domain, so the read value maybe not stabilization, please read it time after time continuously.

17.5.2.1. Watchdog Modules Register

Offset Address: 0x0000

	15	14	13	12	11	10	9	8
R W	WM15	WM14	WM13	WM12	WM11	WM10	WM9	WM8
RESET:	0	0	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
R W	WM7	WM6	WM5	WM4	WM3	WM2	WM1	WM0
RESET:	1	1	1	1	1	1	1	1

Figure 17-2: Watchdog Modulus Register (WMR)

WM[15:0] — Watchdog Modulus Field

WM[15:0] field contains the modulus that is reloaded into the watchdog counter by a service sequence. Writing to WMR immediately loads the new modulus value into the watchdog counter. The new value is also used at the next and all subsequent reloads.

Reading WMR returns the value in the modulus register. Reset initializes the WM[15:0] field to 0x3FFF.

17.5.2.2. Watchdog Control Register

The 16bit read/write Watchdog Control Register (WCR) configures watchdog timer operation.

Offset Address: 0x0002

	15	14	13	12	11	10	9	8
R	0	0	0	0	WAIT	DOZE	STOP	DBG
W								
RESET:	0	0	0	0	1	1	1	1

	7	6	5	4	3	2	1	0
R	IS	WDP[2:0]			IF	IE	0	EN
W							CU	
RESET:	0	1	1	1	0	1	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 17-3: Watchdog Control Register (WCR)

WAIT — Wait Mode Bit

WAIT bit controls the function of the watchdog timer in wait mode. Reset sets WAIT.

1 = Watchdog timer stopped in wait mode

0 = Watchdog timer not affected in wait mode

DOZE — Doze Mode Bit

DOZE bit controls the function of the watchdog timer in doze mode. Reset sets DOZE.

1 = Watchdog timer stopped in doze mode

0 = Watchdog timer not affected in doze mode

STOP — STOP Mode Bit

STOP bit controls the function of the watchdog timer in stop mode. Reset sets STOP.

1 = Watchdog timer stopped in stop mode

0 = Watchdog timer not affected in stop mode

DBG — Debug Mode Bit

DBG bit controls the function of the watchdog timer in debug mode. During debug mode, watchdog timer registers can be written and read normally. When debug mode is exited, timer operation continues from the state it was in before entering debug mode, but any updates made in debug mode remain.

1 = Watchdog timer stopped in debug mode

0 = Watchdog timer not affected in debug mode

NOTE: *Changing the DBG bit from 1 to 0 during debug mode starts the watchdog timer. Changing the DBG bit from 0 to 1 during debug mode stops the watchdog timer.*

EN — Watchdog Enable Bit

EN bit enables the watchdog timer.

1 = Watchdog timer enabled

0 = Watchdog timer disabled

CU — Watchdog Change Update Bit

Write One to CU bit update the WDP[2:0] and WMR to the work latch.

IE — Watchdog Interrupt Enable Bit

IE bit enables the watchdog timer interrupt mode. Once interrupt generated and the EN bit is 1, this bit will be auto clear.

1 = Watchdog timer interrupt mode enabled

0 = Watchdog timer interrupt mode disabled

IF — Watchdog Interrupt Flag Bit

Write One to this bit will clear the flag.

IS — Watchdog Clock Domain Interrupt Status Bit

This bit is read-only, if this bit is 1'b1, the status of watchdog clock domain is not cleared, so if CPU want to sleep or stop, it will be wakeup again. So, before CPU want to sleep or stop, please check this bit first.

WDP[2:0] — Watchdog Timer Prescaler

The WDP[2:0] bits determine the watchdog timer prescaling when the watchdog timer is running. The different prescaling values and their corresponding
time-out periods are shown in table 11-2 Watchdog Timer Prescaler.

Table 17-2: Watchdog Timer Prescaler

WDP[2:0]	Prescaler
000	128KHz/2048
001	128KHz/1024
010	128KHz/512
011	128KHz/256
100	128KHz/128
101	128KHz/64
110	128KHz/32
111	128KHz/16

17.5.2.3. Watchdog Service Register

When the watchdog timer is enabled, writing 0x5555 and then 0xAAAA to the Watchdog Service Register (WSR) before the watchdog counter times out prevents a reset. If WSR is not serviced before the timeout, the watchdog timer sends a signal to the reset controller or interrupt controller module and asserts a system reset or interrupt.

Both writes must occur in the order listed before the timeout, but any number of instructions can be executed between the two writes. However, writing any value other than 0x5555 or 0xAAAA to WSR resets the servicing sequence, requiring both values to be written to keep the watchdog timer from causing a reset.

Offset Address: 0x0004

	15	14	13	12	11	10	9	8
R W	WS15	WS14	WS13	WS12	WS11	WS10	WS9	WS8
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	WS7	WS6	WS5	WS4	WS3	WS2	WS1	WS0
RESET:	0	0	0	0	0	0	0	0

Figure 17-4: Watchdog Service Register (WSR)

17.5.2.4. Watchdog Count Register

Offset Address: 0x0006

	15	14	13	12	11	10	9	8
R	WC15	WC14	WC13	WC12	WC11	WC10	WC9	WC8
W								
RESET:	0	0	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
R	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0
W								
RESET:	1	1	1	1	1	1	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 17-5: Watchdog Count Register (WCNTR)

WC[15:0] — Watchdog Count Field

The read-only WC[15:0] field reflects the current value in the watchdog counter. Reading the 16bit WCNTR with two 8bit reads is not guaranteed to return a coherent value. Writing to WCNTR has no effect, and write cycles are terminated normally. This register is for watchdog work domain, so the read value maybe not stabilization, please read it time after time continuously.

18. Real Time Controller (RTC)

18.1. Introduction

This module is a controller which controls a comprehensive PMU with RTC function. It can reconfigure RTC counter, setting alarms and generating time/alarm interrupts.

18.2. Features

The main features of the module:

- Reconfigure RTC counter and read from seconds, minutes, hours and days counters
- Support alarm settings
- Interrupt sources: second, minute, hour, day interrupts, programmable alarm interrupts, 1KHz/32KHz periodic interrupts.

18.3. Test Mode

In test mode, we can config RTC by CPU or by chip pins, and we can directly check the RTC status and clk signals by chip pins.

18.4. Block Diagram

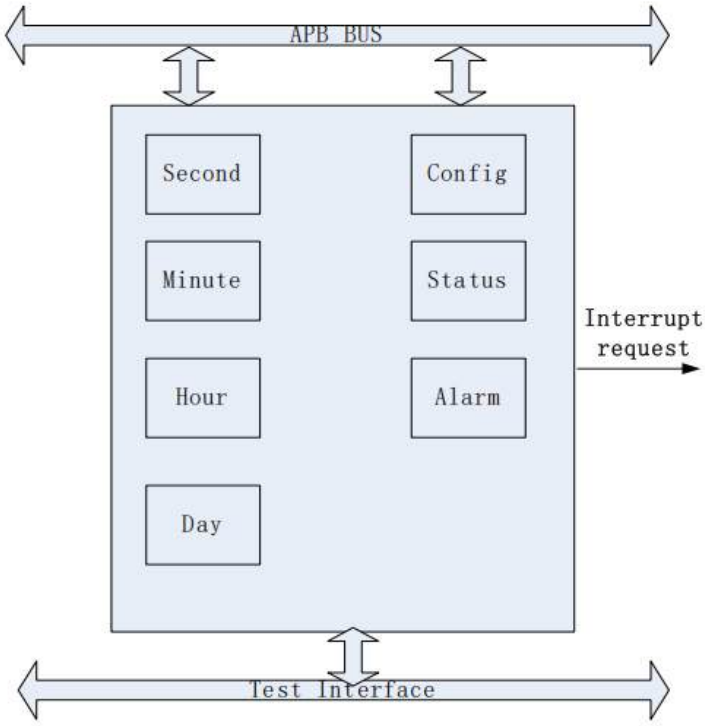


Figure 18-1: RTC Block Diagram

19. Edge Port Module (EPORT)

19.1. Introduction

The edge port module (EPORT) has eight external interrupt pins. Each pin can be configured individually as a low level-sensitive interrupt pin, an edge-detecting interrupt pin (rising edge, falling edge, or both), or a general-purpose input/output/ (I/O) pin. See Figure 21-1.

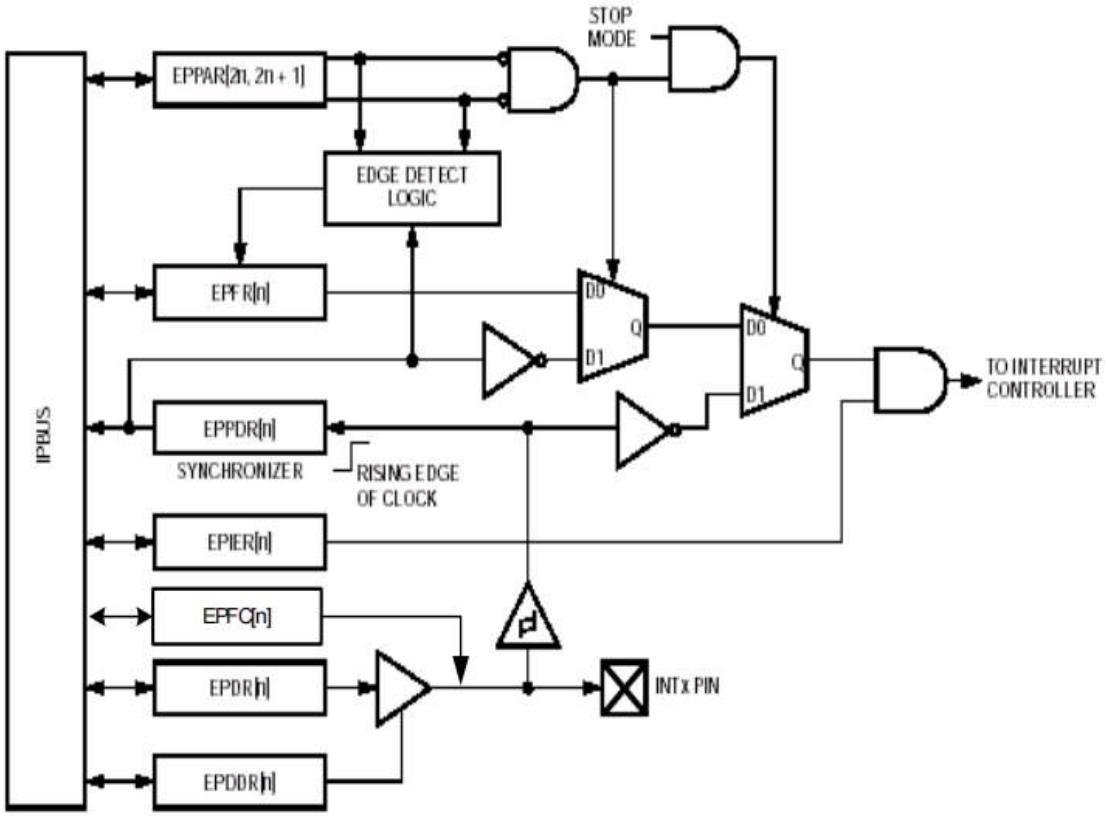


Figure 19-1: EPORT Block Diagram

19.2. Low-Power Mode Operation

This subsection describes the operation of the EPORT module in low-power modes.

19.2.1. Wait and Doze Modes

In wait and doze modes, the EPORT module continues to operate normally and may be configured to exit the low-power modes by generating an interrupt request on either a selected edge or a low level on an external pin.

19.2.2. Stop Mode

In stop mode, there are no clocks available to perform the edge-detect function. Only the level-detect logic is active (if configured) to allow any low level on the external interrupt pin to generate an interrupt (if enabled) to exit stop mode.

NOTE: *The input pin synchronizer is bypassed for the level-detect logic since no clocks are available.*

19.3. Interrupt/General-Purpose I/O Pin Descriptions

All pins default to general-purpose input pins at reset. The pin value is synchronized to the rising edge of CLKOUT when read from the EPORT Pin Data Register (EPPDR). The values used in the edge/level detect logic are also synchronized to the rising edge of CLKOUT. These pins use Schmitt triggered input buffers which have built in hysteresis designed to decrease the probability of generating false edge-triggered interrupts for slow rising and falling input signals.

19.4. 19.4 Memory Map and Registers

This subsection describes the memory map and register structure.

19.4.1. Memory Map

Refer to Table 21-1 for a description of the EPORT memory map.

Table 19-1: Module Memory Map

Offset Address	Bits 15-8	Bits 7-0	Access(1)
0x0000	EPORT Data Direction Register (EPDDR)	EPORT Interrupt Enable Register (EPIER)	S
0x0002	EPORT Pin Assignment Register (EPPAR)		S
0x0004	EPORT Flag Register (EPFR)	EPORT Pin Pull-up enable Register (EPPUE)	S/U
0x0006	EPORT Data Register (EPDR)	EPORT Pin Data Register (EPPDR)	S/U
0x0008	EPORT Digital Filter Control Register(EPFC)	EPORT Bit Set Register (EPBSR)	S
0x000A	EPORT Level Polarity Register (EPLPR)	EPORT Open Drain Register (EPODE)	S
0x000C	Reversed		S
0x000E	EPORT Bit Clear Register (EPBCR)		S

NOTE: *S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.*

19.4.2. Registers

The EPORT programming model consists of these registers:

- The EPORT Pin Assignment Register (EPPAR) controls the function of each pin individually.
- The EPORT Data Direction Register (EPDDR) controls the direction of each one of the pins individually.
- The EPORT Interrupt Enable Register (EPIER) enables interrupt requests for each pin individually.
- The EPORT Data Register (EPDR) holds the data to be driven to the pins.
- The EPORT Pin Data Register (EPPDR) reflects the current state of the pins.

- The EPORT Flag Register (EPFR) individually latches EPORT edge events.
- The EPORT Pin Pull-up Enable Register (EPPUE) controls the pull-up of each one of the pins individually.
- The EPORT Level Polarity Register (EPLPR) controls the level polarity of each one of the pins for level-sensitive.
- The EPORT Open Drain Enable Register (EPODE) controls the Open Drain of each one of the pins for output individually.
- The EPORT Digital Filter Control Register(EPFC) enable the filter and control the width of input pulse will be filtered.

19.4.2.1. Edge Port Interrupt Enable Register

Offset Address: 0x0000

	7	6	5	4	3	2	1	0
R W	EPIE7	EPIE6	EPIE5	EPIE4	EPIE3	EPIE2	EPIE1	EPIE0
RESET:	0	0	0	0	0	0	0	0

Figure 19-2: EPORT Port Interrupt Enable Register (EPIER)

EPIE[7:0] — Edge Port Interrupt Enable Bits

The read/write EPIE[7:0] bits enable EPORT interrupt requests. If a bit in EPIER is set, EPORT generates an interrupt request when:

- The corresponding bit in the EPORT Flag Register (EPFR) is set or later becomes set, or
- The corresponding pin level is low and the pin is configured for level-sensitive operation

Clearing a bit in EPIER negates any interrupt request from the corresponding EPORT pin. Reset clears EPIE[7:0].

1 = Interrupt requests from corresponding EPORT pin enabled
0 = Interrupt requests from corresponding EPORT pin disabled

19.4.2.2. EPORT Data Direction Register

Offset Address: 0x0001

	7	6	5	4	3	2	1	0
R W	EPDD7	EPDD6	EPDD5	EPDD4	EPDD3	EPDD2	EPDD1	EPDD0
RESET:	0	0	0	0	0	0	0	0

Figure 19-3: EPORT Data Direction Register (EPDDR)

EPDD[7:0] — Edge Port Data Direction Bits

Setting any bit in the EPDDR configures the corresponding pin as an output. Clearing any bit in EPDDR configures the corresponding pin as an input. Pin direction is independent of the level/edge detection configuration. Reset clears EPDD[7:0].

To use an EPORT pin as an external interrupt request source, its corresponding bit in EPDDR must be clear. Software can generate interrupt requests by programming the EPORT Data Register when the EPDDR selects output.

1 = Corresponding EPORT pin configured as output

0 = Corresponding EPORT pin configured as input

19.4.2.3. EPORT Pin Assignment Register

Offset Address: 0x0002 ~ 0x0003

	15	14	13	12	11	10	9	8
R W	EPPA7		EPPA6		EPPA5		EPPA4	
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	EPPA3		EPPA2		EPPA1		EPPA0	
RESET:	0	0	0	0	0	0	0	0

Figure 19-4: EPORT Pin Assignment Register (EPPAR)

EPPA[7:0] — EPORT Pin Assignment Select Fields

The read/write EPPAx fields configure EPORT pins for level detection and rising and/or falling edge detection as shows.

Pins configured as level-sensitive are inverted so that a logic 0 or logic 1 on the external pin represents a valid interrupt request. Level-sensitive interrupt inputs are not latched. To guarantee that a level-sensitive interrupt request is acknowledged, the interrupt source must keep the signal asserted until acknowledged by software. Level sensitivity must be selected to bring the device out of stop mode with an INTx interrupt.

Pins configured as edge-triggered are latched and need not remain asserted for interrupt generation. A pin configured for edge detection is monitored regardless of its configuration as input or output.

Table 19-2: EPPAx Field Settings

EPPAx	Pin Configuration
00	Pin INTx level-sensitive
01	Pin INTx rising edge triggered
10	Pin INTx falling edge triggered
11	Pin INTx both falling edge and rising edge triggered

Interrupt requests generated in the EPORT module can be masked by the interrupt controller module. EPPAR functionality is independent of the selected pin direction.

Reset clears the EPPAx fields.

19.4.2.4. EPORT Pin Pull-up enable Register

Offset Address: 0x0004

	7	6	5	4	3	2	1	0
R W	EPPUE7	EPPUE6	EPPUE5	EPPUE4	EPPUE3	EPPUE2	EPPUE1	EPPUE0
RESET:	1	1	1	1	1	1	1	1

Figure 19-5: EPORT Pin Pull-up Enable Register (EPPUE)

CAUTION: EPPUE is not valid in this device

EPPUE[7:0] — Edge Port Pin Pull-up enable Bits

Setting any bit in the EPPUE configures the corresponding pin to enable pull-up. Clearing any bit in EPPUE configures the corresponding pin disable pull-up. Reset sets EPPUE[7:0].

1 = Corresponding EPORT pin configured to enable pull-up

0 = Corresponding EPORT pin configured to disable pull-up

19.4.2.5. Edge Port Flag Register

Offset Address: 0x0005

	7	6	5	4	3	2	1	0
R	EPFR7	EPFR6	EPFR5	EPFR4	EPFR3	EPFR2	EPFR1	EPFR0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

Figure 19-6: EPORT Port Flag Register (EPFR)

EPF[7:0] — Edge Port Flag Bits

When an EPORT pin is configured for edge triggering, its corresponding read/write bit in EPFR indicates that the selected edge has been detected. Reset clears EPF[7:0].

1 = Selected edge for INTx pin has been detected.

0 = Selected edge for INTx pin has not been detected.

Bits in this register are set when the selected edge is detected on the corresponding pin. A bit remains set until cleared by writing a 1 to it. Writing 0 has no effect. If a pin is configured as level-sensitive (EPPARx = 00), pin transitions do not affect this register.

19.4.2.6. Edge Port Pin Data Register

Offset Address: 0x0006

	7	6	5	4	3	2	1	0
R	EPPD7	EPPD6	EPPD5	EPPD4	EPPD3	EPPD2	EPPD1	EPPD0
W								
RESET:	P	P	P	P	P	P	P	P

= Writes have no effect and the access terminates without a transfer error excepiton.
P = Current pin state

Figure 19-7: EPORT Port Pin Data Register (EPPDR)

EPPD[7:0] — Edge Port Pin Data Bits

The read-only EPPDR reflects the current state of the EPORT pins. Writing to EPPDR has no effect, and the write cycle terminates normally. Reset does not affect EPPDR.

19.4.2.7. Edge Port Data Register

Offset Address: 0x0007

	7	6	5	4	3	2	1	0
R W	EPD7	EPD6	EPD5	EPD4	EPD3	EPD2	EPD1	EPD0
RESET:	1	1	1	1	1	1	1	1

Figure 19-8: EPORT Port Data Register (EPDR)

EPD[7:0] — Edge Port Data Bits

Data written to EPDR is stored in an internal register; if any pin of the port is configured as an output, the bit stored for that pin is driven onto the pin. Reading EDPR returns the data stored in the register. Reset sets EPD[7:0].

19.4.2.8. EPORT Port Bit Set Register

Offset Address: 0x0008

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	EPBSR7	EPBSR6	EPBSR5	EPBSR4	EPBSR3	EPBSR2	EPBSR1	EPBSR0
RESET:	0	0	0	0	0	0	0	0

Figure 19-9: EPORT Port Bit Set Register (EPBSR)

EPBSR[7:0] — EPORT Port Bit Set Register

1 = The corresponding bit of EPDR will be set;

0 = The corresponding bit of EPDR will not be affected;

19.4.2.9. EPORT Digital Filter Control Register

Offset Address: 0x0009

	7	6	5	4	3	2	1	0
R W	Filter_Width[6:0]							Filter_En
RESET:	0	0	0	0	0	0	0	0

Figure 19-10: EPORT Digital Filter Control Register (EPFC)

Filter_En — EPORT Digital Filter Enable Bit

1 = EPORT digital filter enable;
0 = EPORT digital filter disable;

Filter_Width[6:0] — EPORT Digital Filter Pulse Width Select Bit

Filter_Width[6:0] determine the width of input pulse will be filtered. If the input pulse width less than (Filter_Width[6:0]+2), the pulse will be filtered.

19.4.2.10. EPORT Open Drain enable Register

Offset Address: 0x000A

	7	6	5	4	3	2	1	0
R W	EPODE7	EPODE6	EPODE5	EPODE4	EPODE3	EPODE2	EPODE1	EPODE0
RESET:	0	0	0	0	0	0	0	0

Figure 19-11: EPORT Open Drain Enable Register (EPODE)

CAUTION: EPODE is not valid in this device

EPODE[7:0] — Edge Port Open Drain enable Bits

If EPORT are configured to output, setting any bit in the EPODE configures the corresponding pin to Open Drain output. Clearing any bit in EPODE configures the corresponding pin CMOS output. Reset clears EPODE[7:0].

1 = Corresponding EPORT pin configured to Open Drain output

0 = Corresponding EPORT pin configured to CMOS output

19.4.2.11. EPORT Level Polarity Register

Offset Address: 0x000B

	7	6	5	4	3	2	1	0
R W	EPLPR7	EPLPR6	EPLPR5	EPLPR4	EPLPR3	EPLPR2	EPLPR1	EPLPR0
RESET:	0	0	0	0	0	0	0	0

Figure 19-12: EPORT Level Polarity Register (EPLPR)

EPLPR[7:0] — Edge Port Level Polarity Bits

If EPORT are configured to level-sensitive, setting any bit in the EPLPR configures the corresponding pin high level-sensitive. Clearing any bit in EPLPR configures the corresponding pin low level-sensitive. Reset clears EPLPR[7:0].

1 = Corresponding EPORT pin configured to high level-sensitive

0 = Corresponding EPORT pin configured to low level-sensitive

19.4.2.12. EPORT Port Bit Clear Register

Offset Address: 0x000F

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	EPBCR7	EPBCR6	EPBCR5	EPBCR4	EPBCR3	EPBCR2	EPBCR1	EPBCR0
RESET:	0	0	0	0	0	0	0	0

Figure 19-13: EPORT Port Bit Clear Register (EPBCR)

EPBCR[7:0] — EPORT Port Bit Clear Register

1 = The corresponding bit of EPDR will be clear;

0 = The corresponding bit of EPDR will not be affected;

20. CANBus Controller (CANBC)

20.1. Introduction

The CANBus module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. A general block diagram is shown in Figure 20-1, which describes the main sub-blocks implemented in the CANBus module, including two embedded memories, one for storing Message Buffers (MB) and another one for storing Rx Individual Mask Registers. Support for up to 64 Message Buffers is provided. The functions of the submodules are described in subsequent sections.

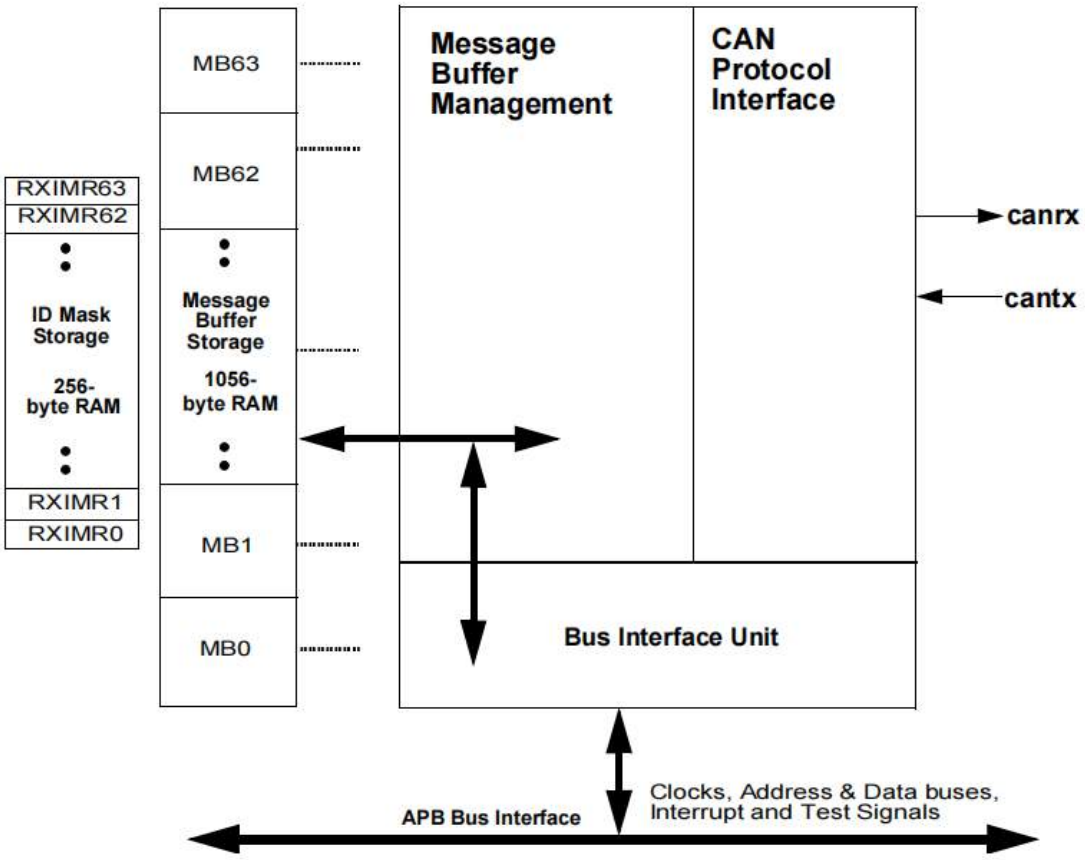


Figure 20-1: CANBus Block Diagram

20.1.1. Overview

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The CANBus module is a full implementation of the CAN protocol specification, version 2.0 B, which supports both standard and extended message frames.

Number of 16 Message Buffers is supported. The Message Buffers are stored in an embedded SRAM dedicated to the CANBus module.

The CAN Protocol Interface (CPI) submodule manages the serial communication on the CAN bus, requesting SRAM access for receiving and transmitting message frames, validating received messages and performing error handling. The Message Buffer Management (MBM) submodule handles Message Buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms. The Bus Interface Unit (BIU) submodule controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the Bus Interface Unit.

20.1.2. CANBus Module Features

The CANBus module includes these distinctive features:

- Full implementation of the CAN protocol specification, version 2.0B
  - Standard data and remote frames
  - Extended data and remote frames
  - 0–8 bytes data length
  - Programmable bit rate up to 1 Mbit/s
  - Content-related addressing
- 16 Message Buffers of zero to eight bytes data length
- Each MB configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Message Buffer
- Includes either 288 bytes (16 MBs) of SRAM used for MB storage
- Includes either 64 bytes (16 MBs) of SRAM used for individual Rx Mask Registers
- Full featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 8 extended, 16 standard or 32 partial (8bits) IDs, with individual masking capability
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused MB and Rx Mask Register space can be used as general purpose SRAM space
- Listen-only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power mode
- Hardware cancellation on Tx message buffers

20.1.3. Modes of Operation

The CANBus module has four functional modes: Normal Mode (User and Supervisor), Freeze Mode, Listen-Only Mode and Loop-Back Mode. There is also a low power mode: Disable Mode.

- Normal Mode (User or Supervisor)

  In Normal Mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze Mode

  It is enabled when the FRZ bit in the MCR is asserted. If enabled, Freeze Mode is entered when the HALT bit in MCR is set or when Debug Mode is requested at MCU level. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See 20.4.10.1, "Freeze Mode" for more information.

- Listen-Only Mode

  The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If CANBus detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back Mode

  The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, CANBus performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). CANBus behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, CANBus ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Module Disable Mode

  This Low Power Mode is entered when the MDIS bit in the MCR is asserted. When disabled, the module shuts down the clocks to the CAN Protocol Interface and Message Buffer Management submodules. Exit from this mode is done by negating the MDIS bit in the MCR. See 20.4.10.2,"Module Disable Mode" for more information.

20.2. External Signal Description

20.2.1. Overview

The CANBus module has two I/O signals connected to the external MCU pins. These signals are summarized in Table 20-1 and described in more detail in the next subsections.

Table 20-1: CANBus Signals

Signal name	Direction	Description
CANRX	Input	CAN receive pin
CANTX	Output	CAN transmit pin

20.2.2. Signal Descriptions

20.2.2.1. CANRX

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'.Recessive state is represented by logic level '1'.

20.2.2.2. CANTX

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'.Recessive state is represented by logic level '1'.

21. Serial Communication Interface Module (SCI)

21.1. Introduction

The Serial Communication Interface Module (SCI) supports basic UART and allows asynchronous serial communications with peripheral devices and other microcontroller units (MCU).This module also supports LIN slave operation.

21.2. Features

Features of each SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13bit modulo divider) with configurable oversampling ratio from 4x to 256x
- Interrupt, polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
  - Receive data match
- Hardware parity generation and checking
- Programmable 8bit, 9bit or 10bit character length
- Programmable 1bit or 2bit stop bits
- Three receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
  - Receive data match
- Automatic address matching to reduce ISR overhead:
  - Address mark matching
  - Idle line address matching
  - Address match start, address match end
- Optional 13bit break character generation / 11bit break character detection
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64 or 128 idle characters
- Selectable transmitter output and receiver input polarity
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse width
- Independent FIFO structure for transmit and receive

– Separate configurable watermark for receive and transmit requests

– Option for receiver to assert request after a configurable number of idle characters if receive FIFO is not empty

21.3. Modes of Operation

- Stop mode
- Wait mode

21.4. Block Diagram

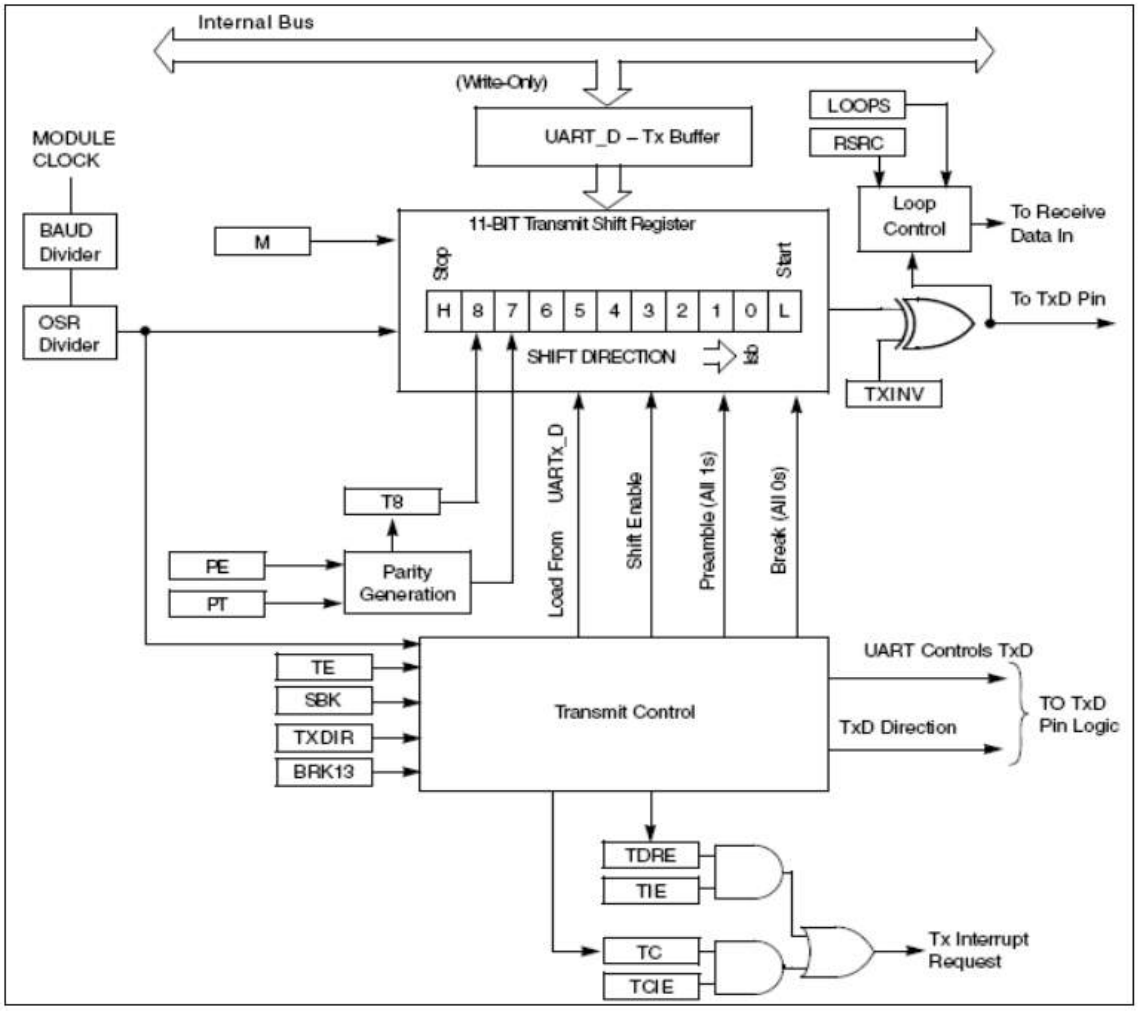


Figure 21-1: SCI transmitter Block Diagram

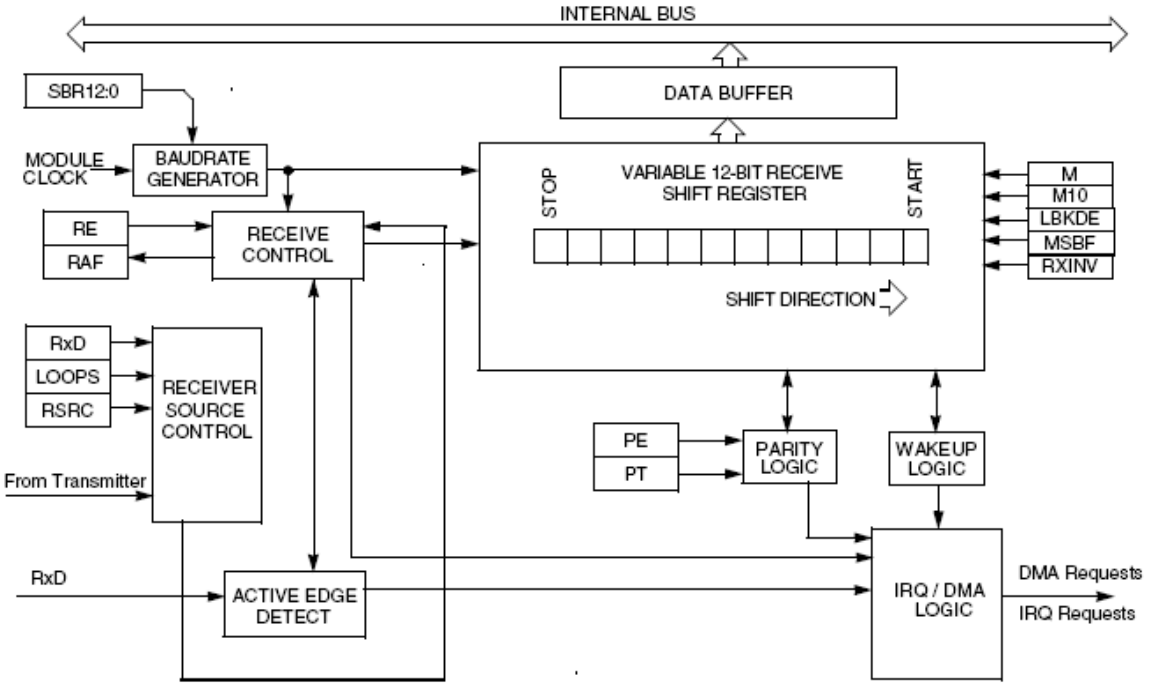


Figure 21-2: SCI Receiver Block Diagram

21.5. Modes of Operation

21.5.1. Stop Mode

The SCI will not be functional during Stop mode.

21.5.2. Wait Mode

The SCI can be configured to Stop in Wait modes, when the DOZEEN bit is set. The transmitter and receiver will finish transmitting/receiving the current word.

21.6. Signal Description

Table 21-1 gives an overview of the signals which are described here.

Table 21-1: Signal Properties

Signal	Description	I/O
TXD	Transmit data. This pin is normally an output, but is an input (tristate) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data.	I/O
RXD	Receive data	I
SCI_DE	RS-485 transceiver's control signal	O

21.7. Memory Map and Registers

Table 21-2 shows the SCI memory map.

Table 21-2: SCI Module Memory Map[1]

Offset Address	Register
0x0000	SCI Version ID (SCI_VERID)
0x0004	SCI Parameter (SCI_PARAM)
0x0008	SCI Reset (SCI_RESET)
0x000C	SCI Pin (SCI_PIN)
0x0010	SCI Baud Rate Register (SCI_BAUD)
0x0014	SCI Status Register (SCI_STAT)
0x0018	SCI Control Register (SCI_CTRL)
0x001C	SCI Data Register (SCI_DATA)
0x0020	SCI Match Address Register (SCI_MATCH)
0x0024	SCI Modem IrDA Register (SCI_MODIR)
0x0028	SCI FIFO Register (SCI_FIFO)
0x002C	SCI Watermark Register (SCI_WATER)
0x0030	SCI Oversampling Ratio Register(SCI_OSR)

NOTE: *Each module is assigned 16 Kbytes of address space, all of which may not be decoded. Accesses outside of the specified module memory map generate a bus error exception.*

21.7.1. SCI Version ID Register

Offset Address: 0x0000

	31	30	29	28	27	26	25	24
R	VERID							
W								
RESET	0	0	0	0	0	1	0	0
	23	22	21	20	19	18	17	16
R	VERID							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	VERID							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	VERID							
W								
RESET:	0	0	0	0	0	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-3: SCI Version ID Register(SCI_VERID)

VERSION_ID[31:0] — SCI Version ID

21.7.2. SCI Parameter Register

Offset Address: 0x0004

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	Reserved				RXFIFO_SZ			
W								
RESET	0	0	0	0	0	0	1	1
	7	6	5	4	3	2	1	0
R	Reserved				TXFIFO_SZ			
W								
RESET:	0	0	0	0	0	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-4: SCI Parameter Register(SCI_PARAM)

RXFIFO_SZ — The receive buffer/FIFO size

This read-only control bit indicates the receive buffer/FIFO size.

TXFIFO_SZ — The transmit buffer/FIFO size

This read-only control bit indicates the transmit buffer/FIFO size.

21.7.3. SCI Reset Register

Offset Address: 0x0008

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved						SWRST	Reserved
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-5: SCI Reset Register(SCI_RESET)

SWRST — Software reset

This read/write bit allows the software to reset SCI IP.

1 = Software reset is asserted

0 = No software reset is asserted

21.7.4. SCI Pin Register

Offset Address: 0x000C

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved						PINCFG	
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-6: SCI Pin Register(SCI_PIN)

PINCFG — Useless in this module

21.7.5. SCI Baud Rate Register

Offset Address: 0x0010

	31	30	29	28	27	26	25	24
R	MAEN1	MAEN2	M10	Reserved				
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	TDMAE	Reserved	RDMAE	Reserved	MATCFG		BOTHEDGE	RESYNC-DIS
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	LBKDIE	RXEDGIE	SBNS	SBR				
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	SBR							
W								
RESET:	0	0	0	0	0	1	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-7: SCI Baud Rate Register (SCI_BAUD)

MAEN1 — Match Address Mode Enable 1

1 = Enables automatic address matching or data matching mode for MATCH[MA1]

0 = Normal operation

MAEN2 — Match Address Mode Enable 1

1 = Enables automatic address matching or data matching mode for MATCH[MA2]

0 = Normal operation

M10 — 10bit Mode select

The M10 bit causes a tenth bit to be part of the serial transmission. This bit should only be changed when the transmitter and receiver are both disabled.

1 = Receiver and transmitter use 10bit data characters

0 = Receiver and transmitter use 8bit or 9bit data characters

TDMAE — Transmitter DMA Enable

TDMAE configures the transmit data register empty flag, LPUART_STAT[TDRE], to generate a DMA request.

1 = DMA request enabled

0 = DMA request disabled

RDMAE — Receiver Full DMA Enable

RDMAE configures the receiver data register full flag, LPUART_STAT[RDRF], to generate a DMA request.

1 = DMA request enabled

0 = DMA request disabled

MATCFG — Match Configuration

Configures the match addressing mode used:

00-- Address Match Wakeup;

01-- Idle Match Wakeup;

10-- Match On and Match Off;

11-- Enables RWU on Data Match and Match On/Off for transmitter CTS input

BOTHEDGE — Both Edge Sampling

Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given oversampling ratio. This bit must be set for oversampling ratios between x4 and x7 and is optional for higher oversampling ratios. This bit should only be changed when the receiver is disabled.

1 = Receiver samples input data using the rising and falling edge of the baud rate clock.

0 = Receiver samples input data using the rising edge of the baud rate clock.

RESYNCDIS — Resynchronization Disable

When set, disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should only be changed when the receiver is disabled.

1 = Resynchronization during received data word is disabled

0 = Resynchronization during received data word is supported

LBKDIE — LIN Break Detect Interrupt Enable

LBKDIE enables the LIN break detect flag, LBKDIF, to generate interrupt requests.

1 = Hardware interrupt requested when SCI_STAT[LBKDIF] flag is 1

0 = Hardware interrupts from SCI_STAT[LBKDIF] disabled (use polling)

RXEDGIE — RX Input Active Edge Interrupt Enable

Enables the receive input active edge, RXEDGIF, to generate interrupt requests. Changing CTRL[LOOP] or CTRL[RSRC] when RXEDGIE is set can cause the RXEDGIF to set.

1 = Hardware interrupt requested when SCI_STAT[RXEDGIF] flag is 1

0 = Hardware interrupts from SCI_STAT[RXEDGIF] disabled (use polling)

SBNS — Stop Bit Number Select

SBNS determines whether data characters are one or two stop bits. This bit should only be changed when the transmitter and receiver are both disabled.

1 = Two stop bits

0 = One stop bits

SBR — Baud Rate Modulo Divisor

The 13 bits in SBR[12:0] set the modulo divide rate for the baud rate generator. When SBR is 1 - 8191, the baud rate equals "baud clock / ((OSR+1) × SBR)". The 13bit baud rate setting [SBR12:SBR0] must only be updated when the transmitter and receiver are both disabled (SCI_CTRL[RE] and SCI_CTRL[TE] are both 0).

21.7.6. SCI Status Register

Offset Address: 0x0014

	31	30	29	28	27	26	25	24
R	LBKDIF	RXEDGIF	MSBF	RXINV	RWUID	BRK13	LBKDE	RAF
W	w1c	w1c						
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W				w1c	w1c	w1c	w1c	w1c
RESET	1	1	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	MA1F	MA2F	Reserved					
W	w1c	w1c						
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-8: SCI Status Register (SCI_STAT)

LBKDIF — LIN Break Detect Interrupt Flag

LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it.

1 = LIN break character has been detected

0 = No LIN break character has been detected

RXEDGIF — RXD Pin Active Edge Interrupt Flag

RXEDGIF is set when an active edge, falling if RXINV = 0, rising if RXINV=1, on the RXD pin occurs. RXEDGIF is cleared by writing a 1 to it.

1 = An active edge on the receive pin has occurred

0 = No active edge on the receive pin has occurred

MSBF — MSB First

Setting this bit reverses the order of the bits that are transmitted and received on the wire. This bit does not affect the polarity of the bits, the location of the parity bit or the location of the start or stop bits. This bit should only be changed when the transmitter and receiver are both disabled.

1 = MSB (bit9, bit8, bit7 or bit6) is the first bit that is transmitted following the start bit depending on the setting of CTRL[M], CTRL[PE] and BAUD[M10]. Further, the first bit received after the start bit is identified as bit9, bit8, bit7 or bit6 depending on the setting of CTRL[M] and CTRL[PE].

0 = LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0

RXINV — Receive Data Inversion

Setting this bit reverses the polarity of the received data input. Setting RXINV inverts the RXD input for all cases: data bits, start and stop bits, break, and idle.

1 = Receive data inverted

0 = Receive data not inverted

RWUID — Receive Wake Up Idle Detect

For RWU on idle character, RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. For address match wakeup, RWUID controls if the IDLE bit is set when the address does not match. This bit should only be changed when the receiver is disabled.

1 = During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. During address match wakeup, the IDLE bit does get set when an address does not match.

0 = During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. During address match wakeup, the IDLE bit does not get set when an address does not match.

BRK13 — Break Character Generation Length

BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. This bit should only be changed when the transmitter is disabled.

1 = Break character is transmitted with length of 13 bit times (if M = 0, SBNS = 0) or 14 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 15 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 16 (if M10 = 1, SNBS = 1).

0 = Break character is transmitted with length of 10 bit times (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1)

LBKDE — LIN Break Detection Enable

LBKDE selects a longer break character detection length. While LBKDE is set, receive data is not stored in the receive data buffer.

1 = Break character is detected at length of 11 bit times (if M = 0, SBNS = 0) or 12 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 14 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 15 (if M10 = 1, SNBS = 1)

0 = Break character is detected at length 10 bit times (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1)

RAF — Receiver Active Flag

RAF is set when the receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line.

1 = SCI receiver active (RXD input not idle)

0 = SCI receiver idle waiting for a start bit

TDRE — Transmit Data Register Empty Flag

When the transmit FIFO is enabled, TDRE will set when the number of data words in the transmit FIFO(SCI_DATA) is equal to or less than the number indicated by SCI_WATER[TXWATER]). To clear TDRE, write to the SCI data register (SCI_DATA) until the number of words in the transmit FIFO is greater than the number indicated by SCI_WATER[TXWATER]. When the transmit FIFO is disabled, TDRE will set when the transmit data register (SCI_DATA) is empty. To clear TDRE, write to the SCI data register (SCI_DATA).

TDRE is not affected by a character that is in the process of being transmitted, it is updated at the start of each transmitted character.

1 = Transmit data buffer empty

0 = Transmit data buffer full

TC — Transmission Complete Flag

TC is cleared when there is a transmission in progress or when a preamble or break character is loaded. TC is set when the transmit buffer is empty and no data, preamble, or break character is being transmitted. When TC is set, the transmit data output signal becomes idle (logic 1). TC is cleared by writing to SCI_DATA to transmit new data, queuing a preamble by clearing and then setting SCI_CTRL[TE], queuing a break character by writing 1 to SCI_CTRL[SBK].

1 = Transmitter idle (transmission activity complete)

0 = Transmitter active (sending data, a preamble, or a break)

RDRF — Receive Data Register Full Flag

When the receive FIFO is enabled, RDRF is set when the number of data words in the receive buffer is greater than the number indicated by SCI_WATER[RXWATER]. To clear RDRF, read SCI_DATA until the number of data words in the receive data buffer is equal to or less than the number indicated by SCI_WATER[RXWATER]. When the receive FIFO is disabled, RDRF is set when the receive buffer (SCI_DATA) is full. To clear RDRF, read the SCI_DATA register.

A character that is in the process of being received does not cause a change in RDRF until the entire character is received. Even if RDRF is set, the character will continue to be received until an overrun condition occurs once the entire character is received.

1 = Receive data buffer full

0 = Receive data buffer empty

IDLE — Idle Line Flag

IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.

To clear IDLE, write logic 1 to the IDLE flag. After IDLE has been cleared, it cannot become set again until after a new character has been stored in the receive buffer or a LIN break character has set the LBKDIF flag. IDLE is set only once even if the receive line remains idle for an extended period.

1 = Idle line was detected

0 = No idle line detected

OR — Receiver Overrun Flag

OR is set when software fails to prevent the receive data register from overflowing with data. The OR bit is set immediately after the stop bit has been completely received for the dataword that overflows the buffer and all the other error flags (FE, NF, and PF) are prevented from setting. The data in the shift register is lost, but the data already in the SCI data registers is not affected. If LBKDE is enabled and a LIN Break is detected, the OR field asserts if LBKDIF is not cleared before the next data character is received.

While the OR flag is set, no additional data is stored in the data buffer even if sufficient room exists. To clear OR, write logic 1 to the OR flag.

1 = Receive overrun (new SCI data lost)

0 = No overrun

NF — Noise Flag

The advanced sampling technique used in the receiver takes three samples in each of the received bits. If any of these samples disagrees with the rest of the samples within any bit

time in the frame then noise is detected for that character. NF is set whenever the next character to be read from SCI_DATA was received with noise detected within the character. To clear NF, write logic one to the NF.

1 = Noise detected in the received character in SCI_DATA

0 = No noise detected

FE — Framing Error Flag

FE is set whenever the next character to be read from SCI_DATA was received with logic 0 detected where a stop bit was expected. To clear FE, write logic one to the FE.

1 = Framing error

0 = No framing error detected. This does not guarantee the framing is correct

PF — Parity Error Flag

PF is set whenever the next character to be read from SCI_DATA was received when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, write a logic one to the PF.

1 = Parity error

0 = No parity error

MA1F — Match 1 Flag

MA1F is set whenever the next character to be read from SCI_DATA matches MA1. To clear MA1F, write a logic one to the MA1F.

1 = Received data is equal to MA1

0 = Received data is not equal to MA1

MA2F — Match 2 Flag

MA2F is set whenever the next character to be read from SCI_DATA matches MA2. To clear MA2F, write a logic one to the MA2F.

1 = Received data is equal to MA2

0 = Received data is not equal to MA2

21.7.7. SCI Control Register

Offset Address: 0x0018

	31	30	29	28	27	26	25	24
R W	R8T9	R9T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	MA1IE	MA2IE	Reserved			IDLECFG		
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	LOOPS	DOZEEN	RSRC	M	WAKE	ILT	PE	PT
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-9: SCI Control Register (SCI_CTRL)

R8T9 — Receive Bit 8 / Transmit Bit 9

R8 is the ninth data bit received when the SCI is configured for 9bit or 10bit data formats. When reading 9bit or 10bit data, read R8 before reading SCI_DATA.

T9 is the tenth data bit received when the SCI is configured for 10bit data formats. When writing 10bit data, write T9 before writing SCI_DATA. If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time SCI_DATA is written.

R9T8 — Receive Bit 9 / Transmit Bit 8

R9 is the tenth data bit received when the SCI is configured for 10bit data formats. When reading 10bit data, read R9 before reading SCI_DATA

T8 is the ninth data bit received when the SCI is configured for 9bit or 10bit data formats. When writing 9bit or 10bit data, write T8 before writing SCI_DATA. If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time SCI_DATA is written.

TXDIR — TXD Pin Direction in Single-Wire Mode

When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TXD pin. When clearing TXDIR, the transmitter will finish receiving the current character (if any) before the receiver starts receiving data from the TXD pin.

1 = TXD pin is an output in single-wire mode

0 = TXD pin is an input in single-wire mode

TXINV — Transmit Data Inversion

Setting this bit reverses the polarity of the transmitted data output. Setting TXINV inverts the TXD output for all cases: data bits, start and stop bits, break, and idle.

1 = Transmit data inverted

0 = Transmit data not inverted

ORIE — Overrun Interrupt Enable

This bit enables the overrun flag (OR) to generate hardware interrupt requests.

1 = Hardware interrupt requested when OR is set

0 = OR interrupts disabled; use polling

NEIE — Noise Error Interrupt Enable

This bit enables the noise flag (NF) to generate hardware interrupt requests.

1 = Hardware interrupt requested when NF is set

0 = NF interrupts disabled; use polling

FEIE — Framing Error Interrupt Enable

This bit enables the framing error flag (FE) to generate hardware interrupt requests.

1 = Hardware interrupt requested when FE is set

0 = FE interrupts disabled; use polling

PEIE — Parity Error Interrupt Enable

This bit enables the parity error flag (PF) to generate hardware interrupt requests.

1 = Hardware interrupt requested when PF is set

0 = PF interrupts disabled; use polling)

TIE — Transmit Interrupt Enable

Enables STAT[TDRE] to generate interrupt requests.

1 = Hardware interrupt requested when TDRE flag is 1

0 = Hardware interrupts from TDRE disabled; use polling

TCIE — Transmission Complete Interrupt Enable for

TCIE enables the transmission complete flag, TC, to generate interrupt requests.

1 = Hardware interrupt requested when TC flag is 1

0 = Hardware interrupts from TC disabled; use polling

RIE — Receiver Interrupt Enable

Enables STAT[RDRF] to generate interrupt requests.

1 = Hardware interrupt requested when RDRF flag is 1

0 = Hardware interrupts from RDRF disabled; use polling

ILIE — Idle Line Interrupt Enable

ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests.

1 = Hardware interrupt requested when IDLE flag is 1

0 = Hardware interrupts from IDLE disabled; use polling

TE — Transmitter Enable

Enables the SCI transmitter. TE can also be used to queue an idle preamble by clearing and then setting TE. When TE is cleared, this register bit will read as 1 until the transmitter has completed the current character and the TXD pin is tristate.

1 = Transmitter enabled

0 = Transmitter disabled

RE — Receiver Enable

Enables the SCI receiver. When RE is written to 0, this register bit will read as 1 until the receiver finishes receiving the current character (if any).

1 = Receiver enabled

0 = Receiver disabled

RWU — Receiver Wakeup Control

This field can be set to place the SCI receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when CTRL[WAKE] is clear or an address match when CTRL[WAKE] is set with STAT[RWUID] is clear.

RWU must be set only with CTRL[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by STAT[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the SCI will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to be reasserted.

1 = SCI receiver in standby waiting for wakeup condition

0 = Normal receiver operation

SBK — Send Break

Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 to 13, or 13 to 16 if SCI_STATBRK13] is set, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK.

1 = Queue break character(s) to be sent

0 = Normal transmitter operation

MA1IE — Match 1 Interrupt Enable

1 = MA1F interrupt enabled

0 = MA1F interrupt disabled

MA2IE — Match 2 Interrupt Enable

1 = MA2F interrupt enabled

0 = MA2F interrupt disabled

IDLECFG — Idle Configuration

Configures the number of idle characters that must be received before the IDLE flag is set.

000 -- 1 idle character;

001 -- 2 idle characters;

010 -- 4 idle characters;

011 -- 8 idle characters;

100 -- 16 idle characters;

101 -- 32 idle characters;

110 -- 64 idle characters;

111 -- 128 idle characters

LOOPS — LOOP Mode Select

When LOOPS is set, the RXD pin is disconnected from the SCI and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function.

1 = Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input (see RSRC bit)

0 = Normal operation - RXD and TXD use separate pins

DOZEEN — Doze Enable

1 = SCI is disabled in Doze mode

0 = SCI is enabled in Doze mode

RSRC — Receiver Source Select

This field has no meaning or effect unless the LOOPS field is set. When LOOPS is set, the RSRC field determines the source for the receiver shift register input.

1 = Single-wire SCI mode where the TXD pin is connected to the transmitter output and receiver input

0 = Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the SCI does not use the RXD pin

M — 9bit or 8bit Mode Select

1 = Receiver and transmitter use 9bit data characters

0 = Receiver and transmitter use 8bit data characters

WAKE — Receiver Wakeup Method Select

Determines which condition wakes the SCI when RWU=1:Address mark in the most significant bit position of a received data character, or an idle condition on the receive pin input signal.

1 = Configures RWU with address-mark wakeup

0 = Configures RWU for idle-line wakeup

ILT — Idle Line Type Select

Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. In case the SCI is programmed with ILT = 1, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count.

1 = Idle character bit count starts after stop bit

0 = Idle character bit count starts after start bit

PE — Parity Enable

Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit.

1 = Parity enabled

0 = No hardware parity generation or checking

PT Parity Type

Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even.

1 = Odd parity

0 = Even parity

21.7.8. SCI Data Register

Offset Address: 0x001C

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	NOISY	PARITYE	FRETSC	RXEMPT	IDLINE	Reserved	R9T9	R8T8
W								
RESET	0	0	0	1	0	0	0	0
	7	6	5	4	3	2	1	0
R	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-10: SCI Data Register (SCI_DATA)

NOISY — The current received dataword contained in DATA[R9:R0] was received with noise

1 = The data was received with noise

0 = The dataword was received without noise

PARITYE — The current received dataword contained in DATA[R9:R0] was received with a parity error

1 = The dataword was received with a parity error

0 = The dataword was received without a parity error

FRETSC — Frame Error / Transmit Special Character

For reads, indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, indicates a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 is used to indicate a break character when 0 and an idle character when 1, he contents of

DATA[T8:T0] should be zero.

1 = The dataword was received with a frame error, transmit an idle or break character on transmit

0 = The dataword was received without a frame error on read, transmit a normal character on write

RXEMPT — Receive Buffer Empty

Asserts when there is no data in the receive buffer. This field does not take into account data that is in the receive shift register.

1 = Receive buffer is empty, data returned on read is not valid

0 = Receive buffer contains valid data

IDLINE — Idle Line

Indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled.

1 = Receiver was idle before receiving this character
0 = Receiver was not idle before receiving this character

R9T9 — Read receive data buffer 9 or write transmit data buffer 9
R8T8 — Read receive data buffer 8 or write transmit data buffer 8
R7T7 — Read receive data buffer 7 or write transmit data buffer 7
R6T6 — Read receive data buffer 6 or write transmit data buffer 6
R5T5 — Read receive data buffer 5 or write transmit data buffer 5
R4T4 — Read receive data buffer 4 or write transmit data buffer 4
R3T3 — Read receive data buffer 3 or write transmit data buffer 3
R2T2 — Read receive data buffer 2 or write transmit data buffer 2
R1T1 — Read receive data buffer 1 or write transmit data buffer 1
R0T0 — Read receive data buffer 0 or write transmit data buffer 0

21.7.9. SCI Match Address Register

Offset Address: 0x0020

	31	30	29	28	27	26	25	24
R	Reserved						MA2	
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	MA2							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	Reserved						MA1	
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	MA1							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-11: SCI Match Address Register (SCI_MATCH)

MA2 — Match Address 2

The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear.

MA1 — Match Address 1

The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear.

21.7.10. SCI Modem IrDA Register

Offset Address: 0x0024

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved					IREN	TNP	
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	RTSWATER							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved		TXCTSS-RC	TXCTSC	RXRTSE	TXRT-SPOL	TXRTSE	TXCTSE
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-12: SCI Modem IrDA Register (SCI_MODIR)

IREN — Infrared enable

Enables/disables the infrared modulation/demodulation.

1 = IR enabled

0 = IR disable

TNP — Transmitter narrow pulse

Enables whether the SCI transmits a 1/OSR, 2/OSR, 3/OSR or 4/OSR narrow pulse.

00 -- 1/OSR;

01 -- 2/OSR;

10 -- 3/OSR;

11 -- 4/OSR

RTSWATER — Receive RTS Configuration

Configures the point at which the RX RTS output negates based on the number of additional characters that can be stored in the Receive FIFO. When configured to 0, RTS negates when the start bit is detected for the character that will cause the FIFO to become full.

1 = RTS asserts when the receive FIFO is less than or equal to the RXWATER configuration and negates when the receive FIFO is greater than the RXWATER configuration.

0 = RTS asserts when the receiver FIFO is full or receiving a character that causes the FIFO to become full.

TXCTSSRC — Transmit CTS Source

Configures the source of the CTS input.

1 = CTS input is the inverted Receiver Match result

0 = CTS input is the SCI_CTS pin

TXCTSC — Transmit CTS Configuration

Configures if the CTS state is checked at the start of each character or only when the transmitter is idle.

1 = CTS input is sampled when the transmitter is idle

0 = CTS input is sampled at the start of each character

RXRTSE — Receiver request-to-send enable

Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. Do not set both RXRTSE and TXRTSE.

1 = RTS is deasserted if the receiver data register is full or a start bit has been detected that would cause the receiver data register to become full. RTS is asserted if the receiver data register is not full and has not detected a start bit that would cause the receiver data register to become full. RTS assertion is configured by the RTSWATER field.

0 = The receiver has no effect on RTS

TXRTSPOL — Transmitter request-to-send polarity

Controls the polarity of the transmitter RTS. TXRTSPOL does not affect the polarity of the receiver RTS. RTS will remain negated in the active low state unless TXRTSE is set.

1 = Transmitter RTS is active high

0 = Transmitter RTS is active low

TXRTSE — Transmitter request-to-send enable

Controls RTS before and after a transmission.

1 = When a character is placed into an empty transmitter data buffer, RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit

0 = The transmitter has no effect on RTS

TXCTSE — Transmitter clear-to-send enable

TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE.

1 = Enables clear-to-send operation. The transmitter checks the state of CTS each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the signal TXD remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS as a character is being sent do not affect its transmission

0 = CTS has no effect on the transmitter

21.7.11. SCI FIFO Register

Offset Address: 0x0028

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	TXEMPT	RXEMPT	Reserved				TXOF	RXUF
W							w1c	w1c
RESET	1	1	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	0	0	Reserved	RXIDEN			TXOFE	RXUFE
W	TXFLUSH	RXFLUSH						
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	TXFE	TXFIFOSIZE			RXFE	RXFIFOSIZE		
W								
RESET:	0	0	1	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-13: SCI FIFO Register (SCI_FIFO)

TXEMPT — Transmit Buffer/FIFO Empty

Asserts when there is no data in the Transmit FIFO/buffer. This field does not take into account data that is in the transmit shift register.

1 = Transmit buffer is empty

0 = Transmit buffer is not empty

RXEMPT — Receive Buffer/FIFO Empty

Asserts when there is no data in the receive FIFO/Buffer. This field does not take into account data that is in the receive shift register.

1 = Receive buffer is empty

0 = Receive buffer is not empty

TXOF — Transmitter Buffer Overflow Flag

Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of TXOFE. However, an interrupt will be issued to the host only if TXOFE is set. This flag is cleared by writing a 1.

1 = At least one transmit buffer overflow has occurred since the last time the flag was cleared

0 = No transmit buffer overflow has occurred since the last time the flag was cleared

RXUF — Receiver Buffer Underflow Flag

Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of RXUFE. However, an interrupt will be issued to the host only if RXUFE is set. This flag is cleared by writing a 1.

1 = At least one receive buffer underflow has occurred since the last time the flag was cleared

0 = No receive buffer underflow has occurred since the last time the flag was cleared

TXFLUSH — Transmit FIFO/Buffer Flush

Writing to this field causes all data that is stored in the transmit FIFO/buffer to be flushed. This does not affect data that is in the transmit shift register.

1 = All data in the transmit FIFO/Buffer is cleared out

0 = No flush operation occurs

RXFLUSH — Receive FIFO/Buffer Flush

Writing to this field causes all data that is stored in the receive FIFO/buffer to be flushed. This does not affect data that is in the receive shift register.

1 = All data in the receive FIFO/buffer is cleared out

0 = No flush operation occurs

RXIDEN — Receiver Idle Empty Enable

When set, enables the assertion of RDRF when the receiver is idle for a number of idle characters and the FIFO is not empty.

000 -- Disable RDRF assertion due to partially filled FIFO when receiver is idle;

001 -- Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character;

010 -- Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters;

011 -- Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters;

100 -- Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters;

101 -- Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters;

110 -- Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters;

111 -- Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters.

TXOFE — Transmit FIFO Overflow Interrupt Enable

When this field is set, the TXOF flag generates an interrupt to the host.

1 = TXOF flag generates an interrupt to the host

0 = TXOF flag does not generate an interrupt to the host

RXUFE — Receive FIFO Underflow Interrupt Enable

When this field is set, the RXUF flag generates an interrupt to the host.

1 = TRXUF flag generates an interrupt to the host

0 = RXUF flag does not generate an interrupt to the host

TXFE — Transmit FIFO Enable

When this field is set, the built in FIFO structure for the transmit buffer is enabled. The size of the FIFO structure is indicated by TXFIFOSIZE. If this field is not set, the transmit buffer operates as a FIFO of depth one dataword regardless of the value in TXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.

1 = Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE

0 = Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support)

TXFIFOSIZE — Transmit FIFO/Buffer Depth

The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only.

000 -- Transmit FIFO/Buffer depth = 1 dataword;

001 -- Transmit FIFO/Buffer depth = 4 datawords;

010 -- Transmit FIFO/Buffer depth = 8 datawords;

011 -- Transmit FIFO/Buffer depth = 16 datawords;

100 -- Transmit FIFO/Buffer depth = 32 datawords;

101 -- Transmit FIFO/Buffer depth = 64 datawords;

110 -- Transmit FIFO/Buffer depth = 128 datawords;

111 -- Transmit FIFO/Buffer depth = 256 datawords.

RXFE — Receive FIFO Enable

When this field is set, the built in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.

1 = Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE

0 = Receive FIFO is not enabled. Buffer is depth 1. (Legacy support)

RXFIFOSIZE — Receive FIFO/Buffer Depth

The maximum number of transmit datawords that can be stored in the receive buffer. This field is read only.

000 -- Receive FIFO/Buffer depth = 1 dataword;

001 -- Receive FIFO/Buffer depth = 4 datawords;

010 -- Receive FIFO/Buffer depth = 8 datawords;

011 -- Receive FIFO/Buffer depth = 16 datawords;

100 -- Receive FIFO/Buffer depth = 32 datawords;

101 -- Receive FIFO/Buffer depth = 64 datawords;

110 -- Receive FIFO/Buffer depth = 128 datawords;

111 -- Receive FIFO/Buffer depth = 256 datawords.

21.7.12. SCI Watermark Register

Offset Address: 0x002C

	31	30	29	28	27	26	25	24
R	RXCOUNT							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	RXWATER							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	TXCOUNT							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	TXWATER							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-14: SCI Watermark Register (SCI_WATER)

RXCOUNT — Receive Counter

The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer.

RXWATER — Receive Watermark

When the number of datawords in the receive FIFO/buffer is greater than the value in this register field, an interrupt is generated. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE] and must be greater than 0.

TXCOUNT — Transmit Counter

The value in this register indicates the number of datawords that are in the transmit FIFO/buffer. If a dataword is being transmitted, that is, in the transmit shift register, it is not included in the count. This value may be used in conjunction with FIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer.

TXWATER — Transmit Watermark

When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt is generated. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by FIFO[TXFIFOSIZE] and FIFO[TXFE].

21.7.13. SCI Oversampling Ratio Register

Offset Address: 0x0030

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	OSR							
W								
RESET:	0	0	0	0	1	1	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 21-15: SCI Oversampling Ratio Register (SCI_OSR)

OSR — Oversampling Ratio

This field configures the oversampling ratio for the receiver between 4x (00000011) and 256x (11111111). Writing an invalid oversampling ratio (for example, a value not between 4x and 256x) will default to an oversampling ratio of 16 (00001111). The OSR field should only be changed when the transmitter and receiver are both disabled. That the oversampling ratio = OSR + 1.

21.8. Functional Description

The SCI supports full-duplex, asynchronous, NRZ serial communication and comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. The following describes each of the blocks of the SCI.

21.9. Baud Rate Generation

A 13bit modulus counter in the baud rate generator derive the baud rate for both the receiver and the transmitter. The value from 1 to 8191 written to SBR[12:0] determines the baud clock divisor for the asynchronous SCI baud clock. The SBR bits are in the SCI baud rate registers, BDH and BDL. The baud rate clock drives the receiver, while the transmitter is driven by the baud rate clock divided by the over sampling ratio. Depending on the over sampling ratio, the receiver has an acquisition rate of 4 to 256 samples per bit time.

Baud rate generation is subject to two sources of error:

- Integer division of the asynchronous SCI baud clock may not give the exact target frequency.
- Synchronization with the asynchronous SCI baud clock can cause phase shift

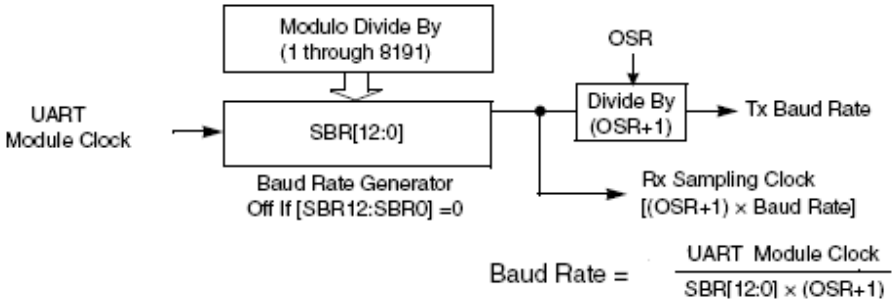


Figure 21-16: SCI Baud Rate Generation

21.10. Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters.

The transmitter output (TXD) idle state defaults to logic high, CTRL[TXINV] is cleared following reset. The transmitter output is inverted by setting CTRL[TXINV]. The transmitter is enabled by setting the CTRL[TE] bit. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register.

The central element of the SCI transmitter is the transmit shift register that is 10bit to 13 bits long depending on the setting in the CTRL[M], BAUD[M10] and

BAUD[SBNS] control bits. For the remainder of this section, assume CTRL[M], BAUD[M10] and BAUD[SBNS] are cleared, selecting the normal 8bit data mode. In 8bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new character, the value waiting in the transmit data register is transferred to the shift register, synchronized with the baud rate clock, and the transmit data register empty (STAT[TDRE]) status flag is set to indicate another character may be written to the transmit data buffer at SCI_DATA.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TXD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TXD high, waiting for more characters to transmit.

Writing 0 to CTRL[TE] does not immediately disable the transmitter. The current transmit activity in progress must first be completed (that could include a data character, idle character or break character), although the transmitter will not start transmitting another character.

21.10.1. Send Break And Queued Idle

The SCI_CTRL[SBK] bit sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0,10bit to 12bit times including the start and stop bits. A longer break of 13bit times can be enabled by setting SCI_STAT[BRK13]. Normally, a program would wait for SCI_STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, write 1, and then write 0 to the SCI_CTRL[SBK] bit. This action queues a break character to be sent as soon as the shifter is available. If SCI_CTRL[SBK] remains 1 when the queued break moves into the shifter, synchronized to the baud rate clock, an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters are received as 0s in all data bits and a framing error (SCI_STAT[FE] = 1) occurs.

A break character can also be transmitted by writing to the SCI_DATA register with bit 13 set and the data bits clear. This supports transmitting the break character as part of the normal data stream.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for SCI_STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the SCI_CTRL[TE] bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while SCI_CTRL[TE] is cleared, the SCI transmitter never actually releases control of the TXD pin.

An idle character can also be transmitted by writing to the SCI_DATA register with bit 13 set and the data bits also set. This supports transmitting the idle character as part of the normal data stream.

The length of the break character is affected by the SCI_STAT[BRK13], SCI_CTRL[M], SCI_BAUD[M10] and SCI_BAUD[SNBS] bits as shown below.

Table 21-3: Break Character Length

BRK13	M	M10	SBNS	Break Character Length
0	0	0	0	10bit times
0	0	0	1	11bit times
0	1	0	0	11bit times
0	1	0	1	12bit times
0	X	1	0	12bit times
0	X	1	1	13bit times
1	0	0	0	13bit times
1	0	0	1	13bit times
1	1	0	0	14bit times

BRK13	M	M10	SBNS	Break Character Length
1	1	0	1	14bit times
1	X	1	0	15bit times
1	X	1	1	15bit times

21.11. Receiver Functional Description

In this section, the receiver block diagram is a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, different variations of the receiver wakeup function are explained.

The receiver input is inverted by setting SCI_STAT[RXINV]. The receiver is enabled by setting the SCI_CTRL[RE] bit. Character frames consist of a start bit of logic 0, eight to ten data bits (MSB or LSB first), and one or two stop bits of logic 1. For information about 9bit or 10bit data mode, refer to 8bit, 9bit and 10bit data modes. For the remainder of this discussion, assume the SCI is configured for normal 8bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (SCI_STAT[RDRF]) status flag is set. If SCI_STAT[RDRF] was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after SCI_STAT[RDRF] is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (SCI_STAT[RDRF] =1), it gets the data from the receive data register by reading SCI_DATA. Refer to Interrupts and status flags for details about flag clearing.

21.11.1. Data Sampling Technique

The SCI receiver supports a configurable oversampling rate of between 4× and 256× of the baud rate clock for sampling. The receiver starts by taking logic level samples at the oversampling rate times the baud rate to search for a falling edge on the RXD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The oversampling baud rate clock divides the bit time into 4 to 256 segments from 1 to OSR (where OSR is the configured oversampling ratio). When a falling edge is located, three more samples are taken at (OSR/2), (OSR/2)+1, and (OSR/2)+2 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a received character. If another falling edge is detected before the receiver is considered synchronized, the receiver restarts the sampling from the first segment.

The receiver then samples each bit time, including the start and stop bits, at (OSR/2), (OSR/2)+1, and (OSR/2)+2 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. If any sample in any bit time, including the start and stop bits, in a character frame fails to agree with the logic level for that bit, the noise flag (SCI_STAT[NF]) is set when the received character is transferred to the receive data buffer.

When the SCI receiver is configured to sample on both edges of the baud rate clock, the number of segments in each received bit is effectively doubled (from 1 to OSR×2). The start and data bits are then sampled at OSR, OSR+1 and OSR+2. Sampling on both edges of the clock must be enabled for oversampling rates of 4× to 7× and is optional for higher oversampling rates.

The falling edge detection logic continuously looks for falling edges. If an edge is detected, the sample clock is resynchronized to bit times (unless resynchronization has been disabled). This improves the

reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

21.11.2. Receiver Wakeup Operation

Receiver wakeup and receiver address matching is a hardware mechanism that allows an SCI receiver to ignore the characters in a message intended for a different receiver.

During receiver wakeup, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up control bit (SCI_CTRL[RWU]). When RWU bit and SCI_S2[RWUID] bit are set, the status flags associated with the receiver, with the exception of the idle bit, IDLE, are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force SCI_CTRL[RWU] to 0 so all receivers wake up in time to look at the first character(s) of the next message.

During receiver address matching, the address matching is performed in hardware and the SCI receiver will ignore all characters that do not meet the address match requirements.

The length of the break character is affected by the SCI_STAT[BRK13], SCI_CTRL[M], SCI_BAUD[M10] and SCI_BAUD[SNBS] bits as shown below.

Table 21-4: Receiver Wakeup Options

RWU	MA1 | MA2	MATCFG	WAKE:RWUID	Receiver Wakeup
0	0	X	X	Normal operation
1	0	00	00	Receiver wakeup on idle line, IDLE flag not set
1	0	00	01	Receiver wakeup on idle line, IDLE flag set
1	0	00	10	Receiver wakeup on address mark
1	1	11	0	Receiver wakeup on data match
0	1	00	X0	Address mark address match, IDLE flag not set for discarded characters
0	1	00	X1	Address mark address match, IDLE flag set for discarded characters
0	1	01	X0	Idle line address match
0	1	10	X0	Address match on and address match off, IDLE flag not set for discarded characters
0	1	10	X1	Address match on and address match off, IDLE flag set for discarded characters

21.11.2.1. Idle-line Wakeup

When wake is cleared, the receiver is configured for idle-line wakeup. In this mode, SCI_CTRL[RWU] is cleared automatically when the receiver detects a full character time of the idle-line level. The SCI_CTRL[M] and SCI_BAUD[M10] control bit selects 8bit to 10bit data mode and the SCI_BAUD[SBNS] bit selects 1bit or 2bit stop bit number that determines how many bit times of idle are needed to constitute a full character time, 10 to 13 bit times because of the start and stop bits.

When SCI_CTRL[RWU] is one and SCI_STAT[RWUID] is zero, the idle condition that wakes up the receiver does not set the SCI_STAT[IDLE] flag. The receiver wakes up and waits for the first data character of the next message that sets the SCI_STAT[RDRF] flag and generates an interrupt if enabled. When SCI_STAT[RWUID] is one, any idle condition sets the SCI_STAT[IDLE] flag and generates an interrupt if enabled, regardless of whether SCI_CTRL[RWU] is zero or one.

The idle-line type (SCI_CTRL[ILT]) control bit selects one of two ways to detect an idle line. When SCI_CTRL[ILT] is cleared, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When SCI_CTRL[ILT] is set, the idle bit counter does not start until after the stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

21.11.2.2. Address-mark Wakeup

When SCI_CTRL[WAKE] is set, the receiver is configured for address-mark wakeup. In this mode, SCI_CTRL[RWU] is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character.

Address-mark wakeup allows messages to contain idle characters, but requires the MSB be reserved for use in address frames. The logic 1 in the MSB of an address frame clears the SCI_CTRL[RWU] bit before the stop bits are received and sets the SCI_STAT[RDRF] flag. In this case, the character with the MSB set is received even though the receiver was sleeping during most of this character time.

21.11.2.3. Data Match Wakeup

When SCI_CTRL[RWU] is set and SCI_BAUD[MATCFG] equals 11, the receiver is configured for data match wakeup. In this mode, SCI_CTRL[RWU] is cleared automatically when the receiver detects a character that matches MATCH[MA1] field when BAUD[MAEN1] is set, or that matches MATCH[MA2] when BAUD[MAEN2] is set.

21.11.2.4. Address Match Operation

Address match operation is enabled when the SCI_BAUD[MAEN1] or SCI_BAUD[MAEN2] bit is set and SCI_BAUD[MATCFG] is equal to 00. In this function, a character received by the RXD pin with a logic 1 in the bit position immediately preceding the stop bit is considered an address and is compared with the associated MATCH[MA1] or MATCH[MA2] field. The character is only transferred to the receive buffer, and SCI_STAT[RDRF] is set, if the comparison matches. All subsequent characters received with a logic 0 in the bit position immediately preceding the stop bit are considered to be data associated with the address and are transferred to the receive data buffer. If no marked address match occurs then no transfer is made to the receive data buffer, and all following characters with logic zero in the bit position immediately preceding the stop bit are also discarded. If both the SCI_BAUD[MAEN1] and SCI_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Address match operation functions in the same way for both MATCH[MA1] and MATCH[MA2] fields.

• If only one of SCI_BAUD[MAEN1] and SCI_BAUD[MAEN2] is asserted, a marked address is compared only with the associated match register and data is transferred to the receive data buffer only on a match.

• If SCI_BAUD[MAEN1] and SCI_BAUD[MAEN2] are asserted, a marked address is compared with both match registers and data is transferred only on a match with either register.

21.11.2.5. Idle Match Operation

Idle match operation is enabled when the SCI_BAUD[MAEN1] or SCI_BAUD[MAEN2] bit is set and SCI_BAUD[MATCFG] is equal to 01. In this function, the first character received by the RXD pin after an idle line condition is considered an address and is compared with the associated MA1 or MA2 register. The character is only transferred to the receive buffer, and SCI_STAT[RDRF] is set, if the comparison matches. All subsequent characters are considered to be data associated with the address and are transferred to the receive data buffer until the next idle line condition is detected. If no address match occurs then no transfer is made to the receive data buffer, and all following frames until the next idle condition are also discarded. If both the SCI_BAUD[MAEN1] and SCI_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Idle match operation functions in the same way for both MA1 and MA2 registers.

- If only one of SCI_BAUD[MAEN1] and SCI_BAUD[MAEN2] is asserted, the first character after an idle line is compared only with the associated match register and data is transferred to the receive data buffer only on a match.
- If SCI_BAUD[MAEN1] and SCI_BAUD[MAEN2] are asserted, the first character after an idle line is compared with both match registers and data is transferred only on a match with either register.

21.11.2.6. Match On Match Off Operation

Match on, match off operation is enabled when both SCI_BAUD[MAEN1] and SCI_BAUD[MAEN2] are set and SCI_BAUD[MATCFG] is equal to 10. In this function, a character received by the RXD pin that matches MATCH[MA1] is received and transferred to the receive buffer, and SCI_STAT[RDRF] is set. All subsequent characters are considered to be data and are also transferred to the receive data buffer, until a character is received that matches MATCH[MA2] register. The character that matches MATCH[MA2] and all following characters are discarded, this continues until another character that matches MATCH[MA1] is received. If both the SCI_BAUD[MAEN1] and SCI_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

21.11.3. Infrared Decoder

The infrared decoder converts the received character from the IrDA format to the NRZ format used by the receiver. It also has a OSR oversampling baud rate clock counter that filters noise and indicates when a 1 is received.

21.11.3.1. Start Bit Detection

When STAT[RXINV] is cleared, the first falling edge of the received character corresponds to the start bit. The infrared decoder resets its counter. At this time, the receiver also begins its start bit detection process. After the start bit is detected, the receiver synchronizes its bit times to this start bit time. For the rest of the character reception, the infrared decoder's counter and the receiver's bit time counter count independently from each other.

21.11.3.2. Noise Filtering

Any further rising edges detected during the first half of the infrared decoder counter are ignored by the decoder. Any pulses less than one oversampling baud clock can be undetected by it regardless of whether it is seen in the first or second half of the count.

21.11.3.3. Low-bit Detection

During the second half of the decoder count, a rising edge is decoded as a 0, which is sent to the receiver. The decoder counter is also reset.

21.11.3.4. High-bit Detection

At OSR oversampling baud rate clocks after the previous rising edge, if a rising edge is not seen, then the decoder sends a 1 to the receiver.

If the next bit is a 0, which arrives late, then a low-bit is detected according to Low-bit detection. The value sent to the receiver is changed from 1 to a 0. Then, if a noise pulse occurs outside the receiver's bit time sampling period, then the delay of a 0 is not recorded as noise.

21.12. Additional SCI Functions

The following sections describe additional SCI functions.

21.12.1. 8bit, 9bit and 10bit Data Modes

The SCI transmitter and receiver can be configured to operate in 9bit data mode by setting the SCI_CTRL[M] or 10bit data mode by setting SCI_CTRL[M10]. In 9bit mode, there is a ninth data bit in 10bit mode there is a tenth data bit. For the transmit data buffer, these bits are stored in SCI_CTRL[T8] and SCI_CTRL[T9]. For the receiver, these bits are held in SCI_CTRL[R8] and SCI_CTRL[R9]. They are also accessible via 16bit or 32bit accesses to the SCI_DATA register.

For coherent 8bit writes to the transmit data buffer, write to SCI_CTRL[T8] and SCI_CTRL[T9] before writing to SCI_DATA[7:0]. For 16bit and 32bit writes to the SCI_DATA register all 10 transmit bits are written to the transmit data buffer at the same time.

If the bit values to be transmitted as the ninth and tenth bit of a new character are the same as for the previous character, it is not necessary to write to SCI_CTRL[T8] and SCI_CTRL[T9] again. When data is transferred from the transmit data buffer to the transmit shifter, the value in SCI_CTRL[T8] and SCI_CTRL[T9] is copied at the same time data is transferred from SCI_DATA[7:0] to the shifter.

The 9bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. The 10bit data mode is typically used with parity and address-mark wakeup so the ninth data bit can serve as the wakeup bit and the tenth bit as the parity bit. In custom protocols, the ninth and/or tenth bits can also serve as software-controlled markers.

21.12.2. Idle Length

An idle character is a character where the start bit, all data bits and stop bits are in the mark position. The CTRL[ILT] register can be configured to start detecting an idle character from the previous start bit (any data bits and stop bits count towards the idle character detection) or from the previous stop bit. The number of idle characters that must be received before an idle line condition is detected can also be configured using the CTRL[IDLECFG] field. This field configures the number of idle characters that must be received before the STAT[IDLE] flag is set, the STAT[RAF] flag is cleared and the

DATA[IDLINE] flag is set with the next received character.

Idle-line wakeup and idle match operation are also affected by the CTRL[IDLECFG] field. When address match or match on/off operation is enabled, setting the STAT[RWUID] bit will cause any discarded characters to be treated as if they were idle characters. independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RXD pin is not used by the SCI.

21.12.3. Single-wire Operation

When SCI_CTRL[LOOPS] is set, the SCI_CTRL[RSRC] bit in the same register chooses between loop mode (SCI_CTRL[RSRC] = 0) or single-wire mode (SCI_CTRL[RSRC] = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RXD pin is not used by the SCI.

21.12.4. Loop Mode

When SCI_CTRL[LOOPS] is set, the RSRC bit in the same register chooses between loop mode (SCI_CTRL[RSRC] = 0) or single-wire mode (SCI_CTRL[RSRC] = 1). Single-wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TXD pin (the RXD pin is not used).

In single-wire mode, the SCI_CTRL[TXDIR] bit controls the direction of serial data on the TXD pin. When SCI_CTRL[TXDIR] is cleared, the TXD pin is an input to the receiver and the transmitter is temporarily disconnected from the TXD pin so an external device can send serial data to the receiver. When SCI_CTRL[TXDIR] is set, the TXD pin is an output driven by the transmitter, the internal loop back connection is disabled, and as a result the receiver cannot receive characters that are sent out by the transmitter.

21.13. Infrared Interface

The SCI provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses and transforming them to serial bits, which are sent to the SCI. The IrDA physical layer specification defines a half-duplex infrared communication link for exchanging data. The full standard includes data rates up to 16 Mbits/s. This design covers data rates only between 2.4 kbits/s and 115.2 kbits/s.

The SCI has an infrared transmit encoder and receive decoder. The SCI transmits serial bits of data that are encoded by the infrared submodule to transmit a narrow pulse for every zero bit. No pulse is transmitted for every one bit. When receiving data, the IR pulses are detected using an IR photo diode and transformed to CMOS levels by the IR receive decoder, external from the SCI. The narrow pulses are then stretched by the infrared receive decoder to get back to a serial bit stream to be received by the UART. The polarity of transmitted pulses and expected receive pulses can be inverted so that a direct connection can be made to external IrDA transceiver modules that use active high pulses.

The infrared submodule receives its clock sources from the SCI. One of these two clocks are selected in the infrared submodule to generate either 1/OSR, 2/OSR, 3/OSR, or 4/OSR narrow pulses during transmission.

21.13.1. Infrared Transmit Encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the TXD signal. A narrow pulse is transmitted for a zero bit and no pulse for a one bit. The narrow pulse is sent at the start of the bit with a duration of 1/OSR, 2/OSR, 3/OSR, or 4/OSR of a bit time. A narrow low pulse is transmitted for a zero bit when SCI_CTRL[TXINV] is cleared, while a narrow high pulse is transmitted for a zero bit when SCI_CTRL[TXINV] is set.

21.13.2. Infrared Receive Decoder

The infrared receive block converts data from the RXD signal to the receive shift register. A narrow pulse is expected for each zero received and no pulse is expected for each one received. A narrow low pulse is expected for a zero bit when SCI_STAT[RXINV] is cleared, while a narrow high pulse is expected for a zero bit when SCI_STAT[RXINV] is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

21.14. Interrupts and Status Flags

The SCI transmitter has two status flags that can optionally generate hardware interrupt requests. Transmit data register empty SCI_STAT[TDRE]) indicates when there is room in the transmit data buffer to write another transmit character to SCI_DATA. If the transmit interrupt enable SCI_CTRL[TIE]) bit is set, a hardware interrupt is requested when SCI_STAT[TDRE] is set. Transmit complete (SCI_STAT[TC]) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TXD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (SCI_CTRL[TCIE]) bit is set, a hardware interrupt is requested when SCI_STAT[TC] is set. Instead of hardware interrupts, software polling may be used to monitor the SCI_STAT[TDRE] and SCI_STAT[TC] status flags if the corresponding SCI_CTRL[TIE] or SCI_CTRL[TCIE] local interrupt masks are cleared.

When a program detects that the receive data register is full (SCI_STAT[RDRF] =1), it gets the data from the receive data register by reading SCI_DATA. The SCI_STAT[RDRF] flag is cleared by reading SCI_DATA.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RXD line remains idle for an extended period of time. IDLE is cleared by writing 1 to the SCI_STAT[IDLE] flag. After SCI_STAT[IDLE] has been cleared, it cannot become set again until the receiver has received at least one new character and has set SCI_STAT[RDRF].

If the associated error was detected in the received character that caused SCI_STAT[RDRF] to be set, the error flags - noise flag (SCI_STAT[NF]), framing error (SCI_STAT[FE]), and parity error flag (SCI_STAT[PF]) - are set at the same time as SCI_STAT[RDRF]. These flags are not set in overrun cases.

If SCI_STAT[RDRF] was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (SCI_STAT[OR]) flag is set instead of the data along with any associated NF, FE, or PF condition is lost.

If the received character matches the contents of MATCH[MA1] and/or MATCH[MA2] then the SCI_STAT[MA1F] and/or SCI_STAT[MA2F] flags are set at the same time that SCI_STAT[RDRF] is set.

At any time, an active edge on the RXD serial data input pin causes the SCI_STAT[RXEDGIF] flag to set. The SCI_STAT[RXEDGIF] flag is cleared by writing a 1 to it. This function depends on the receiver being enabled (SCI_CTRL[RE] = 1).

22. Synchronous Serial Interface (SSI)

22.1. Introduction

SSI is a programmable Synchronous Serial Interface (SSI) peripheral. This is a AMBA 2.0-compliant AHB (Advanced High-performance Bus) component. The host processor accesses data, control, and status information on the SSI through the AHB interface. The SSI may also interface with a DMA Controller using an optional set of DMA signals, which can be selected during configuration.

22.2. Features

Feathers include:

- Serial-master operation.
- DMA controller interface – Enables the SSI to interface to a DMA controller over the bus using handshaking interface for transfer requests.
- Clock stretching support in enhanced SPI transfers.
- Data item size (4 to 32bits) – Item size of each data transfer under control of the programmer.
- FIFO depth – The transmit and receive FIFO buffers are 8 words deep. The FIFO width is fixed at 32bits.
- Enhanced SPI support.
- Execute in Place (XIP) mode support.

22.3. Modes of Operation

The SSI functions in these three modes:

1. Run mode — Run mode is the normal mode of operation.
2. Doze mode — Doze mode is a configurable low-power mode.
3. Stop mode — The SSI is inactive in stop mode.

22.4. Block Diagram

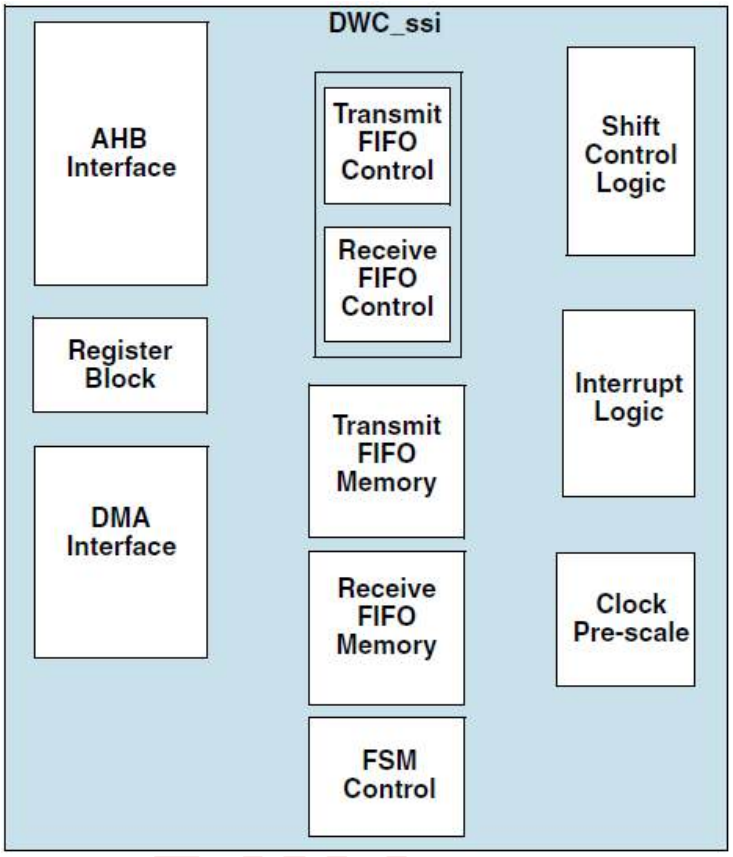


Figure 22-1: SSI Block Diagram

22.5. Module Memory Map

The SSI register memory map is as below.

Table 22-1: SSI Memory Map

Address Offset	Bits 31 ~ 0	Access
0x0000	Control Register 0(CTRLR0)	S/U
0x0004	Control Register 1 (CTRLR1)	S/U
0x0008	SSI Enable Register (SSIENR)	S/U
0x0010	Slave Select Register(SER)	S/U
0x000C	Microwire Control Register(MWCR)	S/U
0x0014	Baud Rate Select Register(BAUDR)	S/U
0x0018	Transmit FIFO Threshold Level Register(TXFTLR)	S/U
0x001C	Receive FIFO Threshold Level Register(RXFTLR)	S/U
0x0020	Transmit FIFO Level Register (TXFLR)	S/U
0x0024	Receive FIFO Level Register (RXFLR)	S/U
0x0028	Status Register (SR)	S/U
0x002C	Interrupt Mask Register (IMR)	S/U
0x0030	Interrupt Status Register (ISR)	S/U
0x0034	Raw Interrupt Status Register (RISR)	S/U
0x0038	Transmit FIFO Overflow Interrupt Clear Register(TXOICR)	S/U
0x003C	Receive FIFO Overflow Interrupt Clear Register(RXOICR)	S/U
0x0040	Receive FIFO Underflow Interrupt Clear Register(RXUICR)	S/U
0x0048	Interrupt Clear Register(ICR)	S/U
0x004C	DMA Control Register(DMACR)	S/U
0x0050	DMA Transmit Data Level Register(DMATDLR)	S/U
0x0054	DMA Receive Data Level Register(DMARDLR)	S/U
0x0058	Identification Register(IDR)	S/U
0x005C	Version ID Register(VIDR)	S/U
0x0060+i*0x4	SSI Data Register (DRx)	S/U
0x00F0	RX Sample Delay Register (RXSDR)	S/U
0x00F4	SPI Control Register 0(SPICTRLR0)	S/U
0x00FC	XIP Mode Bits(XIPMBR)	S/U
0x0100	XIP Incr Inst Register(XIPIIR)	S/U
0x0104	XIP Wrap Inst Register(XIPWIR)	S/U

Address Offset	Bits 31 ~ 0	Access
0x0108	XIP Control Register(XIPCR)	S/U
0x010C	XIP Slave Enable Register(XIPSER)	S/U
0x0110	XIP Receive FIFO Overflow Interrupt Clear Register(XRXIOCR)	S/U
0x0114	XIP Continues Transfer Time Out Register(XIPCTTOR)	S/U

22.6. Register Descriptions

22.6.1. Control Register 0(CTRLR0)

Offset Address: 0x0000

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	reserved
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	SPI_FRF		0	0	CFS			
W								
RESET:	1	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	SSTE	SRL	SLV_OE	TMOD		SCPOL	SCPH
W								
RESET:	0	1	0	0	1	0	0	0

	7	6	5	4	3	2	1	0
R	FRF		0	DFS				
W								
RESET:	0	0	0	1	1	1	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-2: Control Register 0(CTRLR0)

This register controls the serial data transfer. It is impossible to write to this register when the SSI is enabled. The SSI is enabled and disabled by writing to the SSIENR.

SPI_FRF — SPI Frame Format

Selects data frame format for Transmitting/Receiving the data. Bits only valid when SSIC_SPI_MODE is either set to "Dual" or "Quad" or "Octal" mode.

0x0 (SPI_STANDARD): Standard SPI Format

0x1 (SPI_DUAL): Dual SPI Format

0x2 (SPI_QUAD): Quad SPI Format

0x3 (SPI_OCTAL): Octal SPI Format

CFS — Control Frame Size

Selects the length of the control word for the Microwire frame format.

0x0 (SIZE_01_BIT): 01-bit Control Word

0x1 (SIZE_02_BIT): 02-bit Control Word

0x2 (SIZE_03_BIT): 03-bit Control Word
0x3 (SIZE_04_BIT): 04-bit Control Word
0x4 (SIZE_05_BIT): 05-bit Control Word
0x5 (SIZE_06_BIT): 06-bit Control Word
0x6 (SIZE_07_BIT): 07-bit Control Word
0x7 (SIZE_08_BIT): 08-bit Control Word
0x8 (SIZE_09_BIT): 09-bit Control Word
0x9 (SIZE_10_BIT): 10-bit Control Word
0xA (SIZE_11_BIT): 11-bit Control Word
0xB (SIZE_12_BIT): 12-bit Control Word
0xC (SIZE_13_BIT): 13-bit Control Word
0xD (SIZE_14_BIT): 14-bit Control Word
0xE (SIZE_15_BIT): 15-bit Control Word
0xF (SIZE_16_BIT): 16-bit Control Word

SSTE — Slave Select Toggle Enable.

While operating in SPI mode with clock phase (SCPH) set to 0, this register controls the behavior of the slave select line (ss_*_n) between data frames.

1 = (TOGGLE_EN): ss_*_n line will toggle between consecutive data frames, with the serial clock (SCLK) being held to its default value while ss_*_n is high

0 = (TOGGLE_DISABLE): ss_*_n will stay low and sclk will run continuously for the duration of the transfer

SRL — Shift Register Loop.

Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. Can be used in both serial-slave and serial-master modes. When the SSI is configured as a slave in loopback mode, the ss_in_n and ssi_clk signals must be provided by an external source. In this mode, the slave cannot generate these signals because there is nothing to which to loop back.

1 = (TESTING_MODE): Test Mode Operation

0 = (NORMAL_MODE): Normal mode operation

SLV_OE — Slave Output Enable.

Relevant only when the SSI is configured as a serial-slave device. When configured as a serial master, this bit field has no functionality.

1 = (DISABLED): Slave Output is disabled

0 = (ENABLED): Slave Output is enabled

TMOD — Transfer Mode.

Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid.

0x0 (TX_AND_RX): Transmit & Receive; Not Applicable in enhanced SPI operating mode

0x1 (TX_ONLY): Transmit only mode; Or write in enhanced SPI operating mode

0x2 (RX_ONLY): Receive only mode; Or read in enhanced SPI operating mode

0x3 (EEPROM_READ): EEPROM Read mode; Not Applicable in enhanced SPI operating mode

SCOPL — Serial Clock Polarity.

Valid when the frame format (FRF) is set to Motorola SPI. Used to select the polarity of the inactive serial clock, which is held inactive when the SSI master is not actively transferring data on the serial bus.

1 = (INACTIVE_LOW): Inactive state of serial clock is high

0 = (INACTIVE_HIGH): Inactive state of serial clock is low

SCPH — Serial Clock Phase.

Valid when the frame format (FRF) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal.

When SCPH = 0, data are captured on the first edge of the serial clock. When SCPH = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data are captured on the second edge of the serial clock.

1 = (START_BIT): Serial clock toggles at start of first bit

0 = (MIDDLE_BIT): Serial clock toggles in middle of first bit

FRF — Frame Format.

Selects which serial protocol transfers the data.

0x0 (SPI): Motorola SPI Frame Format

0x1 (SSP): Texas Instruments SSP Frame Format

0x2 (MICROWIRE): National Semiconductors Microwire Frame Format

0x3 (RESERVED): Reserved

DFS — Data Frame Size.

Selects the data frame length. When the data frame size is programmed to be less than 32bits, the receive data is automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded.

You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data.

0x0 (DFS_01_BIT): Reserved

0x1 (DFS_02_BIT): Reserved

0x2 (DFS_03_BIT): Reserved

0x3 (DFS_04_BIT): 04-bit serial data transfer

0x4 (DFS_05_BIT): 05-bit serial data transfer

0x5 (DFS_06_BIT): 06-bit serial data transfer

0x6 (DFS_07_BIT): 07-bit serial data transfer

0x7 (DFS_08_BIT): 08-bit serial data transfer

0x8 (DFS_09_BIT): 09-bit serial data transfer

0x9 (DFS_10_BIT): 10-bit serial data transfer

0xA (DFS_11_BIT): 11-bit serial data transfer

0xB (DFS_12_BIT): 12-bit serial data transfer

0xC (DFS_13_BIT): 13-bit serial data transfer

0xD (DFS_14_BIT): 14-bit serial data transfer

0xE (DFS_15_BIT): 15-bit serial data transfer

0xF (DFS_16_BIT): 16-bit serial data transfer

0x10 (DFS_17_BIT): 17-bit serial data transfer

0x11 (DFS_18_BIT): 18-bit serial data transfer

0x12 (DFS_19_BIT): 19-bit serial data transfer

0x13 (DFS_20_BIT): 20-bit serial data transfer

0x14 (DFS_21_BIT): 21-bit serial data transfer

0x15 (DFS_22_BIT): 22-bit serial data transfer

0x16 (DFS_23_BIT): 23-bit serial data transfer

0x17 (DFS_24_BIT): 24-bit serial data transfer

0x18 (DFS_25_BIT): 25-bit serial data transfer
0x19 (DFS_26_BIT): 26-bit serial data transfer
0x1A (DFS_27_BIT): 27-bit serial data transfer
0x1B (DFS_28_BIT): 28-bit serial data transfer
0x1C(DFS_29_BIT): 29-bit serial data transfer
0x1D(DFS_30_BIT): 30-bit serial data transfer
0x1E (DFS_31_BIT): 31-bit serial data transfer
0x1F (DFS_32_BIT): 32-bit serial data transfer

22.6.2. Control Register 1(CTRLR1)

This register exists only when the SSI is configured as a master device. When the SSI is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. Control register 1 controls the end of serial transfers when in RX-ONLY mode and TX-ONLY mode. It is impossible to write to this register when the SSI is enabled. The SSI is enabled and disabled by writing to the SSIENR register.

Offset Address: 0x0004

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	NDF							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	NDF							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-3: Control Register 1(CTRLR1)

NDF — Number of Data Frames.

When TMOD = 01 or TMOD = 10 or TMOD = 11, this register field sets the number of data frames to be continuously received or transmitted by the SSI. The SSI continues to receive or transmit serial data until the number of data frames is equal to this register value plus 1.

This register serves no purpose and is not present when the SSI is configured as a serial slave. When TMOD = 01, CLK_STRETCH_EN in SPI_CTRLR0 must be set.

22.6.3. SSI Enable Register(SSIENR)

Offset Address: 0x0008

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SSIC_EN
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-4: SSI Enable Register(SSIENR)

SSIC_EN — SSI Enable.

Enables and disables all SSI operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the SSI control registers when enabled. When disabled, the SSI sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.

1 = (ENABLED): Enables SSI

0 = (DISABLE): Disables SSI

22.6.4. Microwire Control Register(MWCR)

Offset Address: 0x000C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	MHS	MDD	MWMOD
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-5: Microwire Control Register(MWCR)

This register controls the direction of the data word for the half-duplex Microwire serial protocol. It is impossible to write to this register when the SSI is enabled. The SSI is enabled and disabled by writing to the SSIENR register.

MHS — Microwire Handshaking.

Relevant only when the SSI is configured as a serial-master device. When configured as a serial slave, this bit field has no functionality. Used to enable and disable the busy/ready handshaking interface for the Microwire protocol. When enabled, the SSI checks for a ready status from the target slave, after the transfer of the last data/control bit, before clearing the BUSY status in the SR register.

1 = (ENABLED): handshaking interface is enabled

0 = (DISABLE): handshaking interface is disabled

MDD — Microwire Control.

Defines the direction of the data word when the Microwire serial protocol is used. When this bit is set to 0, the data word is received by the SSI MacroCell from the external serial device. When this bit is set to 1, the data word is transmitted from the SSI MacroCell to the external serial device.

1 = (TRANSMIT): SSI transmits data

0 = (RECEIVE): SSI receives data

MWMOD — Microwire Transfer Mode.

Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there must be a control word for each data word that is transmitted or received.

1 = (SEQUENTIAL): Sequential Transfer

0 = (NON_SEQUENTIAL): Non-Sequential Transfer

22.6.5. Slave Enable Register(SER)

Offset Address: 0x0010

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SER
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-6: Slave Enable Register(SER)

This register is valid only when the SSI is configured as a master device. When the SSI is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. The register enables the individual slave select output lines from the SSI master. Up to 16 slave-select output pins are available on the SSI master. You cannot write to this register when SSI is busy and when SSIC_EN = 1.

SER — Slave Select Enable Flag.

1 = Selected

0 = Not Selected

22.6.6. Baud Rate Select(BAUDR)

Offset Address: 0x0014

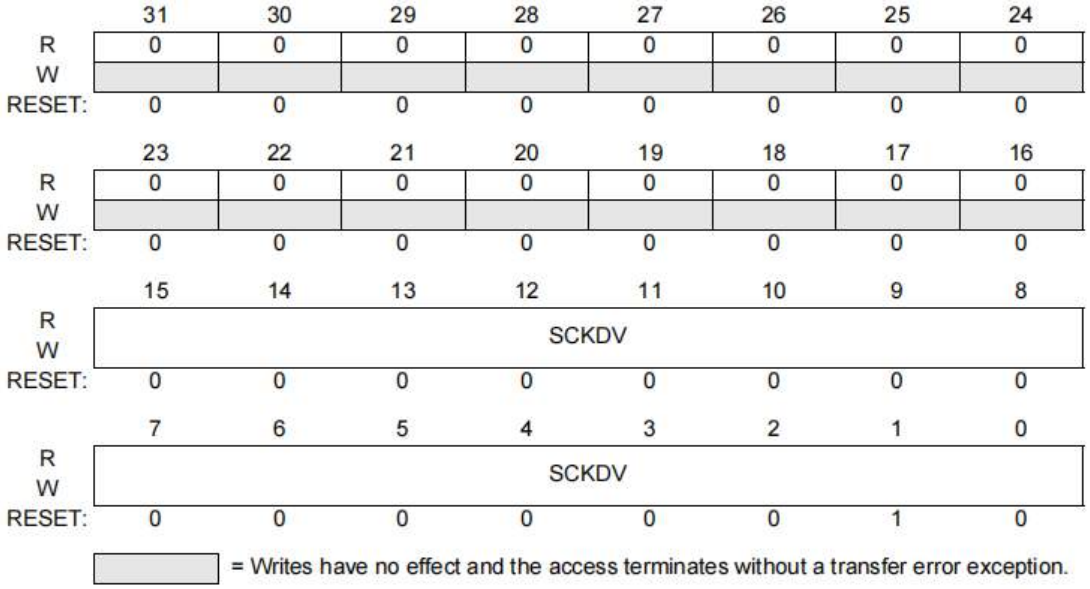


Figure 22-7: Baud Rate Select(BAUDR)

SCKDV — SSI Clock Divider.

The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (sclk_out) is disabled. The frequency of the sclk_out is derived from the following equation:

Fsclk_out = Fssi_clk/SCKDV

where SCKDV is any even value between 2 and 65534. For example: for Fssi_clk = 3.6864MHz and SCKDV =2 Fsclk_out = 3.6864/2 = 1.8432MHz

22.6.7. Transmit FIFO Threshold Level(TXFTLR)

Offset Address: 0x0018

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	TXFTHR				
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	TFT				
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-8: Transmit FIFO Threshold Level(TXFTLR)

This register controls the threshold value for the transmit FIFO memory. The SSI is enabled and disabled by writing to the SSIENR register.

TXFTHR — Transfer start FIFO level.

Used to control the level of entries in transmit FIFO above which transfer will start on serial line. This register can be used to ensure that sufficient data is present in transmit FIFO before starting a write operation on serial line. These field is valid only for Master mode of operation.

TFT — Transmit FIFO Threshold.

Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt. The FIFO depth is configurable in the range 8-256; this register is sized to the number of address bits needed to access the FIFO. If you attempt to set this value greater than or equal to the depth of the FIFO, this field is not written and retains its current value. When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.

22.6.8. Receive FIFO Threshold Level(RXFTLR)

Offset Address: 0x001C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	RFT				
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-9: Receive FIFO Threshold Level(RXFTLR)

This register controls the threshold value for the receive FIFO memory. The SSI is enabled and disabled by writing to the SSIENR register.

RFT — Receive FIFO Threshold.

Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt. This register is sized to the number of address bits needed to access the FIFO. If you attempt to set this value greater than the depth of the FIFO, this field is not written and retains its current value. When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered.

22.6.9. Transmit FIFO Level Register(TXFLR)

Offset Address: 0x0020

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	TXTFL					
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-10: Transmit FIFO Level Register(TXFLR)

TXFLR — Transmit FIFO Level.

Contains the number of valid data entries in the transmit FIFO.

22.6.10. Receive FIFO Level Register(RXFLR)

Offset Address: 0x0024

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	RXTFL					
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-11: Receive FIFO Level Register(RXFLR)

RXTFL — Receive FIFO Level.

Contains the number of valid data entries in the receive FIFO.

22.6.11. Status Register(SR)

Offset Address: 0x0028

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	reserved	TXE	RFF	RFNE	TFE	TFNF	BUSY
W								
RESET:	0	0	0	0	0	1	1	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-12: Status Register(SR)

TXE — Transmission Error.

Set if the transmit FIFO is empty when a transfer is started. This bit can be set only when the SSI is configured as a slave device. Data from the previous transmission is resent on the TXD line. This bit is cleared when read.

1 = (TX_ERROR): Transmission Error

0 = (NO_ERROR): No Error

RFF — Receive FIFO Full.

When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared.

1 = (FULL): Receive FIFO is full

0 = (NOT_FULL): Receive FIFO is not full

RFNE — Receive FIFO Not Empty.

Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO.

1 = (NOT_EMPTY): Receive FIFO is not empty

0 = (EMPTY): Receive FIFO is empty

TFE — Transmit FIFO Empty.

When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt.

1 = (EMPTY): Transmit FIFO is empty

0 = (NOT_EMPTY): Transmit FIFO is not empty

TFNF — Transmit FIFO Not Full.

Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full.

1 = (NOT_FULL): Tx FIFO is not Full

0 = Tx FIFO is full

BUSY — SSI Busy Flag.

When set, indicates that a serial transfer is in progress; when cleared indicates that the SSI is idle or disabled.

1 = (ACTIVE): SSI is actively transferring data

0 = (INACTIVE): SSI is idle or disabled

22.6.12. Interrupt Mask Register(IMR)

Offset Address: 0x002C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	XRXOIM	reserved	RXFIM	RXOIM	RXUIM	TXOIM	TXEIM
W								
RESET:	0	1	1	1	1	1	1	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-13: Interrupt Mask Register(IMR)

XRXOIM — XIP Receive FIFO Overflow Interrupt Mask

1 = (UNMASKED): ssi_xrxo_intr interrupt is not masked

0 = (MASKED): ssi_xrxo_intr interrupt is masked

RXFIM — Receive FIFO Full Interrupt Mask

1 = (UNMASKED): ssi_rxf_intr interrupt is not masked

0 = (MASKED): ssi_rxf_intr interrupt is masked

RXOIM — Receive FIFO Overflow Interrupt Mask

1 = (UNMASKED): ssi_rxo_intr interrupt is not masked

0 = (MASKED): ssi_rxo_intr interrupt is masked

RXUIM — Receive FIFO Underflow Interrupt Mask

1 = (UNMASKED): ssi_rxu_intr interrupt is not masked

0 = (MASKED): ssi_rxu_intr interrupt is masked

TXOIM — Transmit FIFO Overflow Interrupt Mask

1 = (UNMASKED): ssi_txo_intr interrupt is not masked

0 = (MASKED): ssi_txo_intr interrupt is masked

TXEIM — Transmit FIFO Empty Interrupt Mask

1 = (UNMASKED): ssi_txe_intr interrupt is not masked

0 = (MASKED): ssi_txe_intr interrupt is masked

22.6.13. Interrupt Status Register(ISR)

Offset Address: 0x0030

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	XRXOIS	reserved	RXFIS	RXOIS	RXUIS	TXOIS	TXEIS
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-14: Interrupt Status Register(ISR)

This register reports the status of the SSI interrupts after they have been masked.

XRXOIS — XIP Receive FIFO Overflow Interrupt Status

1 = (ACTIVE): ssi_xrxo_intr interrupt is active after masking

0 = (INACTIVE): ssi_xrxo_intr interrupt is not active after masking

RXFIS — Receive FIFO Full Interrupt Status

1 = (ACTIVE): ssi_rxf_intr interrupt is active after masking

0 = (INACTIVE): ssi_rxf_intr interrupt is not active after masking

RXOIS — Receive FIFO Overflow Interrupt Status

1 = (ACTIVE): ssi_rxo_intr interrupt is active after masking

0 = (INACTIVE): ssi_rxo_intr interrupt is not active after masking

RXUIS — Receive FIFO Underflow Interrupt Status

1 = (ACTIVE): ssi_rxu_intr interrupt is active after masking
0 = (INACTIVE): ssi_rxu_intr interrupt is not active after masking

TXOIS — Transmit FIFO Overflow Interrupt Status
1 = (ACTIVE): ssi_txo_intr interrupt is active after masking
0 = (INACTIVE): ssi_txo_intr interrupt is not active after masking

TXEIS — Transmit FIFO Empty Interrupt Status
1 = (ACTIVE): ssi_txe_intr interrupt is active after masking
0 = (INACTIVE): ssi_txe_intr interrupt is not active after masking

22.6.14. Raw Interrupt Status Register(RISR)

Offset Address: 0x0034

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	XRXOIR	reserved	RXFIR	RXOIR	RXUIR	TXOIR	TXEIR
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-15: Raw Interrupt Status Register(RISR)

XRXOIR — XIP Receive FIFO Overflow Raw Interrupt Status
1 = (ACTIVE): ssi_xrxo_intr interrupt is active prior to masking
0 = (INACTIVE): ssi_xrxo_intr interrupt is not active prior masking

RXFIR — Receive FIFO Full Raw Interrupt Status
1 = (ACTIVE): ssi_rxf_intr interrupt is active prior to masking
0 = (INACTIVE): ssi_rxf_intr interrupt is not active prior masking

RXOIR — Receive FIFO Overflow Raw Interrupt Status
1 = (ACTIVE): ssi_rxo_intr interrupt is active prior to masking
0 = (INACTIVE): ssi_rxo_intr interrupt is not active prior masking

RXUIR — Receive FIFO Underflow Raw Interrupt Status
1 = (ACTIVE): ssi_rxu_intr interrupt is active prior to masking
0 = (INACTIVE): ssi_rxu_intr interrupt is not active prior masking

TXOIS — Transmit FIFO Overflow Raw Interrupt Status

1 = (ACTIVE): ssi_txo_intr interrupt is active prior to masking

0 = (INACTIVE): ssi_txo_intr interrupt is not active prior masking

TXEIS — Transmit FIFO Empty Raw Interrupt Status

1 = (ACTIVE): ssi_txe_intr interrupt is active prior to masking

0 = (INACTIVE): ssi_txe_intr interrupt is not active prior masking

22.6.15. Transmit FIFO Overflow Interrupt Clear Registers.(TXOICR)

Offset Address: 0x0038

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	TXOICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-16: Transmit FIFO Overflow Interrupt Clear Registers.(TXOICR)

TXOICR — Clear Transmit FIFO Overflow Interrupt.

This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.

22.6.16. Receive FIFO Overflow Interrupt Clear Register(RXOICR)

Offset Address: 0x003C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RXOICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-17: Receive FIFO Overflow Interrupt Clear Register(RXOICR)

RXOICR — Clear Receive FIFO Overflow Interrupt.

This register reflects the status of the interrupt. A read from this register clears the ssi_rxo_intr interrupt; writing has no effect.

22.6.17. Receive FIFO Underflow Interrupt Clear Register(RXUICR)

Offset Address: 0x0040

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RXUICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-18: Receive FIFO Underflow Interrupt Clear Register(RXUICR)

RXUICR — Clear Receive FIFO Underflow Interrupt.

This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect.

22.6.18. Interrupt Clear Register(ICR)

Offset Address: 0x0048

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	ICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-19: Interrupt Clear Register(ICR)

ICR — Clear Interrupts.

This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the ssi_mst_intr interrupts. Writing to this register has no effect.

22.6.19. DMA Control Register(DMACR)

Offset Address: 0x004C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	TDMAE	RDMAE
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-20: DMA Control Register(DMACR)

This register is only valid when SSI is configured with a set of DMA Controller interface signals (SSIC_HAS_DMA = 1). When SSI is not configured for DMA operation, this register will not exist and writing to the register's address will have no effect; reading from this register address will return zero. The register is used to enable the DMA Controller interface operation.

TDMAE — Transmit DMA Enable

This bit enables/disables the transmit FIFO DMA channel.

1 = (ENABLED): Transmit DMA enabled

0 = (DISABLE): Transmit DMA disabled

RDMAE — Receive DMA Enable

This bit enables/disables the receive FIFO DMA channel.

1 = (ENABLED): Receive DMA enabled

0 = (DISABLE): Receive DMA disabled

22.6.20. DMA Transmit Data Level(DMATDLR)

Offset Address: 0x0050

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	DMATDL				
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-21: DMA Transmit Data Level(DMATDLR)

This register is only valid when the SSI is configured with a set of DMA interface signals (SSIC_HAS_DMA = 1). When SSI is not configured for DMA operation, this register will not exist and writing to its address will have no effect; reading from its address will return zero.

DMATDL — Transmit Data Level.

This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.

22.6.21. DMA Receive Data Level(DMARDLR)

Offset Address: 0x0054

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	DMARDL				
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-22: DMA Receive Data Level(DMARDLR)

This register is only valid when SSI is configured with a set of DMA interface signals (SSIC_HAS_DMA = 1). When SSI is not configured for DMA operation, this register will not exist and writing to its address will have no effect; reading from its address will return zero.

DMARDL — Receive Data Level.

This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1.

22.6.22. Identification Register(IDR)

Offset Address: 0x0058

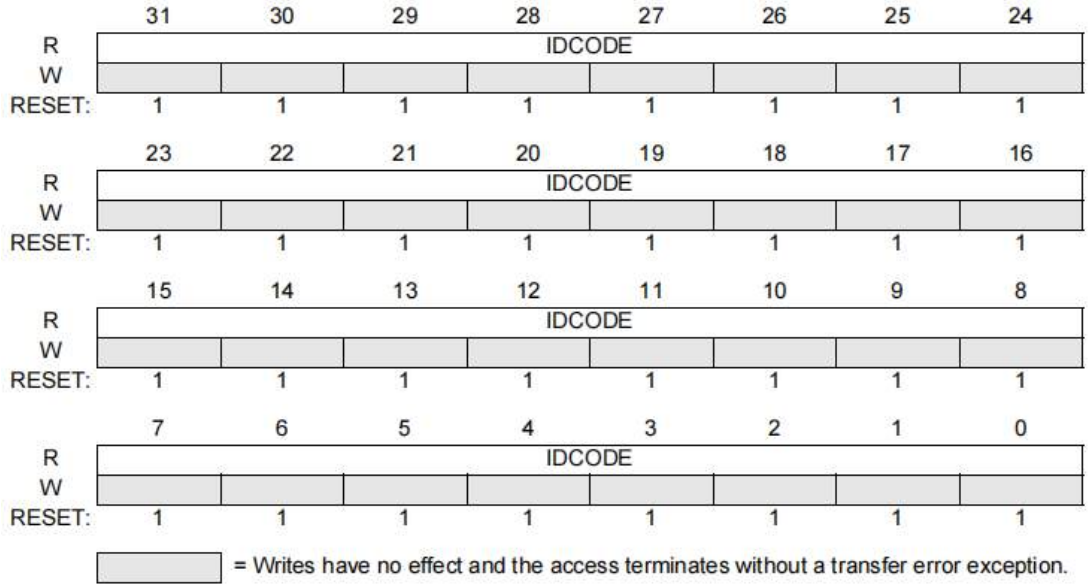


Figure 22-23: Identification Register(IDR)

IDCODE — Identification code.

The register contains the peripheral's identification code, which is written into the register at configuration time using Core Consultant.

22.6.23. Version ID Register(VIDR)

Offset Address: 0x005C

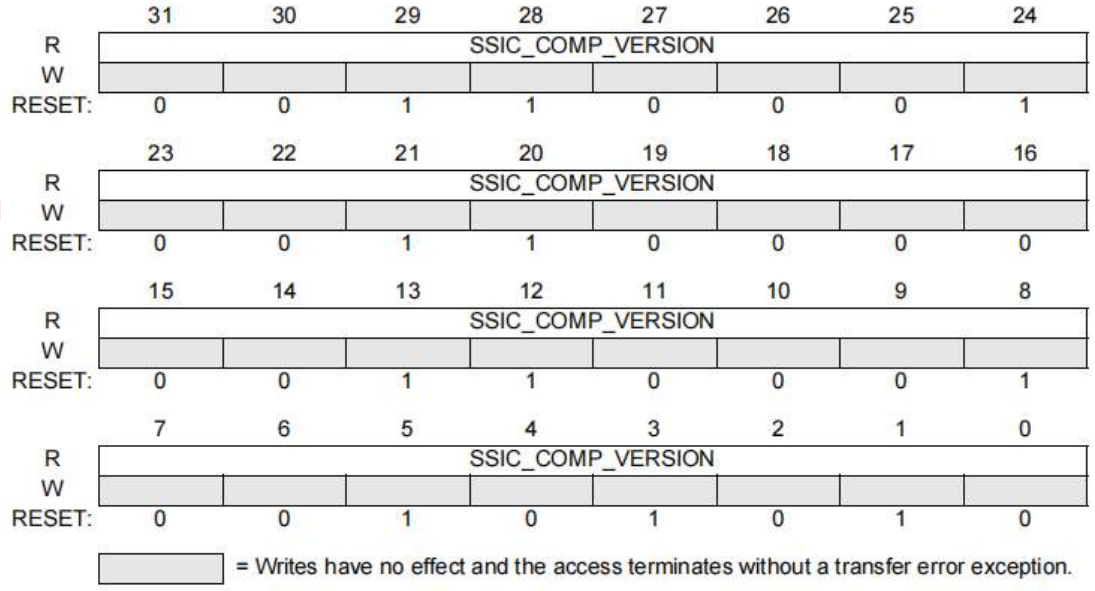


Figure 22-24: Version ID Register(VIDR)

SSIC_COMP_VERSION

Contains the hex representation of the Synopsys component version. Consists of ASCII value for each number in the version, followed by *. For example, 32_30_31_2A represents the version 2.01*.

22.6.24. SSI Data Register(DRx)

Offset Address: 0x0060

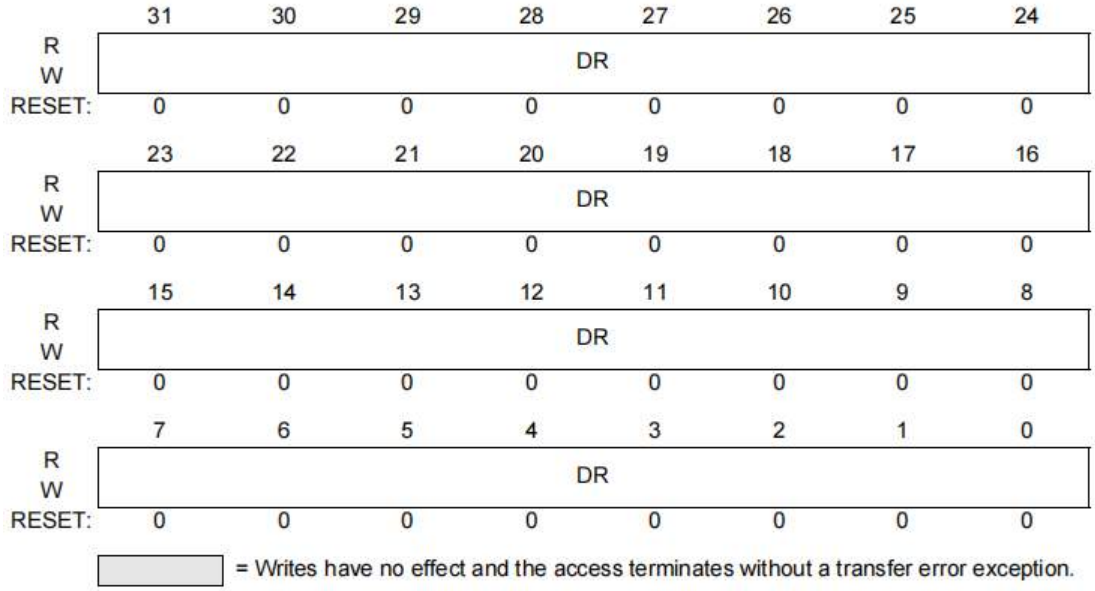


Figure 22-25: SSI Data Register(DRx)

The SSI data register is a 32bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSIC_EN = 1. FIFOs are reset when SSIC_EN = 0.

DR — Data Register.

When writing to this register, you must right-justify the data. Read data are automatically right-justified.

Read = Receive FIFO buffer

Write = Transmit FIFO buffer.

22.6.25. RX Sample Delay Register (RXSDR)

Offset Address: 0x00F0

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	SE
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	RSD							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-26: RX Sample Delay Register (RXSDR)

SE — Receive Data (RXD) Sampling Edge.

1 = negative edge of ssi_clk will be used to sample the incoming data

0 = posedge edge of ssi_clk will be used to sample the incoming data

RSD — Receive Data (RXD) Sample Delay.

This register is used to delay the sample of the RXD input port. Each value represents a single ssi_clk delay on the sample of RXD.

22.6.26. SPI Control Register 0(SPICTRLR0)

Offset Address: 0x00F4

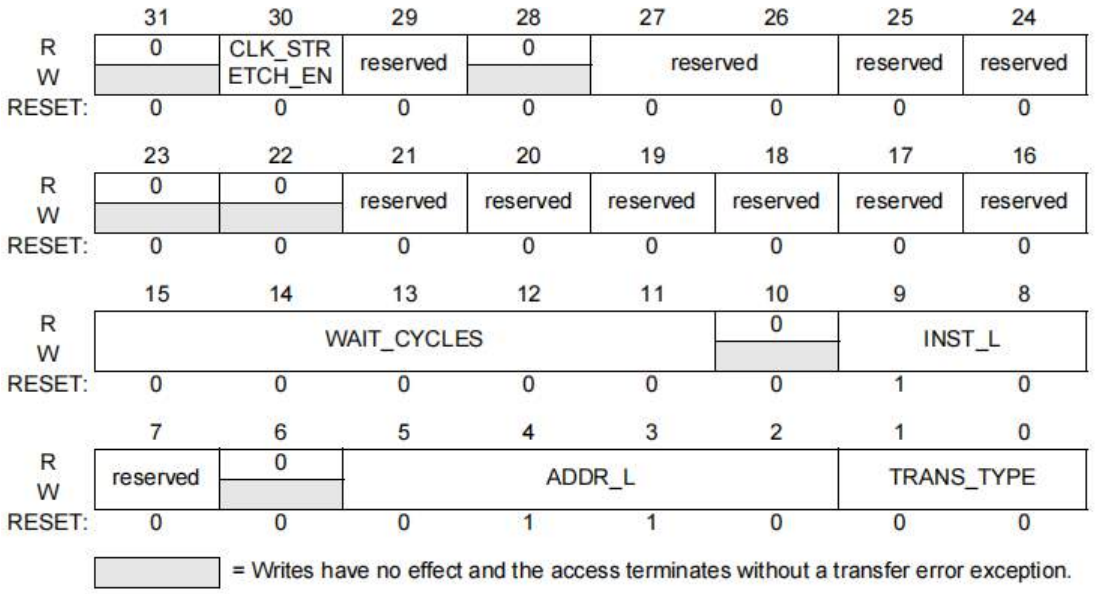


Figure 22-27: SPI Control Register 0(SPICTRLR0)

CLK_STRETCH_EN

Enables clock stretching capability in SPI transfers.

In case of write, if the FIFO becomes empty SSI will stretch the clock until FIFO has enough data to continue the transfer. In case of read, if the receive FIFO becomes full SSI will stop the clock until data has been read from the FIFO.

NOTE: *Recommend always to set this bit*

1 = CLK_STRETCH_ENABLE

0 = CLK_STRETCH_DISABLE

WAIT_CYCLES — Wait cycles in Dual/Quad/Octal mode between control frames transmit and data reception. Specified as number of SPI clock cycles.

INST_L — Dual/Quad/Octal mode instruction length in bits.

0x0 (INST_L0): No Instruction

0x1 (INST_L4): 4bit Instruction length

0x2 (INST_L8): 8bit Instruction length

0x3 (INST_L16): 16bit Instruction length

ADDR_L — Length of Address to be transmitted

0x0 (ADDR_L0): No Address

0x1 (ADDR_L4): 4bit Address length

0x2 (ADDR_L8): 8bit Address length

0x3 (ADDR_L12): 12bit Address length

0x4 (ADDR_L16): 16bit Address length

0x5 (ADDR_L20): 20bit Address length

0x6 (ADDR_L24): 24bit Address length

0x7 (ADDR_L28): 28bit Address length
0x8 (ADDR_L32): 32bit Address length
0x9 (ADDR_L36): 36bit Address length
0xA (ADDR_L40): 40bit Address length
0xB (ADDR_L44): 44bit Address length
0xC (ADDR_L48): 48bit Address length
0xD (ADDR_L52): 52bit Address length
0xE (ADDR_L56): 56bit Address length
0xF (ADDR_L60): 60bit Address length

TRANS_TYPE — Address and instruction transfer format.

Selects whether SSI will transmit instruction/address either in Standard SPI mode or the SPI mode selected in CTRLR0.SPI_FRF field.

0x0 (TT0): Instruction and Address will be sent in Standard SPI Mode.

0x1 (TT1): Instruction will be sent in Standard SPI Mode and Address will be sent in the mode specified by CTRLR0.SPI_FRF.

0x2 (TT2): Both Instruction and Address will be sent in the mode specified by SPI_FRF.

0x3 (TT3): Reserved.

22.6.27. XIP Mode Bits(XIPMBR)

Offset Address: 0x00FC

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	XIP_MD_BITS							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	XIP_MD_BITS							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-28: XIP Mode Bits(XIPMBR)

This register carries the mode bits which are sent in the XIP mode of operation after address phase. This is an 8bit register and can only be written when SSIENR register is set to 0.

XIP_MD_BITS

XIP mode bits to be sent after address phase of XIP transfer.

22.6.28. XIP Incr Inst Register(XIPIIR)

Offset Address: 0x0100

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	INCR_INST							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	INCR_INST							
W								
RESET:	0	1	1	0	1	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-29: XIP Incr Inst Register(XIPIIR)

This Register is valid only when SSIC_XIP_INST_EN is equal to 1. This register is used to store the instruction op-code to be used in INCR transactions when the same is requested on AHB interface. It is not possible to write to this register when the SSI is enabled (SSIC_EN=1). The SSI is enabled and disabled by writing to the SSIENR register.

INCR_INST — XIP INCR transfer opcode.

When SPI_CTRLR0.XIP_INST_EN bit is set to 1, SSI sends instruction for all XIP transfers, this register field stores the instruction op-code to be sent when an INCR type transfer is requested on AHB bus. The number of bits to be send in instruction phase is determined by SPI_CTRL0.INST_L field.

22.6.29. XIP Wrap Inst Register(XIPWIR)

Offset Address: 0x0104

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	WRAP_INST							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	WRAP_INST							
W								
RESET:	0	1	1	0	1	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-30: XIP Wrap Inst Register(XIPWIR)

This Register is valid only when SSIC_XIP_INST_EN is equal to 1. This register is used to store the instruction op-code to be used in WRAP transactions when the same is requested on AHB interface. It is not possible to write to this register when the SSI is enabled (SSIC_EN=1). The SSI is enabled and disabled by writing to the SSIENR register.

WRAP_INST — XIP WRAP transfer opcode.

When SPI_CTRLR0.XIP_INST_EN bit is set to 1, SSI sends instruction for all XIP transfers, this register field stores the instruction op-code to be sent when a WRAP type transfer is requested on AHB bus. The number of bits to be send in instruction phase is determined by SPI_CTRL0.INST_L field.

22.6.30. XIP Control Register(XIPCR)

Offset Address: 0x0108

	31	30	29	28	27	26	25	24
R	0	0	XIP_PREFETCH_EN	0	XIP_MBL		reserved	reserved
W								
RESET:	0	0	1	0	1	0	0	0

	23	22	21	20	19	18	17	16
R	CONT_XFER_EN	INST_EN	reserved	reserved	reserved	reserved	WAIT_CYCLES	
W								
RESET:	1	1	0	0	0	0	0	1

	15	14	13	12	11	10	9	8
R	WAIT_CYCLES			MD_BITS_EN	0	INST_L		0
W								
RESET:	0	0	0	0	0	1	0	0

	7	6	5	4	3	2	1	0
R	ADDR_L				TRANS_TYPE		FRF	
W								
RESET:	0	1	1	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-31: XIP Control Register(XIPCR)

This Register is valid only when SSIC_CONCURRENT_XIP_EN is equal to 1. This register is used to store the control information that the XIP transfer will be using in the concurrent mode.

XIP_PREFETCH_EN

1 = Enables XIP pre-fetch functionality in SSI.

0 = Disables XIP pre-fetch functionality in SSI.

XIP_MBL — XIP Mode bits length.

Sets the length of mode bits in XIP mode of operation. These bits are valid only when XIP_CTRL.XIP_MD_BIT_EN is set to 1.

0x0 (MBL_2): Mode bits length equal to 2

0x1 (MBL_4): Mode bits length equal to 4

0x2 (MBL_8): Mode bits length equal to 8

0x3 (MBL_16): Mode bits length equal to 16

CONT_XFER_EN

1 = Enable continuous transfer in XIP mode.

0 = Disables continuous transfer in XIP mode.

INST_EN

1 = XIP transfers will have instruction phase.

0 = XIP transfers will not have instruction phase.

WAIT_CYCLES — Wait cycles in Dual/Quad/Octal mode between control frames transmit and data reception. Specified as number of SPI clock cycles.

0x0: 0 wait cycles

0x1: 1 wait cycles

0x2: 2 wait cycles

….

….

0x1F: 31 wait cycles

MD_BITS_EN — Mode bits enable in XIP mode.

1 = insert mode bits after the address phase.

0 = no mode bits after the address phase.

INST_L

Dual/Quad/Octal mode instruction length in bits.

0x0 (INST_L0): No Instruction

0x1 (INST_L4): 4bit Instruction length

0x2 (INST_L8): 8bit Instruction length

0x3 (INST_L16): 16bit Instruction length

ADDR_L

This bit defines Length of Address to be transmitted. Only after this much bits are programmed in to the FIFO the transfer can begin.

0x0 (ADDR_L0): No Address

0x1 (ADDR_L4): 4bit Address length

0x2 (ADDR_L8): 8bit Address length

0x3 (ADDR_L12): 12bit Address length

0x4 (ADDR_L16): 16bit Address length

0x5 (ADDR_L20): 20bit Address length

0x6 (ADDR_L24): 24bit Address length

0x7 (ADDR_L28): 28bit Address length

0x8 (ADDR_L32): 32bit Address length

0x9 (ADDR_L36): 36bit Address length

0xA (ADDR_L40): 40bit Address length

0xB (ADDR_L44): 44bit Address length

0xC (ADDR_L48): 48bit Address length

0xD (ADDR_L52): 52bit Address length

0xE (ADDR_L56): 56bit Address length

0xF (ADDR_L60): 60bit Address length

TRANS_TYPE

Address and instruction transfer format.

Selects whether SSI will transmit instruction/address either in Standard SPI mode or the SPI mode selected in CTRLR0.SPI_FRF field.

0x0 (TT0): Instruction and Address will be sent in Standard SPI Mode.

0x1 (TT1): Instruction will be sent in Standard SPI Mode and Address will be sent in the mode specified by XIP_CTRL.SPI_FRF.

0x2 (TT2): Both Instruction and Address will be sent in the mode specified by XIP_CTRL.FRF.

0x3 (TT3): Reserved.

FRF — SPI Frame Format

Selects data frame format for Transmitting/Receiving the data.

0x0 (RSVD): Reserved

0x1 (SPI_DUAL): Dual SPI Format

0x2 (SPI_QUAD): Quad SPI Format

0x3 (SPI_OCTAL): Octal SPI Format

22.6.31. XIP Slave Enable Register(XIPSER)

Offset Address: 0x010C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SER
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-32: XIP Slave Enable Register(XIPSER)

This register is valid only when the SSIC_CONCURRENT_XIP_EN is equal to 1. The register enables the individual slave select output lines from the SSI master for

XIP mode of operation. Up to 16 slave-select output pins are available on the SSI master. You cannot write to this register when SSI is busy or when SSIC_EN = 1.

SER — Slave Select Enable Flag.

Each bit in this register corresponds to a slave select line (ss_x_n) from the SSI master. When a bit in this register is set (1), the corresponding slave select line from the master is activated when a XIP transfer begins. It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a XIP transfer is started. Before beginning a transfer, you should enable the bit in this register that corresponds to the slave device with which the master wants to communicate. When not operating in broadcast mode, only one bit in this field should be set.

1 = Select

0 = Not Select

22.6.32. XIP Receive FIFO Overflow Interrupt Clear Register(XRXIOCR)

Offset Address: 0x0110

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	XRXOICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-33: XIP Receive FIFO Overflow Interrupt Clear Register(XRXIOCR)

XRXOICR — Clear XIP Receive FIFO Overflow Interrupt.

This register reflects the status of the interrupt. A read from this register clears the ssi_xrxo_intr(_n) interrupt; writing has no effect.

22.6.33. XIP Continues Transfer Time Out Register(XIPCTTOR)

Offset Address: 0x0114

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	XTOC							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 22-34: XIP Continues Transfer Time Out Register(XIPCTTOR)

XIP count down register for continuous mode. The counter is used to de-select the slave during continuous transfer mode. It is not possible to write to this register when the SSI is enabled (SSIC_EN=1). The SSI is enabled and disabled by writing to the SSIENR register.

XTOC — XIP time out value in terms of hclk.

Once slave is selected in continuous XIP mode this counter will be used to de-select the slave if there is no request for the time specified in the counter.

22.7. Functional Description

22.7.1. Master Mode

This mode enables serial communication with serial-slave peripheral devices. When configured as a serial-master device, the SSI initiates and controls all serial transfers. Figure 31-35 shows an example of the SSI configured as a serial master with all other devices on the serial bus configured as serial slaves.

The serial bit-rate clock, generated and controlled by the SSI, is driven out on the sclk_out line. When the SSI is disabled (SSIC_EN = 0), no serial transfers can occur and sclk_out is held in "inactive" state, as defined by the serial protocol under which it operates.

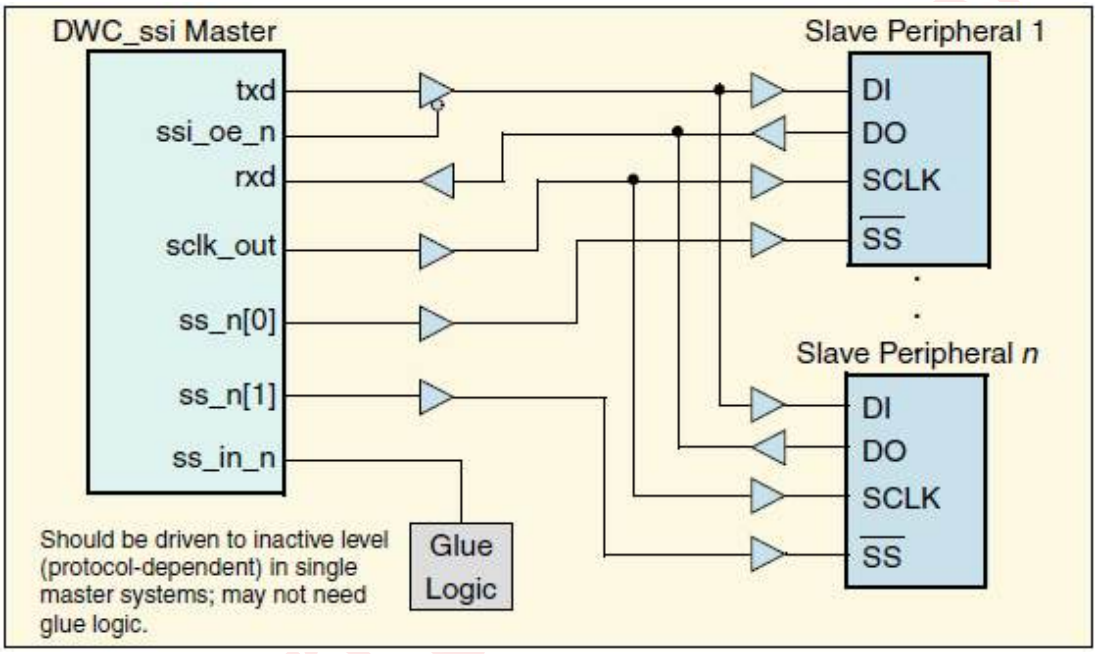


Figure 22-35: SSI Configured as Master Device

22.7.2. Clock Ratios

SSI works on an oversampling architecture. For the master mode of operation, the peripheral clock (sclk_out) period is a multiple of the internal core clock (ssi_clk).

When the SSI macrocell is configured as a master device, the maximum frequency of the bit-rate clock (sclk_out) is one-half the frequency of ssi_clk. This is to allow the shift control logic to capture data on one clock edge of sclk_out and propagate data on the opposite edge.

The frequency of sclk_out can be derived from the following equation.

$$F_{sclk\_out} = \frac{F_{ssi\_clk}}{SCKDV}$$

SCKDV is a programmable register holding any even value in the range 0 ~ 65,534. If SCKDV=0, sclk_out is disabled.

22.7.3. Receive and Transmit FIFO Buffers

The FIFO buffers used by the SSI are internal D-type flip-flops. The widths of both transmit and receive FIFO buffers are fixed at 32bits due to the serial specifications, which state that a serial transfer (data frame) can be 4 to 32bits in length. Data frames that are less than 32bits in size must be right-justified when written into the transmit FIFO buffer. The shift control logic automatically right-justifies receive data in the receive FIFO buffer.

22.7.3.1. Transmit FIFO

The transmit FIFO is loaded by AHB write commands to the SSI data register (DR). Data are popped (removed) from the transmit FIFO by the shift control logic into the transmit shift register. The transmit FIFO generates a FIFO empty interrupt request (ssi_txe_intr) when the number of entries in the FIFO is less than or equal to the FIFO threshold value. The threshold value, set through the programmable register TXFTLR, determines the level of FIFO entries at which an interrupt is generated. The threshold value allows you to provide early indication to the processor that the transmit FIFO is nearly empty. A transmit FIFO overflow interrupt (ssi_txo_intr) is generated if you attempt to write data into an already full transmit FIFO.

22.7.3.2. Receive FIFO

Data are popped from the receive FIFO by AHB read commands to the SSI data register (DR). The receive FIFO is loaded from the receive shift register by the shift control logic. The receive FIFO generates a FIFO-full interrupt request (ssi_rxf_intr) when the number of entries in the FIFO is greater than or equal to the FIFO threshold value plus 1. The threshold value, set through programmable register RXFTLR, determines the level of FIFO entries at which an interrupt is generated.

The threshold value allows you to provide early indication to the processor that the receive FIFO is nearly full. A receive FIFO overrun interrupt (ssi_rxo_intr) is generated when the receive shift logic attempts to load data into a completely full receive FIFO. However, this newly received data are lost. A receive FIFO underflow interrupt (ssi_rxu_intr) is generated if you attempt to read from an empty receive FIFO. This alerts the processor that the read data are invalid.

22.7.4. DMA Operation

The SSI has optional built-in DMA capability which can be selected at configuration time; it has a handshaking interface to a DMA Controller to request and control transfers. The AHB bus is used to perform the data transfer to or from the DMA. While the SSI DMA operation is designed in a generic way to fit any DMA controller as easily as possible, it is designed to work seamlessly, and best used. To enable the DMA Controller interface on the SSI, you must write the DMA Control Register (DMACR). Writing a 1 into the TDMAE bit field of DMACR register enables the SSI transmit handshaking interface. Writing a 1 into the RDMAE bit field of the DMACR register enables the SSI receive handshaking interface.

22.7.5. SSI Interrupts

The SSI interrupts are described as follows:

Transmit FIFO Empty Interrupt (ssi_txe_intr) – Set when the transmit FIFO is equal to or below its threshold value and requires service to prevent an under-run. The threshold value, set through a software-programmable register, determines the level of transmit FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are written into the transmit FIFO buffer, bringing it over the threshold level.

Transmit FIFO Overflow Interrupt (ssi_txo_intr) – Set when an AHB access attempts to write into the transmit FIFO after it has been completely filled. When set, data written from the AHB is discarded. This interrupt remains set until you read the transmit FIFO overflow interrupt clear register (TXOICR).

Receive FIFO Full Interrupt (ssi_rxf_intr) – Set when the receive FIFO is equal to or above its threshold value plus 1 and requires service to prevent an overflow. The threshold value, set through a software-programmable register, determines the level of receive FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are read from the receive FIFO buffer, bringing it below the threshold level.

Receive FIFO Overflow Interrupt (ssi_rxo_intr) – Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. When set, newly received data are discarded. This interrupt remains set until you read the receive FIFO overflow interrupt clear register (RXOICR).

Receive FIFO Underflow Interrupt (ssi_rxu_intr) – Set when an AHB access attempts to read from the receive FIFO when it is empty. When set, zeros are read back from the receive FIFO. This interrupt remains set until you read the receive FIFO underflow interrupt clear register (RXUICR).

Combined Interrupt Request (ssi_intr) – OR'ed result of all the above interrupt requests after masking. To mask this interrupt signal, you must mask all other SSI interrupt requests.

22.7.6. Enhanced SPI Modes

SSI supports the dual, quad, and octal modes of SPI using the SSIC_SPI_MODE configuration parameter. The possible values for this parameter are Standard, Dual SPI, Quad SPI and Octal SPI modes. When dual, quad, or octal mode is selected for this parameter, the width of TXD, RXD and ssi_oe_n signals change to 2, 4, or 8, respectively. Hence, the data is shifted out/in on more than one line, increasing the overall throughput. Dual SPI, Quad or Octal SPI modes function similarly except for the width of TXD, RXD and ssi_oe_n signals. The mode of operation (write/read) can be selected using the CTRLR0.TMOD field.

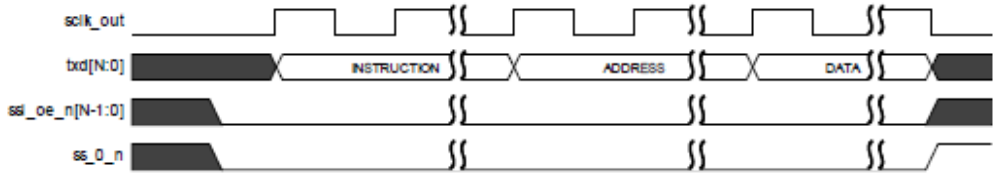


Figure 22-36: Typical Write Operation Dual/Quad/Octal SPI Mode

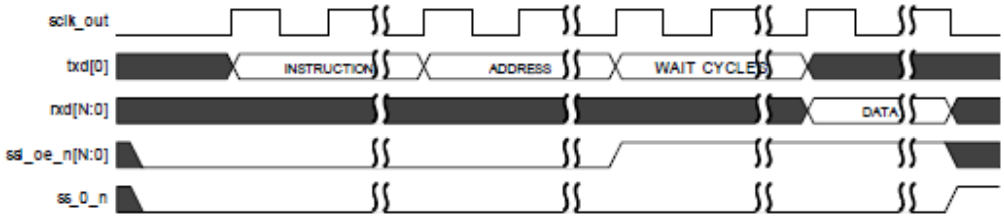


Figure 22-37: Typical Read Operation Dual/Quad/Octal SPI Mode

22.7.7. Execute In Place (XIP) Mode

SSI provides a function to directly perform memory read operation from AHB transaction. This is called execute in place mode, in which SSI acts as memory mapped interface to an SPI memory. The XIP mode can be enabled in SSI by selecting the configuration parameter SSIC_XIP_EN. This includes an extra sideband signal xip_en on an AHB interface. This signal level decides if the AHB transfers are register read-write or XIP reads. Only AHB READs are supported during XIP operation. If *xip_en* signal is driven to 1, then SSI expects a read request to be made on the AHB interface. This request is translated to SPI read on the serial interface. As soon as the data is received, it is returned to AHB interface in same transaction. *haddr* is used to derive the address to be sent on the SPI interface. Certain devices expect the instruction phase to exist during XIP transfers. SSI supports inclusion of some fixed instruction sets during the XIP mode of operation.

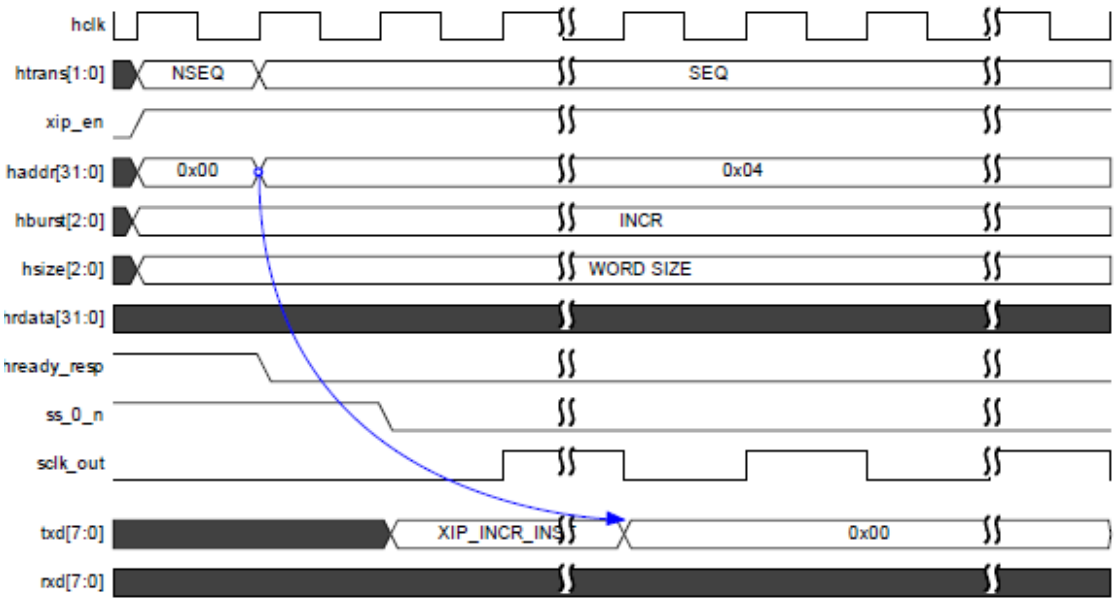


Figure 22-38: XIP Transfer with Instruction Phase

22.7.8. Continuous Transfer Mode in XIP

When SSI receives an XIP request, address from the AHB interface is transmitted onto the SPI interface directly. Each new transfer (XIP Read) on the AHB interface is treated in the same manner. Therefore, for every request, a new address must be sent to the device thereby contributing to the latency of the system.

If a memory device allows the stretching of slave select signal in between the XIP read transfers, then SSI can be programmed for continuous XIP mode to achieve higher performance. In this mode, the host fuses two or more AHB burst requests into a single SPI command by ensuring that the command and address are not retransmitted and the host controller need not wait for any dummy cycles in between these bursts.

When this function is enabled, then SSI functions in continuous XIP mode as soon as first XIP command is received. For the first XIP transfer, the address is sent on the SPI interface. After reception of requested data, SSI continues to keep the slave selected and the clock (sclk_out) remains in the default state. For subsequent XIP transfers on the AHB interface, SSI resumes the clock (sclk_out) and neither the command nor the address is transmitted onto the SPI interface and the data to be fetched from the device immediately (no dummy cycles).

- *An undefined INCR (hburst = 001) burst is not supported in continuous read mode.*

During the continuous transfer, a lot of power is dissipated on the slave device since the slave is selected all the times. To avoid such condition, SSI provides a configuration option to enable a watchdog timer to de-select the slave after the counter runs out.

SSI can de-select the slave under the following conditions:

- *A non-XIP command is received on an XIP interface (effectively any AHB transaction with xip_en driven to 0).*
- *When the AHB transaction is to a non-consecutive address, the slave select is removed and then SSI initiates a new XIP request.*
- *SSI does not detect any XIP transfer on AHB interface for the time-period specified in XIP_CNT_TIME_OUT register.*

22.7.9. Data Pre-fetch in XIP Operations

Using the data pre-fetch feature in SSI, the controller pre-fetches the data for the successive burst during the current XIP transaction. If the next transaction request is made to the successive address, the data can be read directly from the RX FIFO instead of waiting for a new address and data to be sent to the device. This improves the overall performance of the system.

The amount of the data to be pre-fetched should be equal to the burst length of the last AHB request or the FIFO depth (whichever is lower). For example, if AHB defines a burst length of 16 starting from address 0x00, then SSI fetches 16 beats to complete the current transfer, and then again 16 more data frames will be fetched and kept into the data register. If AHB bus again requests for data starting from end address for last transfer, then the rest of the data will be sent to the device from RX FIFO itself. In parallel, SSI again starts an XIP transfer to the device to pre-fetch the next chunk of data. If AHB master does not define the contiguous address, then current data is flushed out from the FIFO and the controller starts a fresh transaction.

When SSI completes the current burst, and is pre-fetching the data for the next burst, a new XIP request may be placed. In this case, SSI can terminate the current transfer, or increase the number to data beats to be fetched depending on the address.

- *If XIP pre-fetching is enabled, AHB request for incremental transfer of undefined length (hburst = 3'b001) is not allowed.*

23. Serial Peripheral Interface Module (SPI)

23.1. Introduction

The serial peripheral interface (SPI) module allows full-duplex, synchronous, serial communication between the microcontroller unit(MCU) and peripheral devices. Software can poll the SPI status flags or SPI operation can be interrupt driven.

23.2. Features

Features include:

- Master mode and slave mode
- Wired-OR mode
- Slave-select output
- Mode fault error flag with central processor unit (CPU) interrupt capability
- Control of SPI operation during doze mode
- Reduced drive control for lower power consumption
- Programmable interface operation for Freescale SPI or Texas Instruments synchronous serial interfaces
- Separate transmit and receive FIFOs, each 8bits wide and 8 locations deep
- Programmable data frame size from 4 to 16bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- Efficient transfers with DMA interface
- Visibility into TX and RX FIFOs for ease of debugging
- High Speed Mode for transfer timing adjustment

23.3. Modes of Operation

The SPI functions in these three modes:

1. Run mode — Run mode is the normal mode of operation.
2. Doze mode — Doze mode is a configurable low-power mode.
3. Stop mode — The SPI is inactive in stop mode.

23.4. Block Diagram

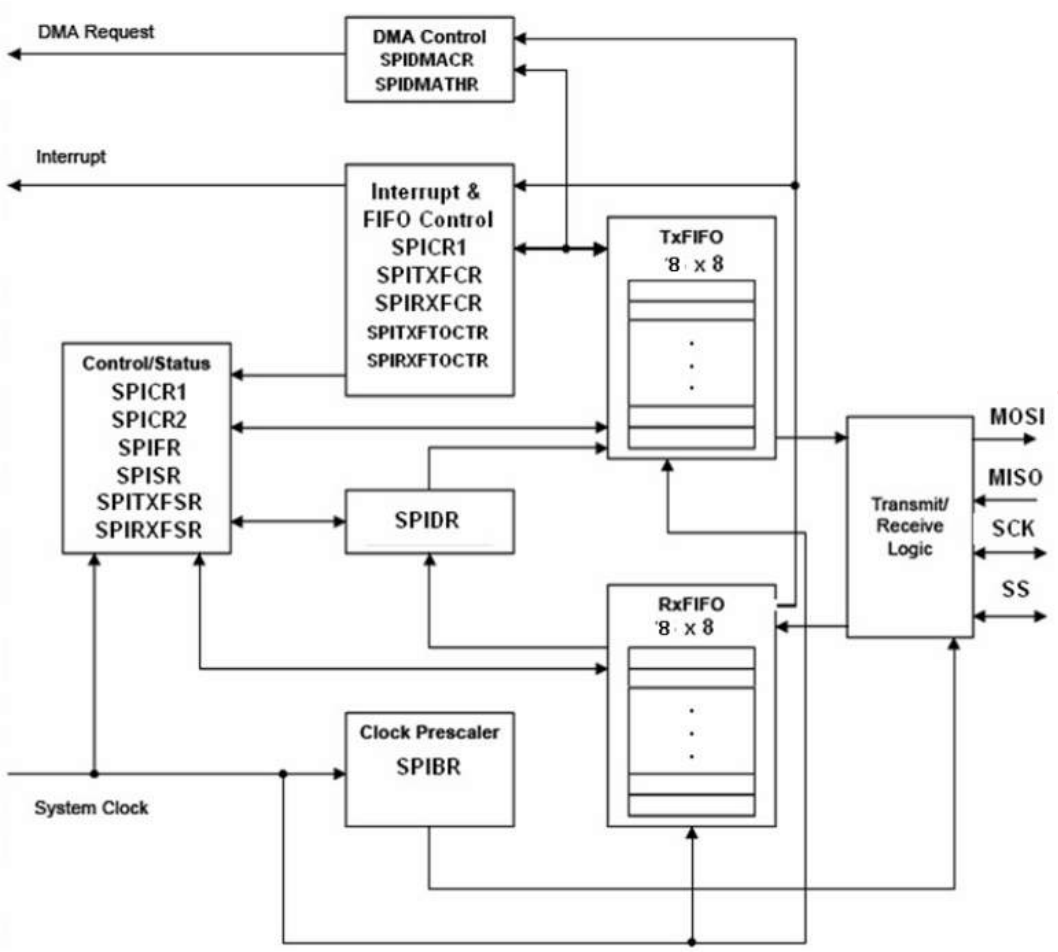


Figure 23-1: SPI Block Diagram

23.5. Signal Description

An overview of the signals is provided in **Table 23-1**.

Table 23-1: Signal Properties

Name	Port	Function[1]	Reset State
MISO	SPIPORT[0]	Master Data In / Slave Data Out	0
MOSI	SPIPORT[1]	Master Data Out / Slave Data In	0
SCK	SPIPORT[2]	Serial Clock	0
SS#	SPIPORT[3]	Slave Select	0

NOTE: *The specific SPI ports (MISO,MOSI,SCK,SS#) are GP I/O ports when the SPI is disabled (SPE=0).*

23.5.1. MISO (Master In/Slave Out)

MISO is one of the two SPI data pins.

- In master mode, MISO is the data input.
- In slave mode, MISO is the data output and is three-stated until a master drives the SS input pin low.
- In bidirectional mode, a slave MISO pin is the SISO pin (slave in/slave out).
- In a multiple-master system, all MISO pins are tied together.

23.5.2. MOSI (Master Out/Slave In)

MOSI is one of the two SPI data pins.

- In master mode, MOSI is the data output.
- In slave mode, MOSI is the data input.
- In bidirectional mode, a master MOSI pin is the MOMI pin (master out/master in).
- In a multiple-master system, all MOSI pins are tied together.

23.5.3. SCK (Serial Clock)

The SCK pin is the serial clock pin for synchronizing transmissions between master and slave devices.

- In master mode, SCK is an output.
- In slave mode, SCK is an input.
- In a multiple-master system, all SCK pins are tied together.

23.5.4. SS (Slave Select)

In master mode, the SS pin can be:

- A mode-fault input
- A general-purpose input
- A general-purpose output
- A slave-select output

In slave mode, the SS pin is always a slave-select input.

23.6. Memory Map and Registers

Table 23-2 shows the SPI memory map.

Table 23-2: SPI Memory Map

Offset Address	Bits 7 ~ 0	Access
0x0000	SPI Control Register 1 (SPICR1)	S/U
0x0001	SPI Control Register 2 (SPICR2)	S/U
0x0002	SPI Baud Rate Register (SPIBR)	S/U
0x0003	SPI Frame Register (SPIFR)	S/U
0x0004	SPI RXFIFO Control Register(SPIRXFCR)	S/U
0x0005	SPI TXFIFO Control Register(SPITXFCR)	S/U
0x0006	SPI RX FIFO Timeout Counter Register(SPIRXFTOCTR)	S/U
0x0007	SPI TX FIFO Timeout Counter Register(SPITXFTOCTR)	S/U
0x0008	SPI Port Data Direction Register (SPIDDR)	S/U
0x0009	SPI Pullup and Reduced Drive Register (SPIPURD)	S/U
0x000A	SPI After SCK Delay Register (SPIASCDR)	S/U
0x000B	SPI Before SCK Delay Register (SPIBSCDR)	S/U
0x000C	SPI Port Data Register (SPIPORT)	S/U
0x000D ~ 0x000F	SPI Transmit Counter Register (SPITCNT)	S/U
0x0010	SPI Data Register (SPIDR)	S/U
0x0014	SPI Status Register(SPISR)	S/U
0x0016	SPI RX FIFO Status Register(SPIRXFSR)	S/U
0x0017	SPI TX FIFO Status Register(SPITXFSR)	S/U
0x0018	SPI DMA Control Register(SPIDMACR)	S/U
0x0019	SPI DMA Threshold Register(SPIDMATHR)	S/U
0x001A	SPI FIFO Debug Control Register(SPIFDCR)	S/U
0x001B	SPI Interrupt Control Register(SPIICR)	S/U
0x001C	SPI RX FIFO Debug Register (SPIRXFDBGR)	S/U
0x001E	SPI TX FIFO Debug Register (SPITXFDBGR)	S/U

23.6.1. SPI Control Register

Offset Address: 0x0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBFE
RESET:	0	0	0	0	0	1	0	0

Figure 23-2: SPI Control Register 1 (SPICR1)

Read: Anytime

Write: Anytime

SPIE — SPI Interrupt Enable Bit

The SPIE bit enables the EOTF flag to generate interrupt requests. Reset clears SPIE.

1 = EOTF interrupt requests enabled

0 = EOTF interrupt requests disabled

SPE — SPI System Enable Bit

The SPE bit enables the SPI and dedicates SPI port pins [3:0] to SPI functions. When SPE is clear, the SPI system is initialized but in a low-power disabled state. Reset clears SPE.

1 = SPI enabled

0 = SPI disabled

SWOM — SPI Wired-OR Mode Bit

The SWOM bit configures the output buffers of SPI port pins [3:0] as open-drain outputs. SWOM controls SPI port pins [3:0] whether they are SPI outputs or general-purpose outputs. Reset clears SWOM.

1 = Output buffers of SPI port pins [3:0] open-drain

0 = Output buffers of SPI port pins [3:0] CMOS drive

NOTE: *The SWOM bit has no effect in this part.*

MSTR — Master Bit

The MSTR bit selects SPI master mode or SPI slave mode operation. Reset clears MSTR.

1 = Master mode

0 = Slave mode

CPOL — Clock Polarity Bit

The CPOL bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. Reset clears CPOL.

1 = Active-low clock; SCK idles high

0 = Active-high clock; SCK idles low

CPHA — Clock Phase Bit

The CPHA bit delays the first edge of the SCK clock. Reset sets CPHA.

1 = First SCK edge at start of transmission

0 = First SCK edge 1/2 cycle after start of transmission

NOTE: *Any value change of CPOL or CPHA during a transmission(when SS is low) may cause spurious results. Please change CPOL or CPHA value before a transmission (when*

SS is high) is to be launched.

SSOE — Slave Select Output Enable Bit

The SSOE bit and the DDRSP3 bit configure the SS pin as a general-purpose input or a slave-select output. Reset clears SSOE.

Table 23-3: SS Pin I/O Configurations

DDRSP3	SSOE	Master Mode	Slave Mode
0	0	Mode-fault input	Slave-select input
0	1	General-purpose input	Slave-select input
1	0	General-purpose output	Slave-select input
1	1	Slave-select output	Slave-select input

NOTE: *Setting the SSOE bit disables the mode fault detect function.*

LSBFE — LSB-First Enable Bit

The LSBFE enables data to be transmitted LSB first. Reset clears LSBFE.

1 = Data transmitted LSB first.

0 = Data transmitted MSB first

23.6.2. SPI Control Register 2

Offset Address: 0x0001

	Bit7	6	5	4	3	2	1	Bit0
Read: Write:	GT[5:0]						SPISDOZ	SPC0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 23-3: SPI Control Register 2 (SPICR2)

Read: Anytime

Write: Anytime;

GT — Guard Time Bits

$GT = (GT[5:3]+1)*2^{(GT[2:0]+1)}$

SPISDOZ — SPI Stop in Doze Bit

The SPIDOZ bit stops the SPI clocks when the CPU is in doze mode.

1 = SPI inactive in doze mode

0 = SPI active in doze mode

SPC0 — Serial Pin Control Bit 0

The SPC0 bit enables the bidirectional pin configurations shown in **Table 23-4**.

Table 23-4: Bidirectional Pin Configurations

	Pin Mode	SPC0	MSTR	MISO Pin[1]	MOSI Pin[2]	SCK Pin[3]	SS Pin[4]
A	Normal	0	0	Slave Data Output	Slave Data Input	SCK Input	Slave-Select Input
B			1	Master Data Input	Master Data Output	SCK Output	MODF/GP Input (DDRSP3 = 0) Or GP Output (DDRSP3 = 1)
C	Bidirectional	1	0	Slave Data I/O	GP[5] I/O	SCK Input	Slave-Select Input
D			1	Gp I/O	Master Data I/O	Sck Output	MODF/GP Input (DDRSP3 = 0) Or GP Output (DDRSP3 = 1)

NOTES:

1. *Slave output is enabled if SPIDDR bit 0 = 1, SS = 0, and MSTR = 0 (A, C).*
2. *Master output is enabled if SPIDDR bit 1 = 1 and MSTR = 1 (B, D).*
3. *SCK output is enabled if SPIDDR bit 2 = 1 and MSTR = 1 (B, D).*
4. *SS output is enabled if SPIDDR bit 3 = 1, SPICR1 bit 1 (SSOE) = 1, and MSTR = 1 (B, D). MODF input is enabled if SPI DDR bit 3 = 0 and SSOE = 0. GP input is enabled if SPI DDR bit 3 = 0 and SSOE = 1.*
5. *GP = General-purpose*

23.6.3. SPI Baud Rate Register

Offset Address: 0x0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	SPPR6	SPPR5	SPPR4	0	SPR2	SPR1	SPR0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 23-4: SPI Baud Rate Register (SPIBR)

Read: Anytime

Write: Anytime; writing to unimplemented bits has no effect

SPPR[6:4] — SPI Baud Rate Preselection Bits

The SPPR[6:4] and SPR[2:0] bits select the SPI clock divisor as shown in **Table 23-5**. Reset clears SPPR[6:4] and SPR[2:0], selecting an SPI clock divisor of 2.

SPR[2:0] — SPI Baud Rate Bits

The SPPR[6:4] and SPR[2:0] bits select the SPI clock divisor as shown in **Table 23-5**. Reset clears SPPR[6:4] and SPR[2:0], selecting an SPI clock divisor of 2.

NOTE: *Writing to SPIBR during a transmission may cause spurious results.*

Table 23-5: SPI Baud Rate Selection (10-MHz Module Clock)

SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate	SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate
000	000	2	5 MHz	100	000	10	1 MHz
000	001	4	2.5 MHz	100	001	20	0.5 MHz
000	010	8	1.25 MHz	100	010	40	0.25 MHz
000	011	16	0.625 MHz	100	011	80	125 kHz
000	100	32	0.31 MHz	100	100	160	62.5 kHz
000	101	64	156.25 kHz	100	101	320	31.25 kHz
000	110	128	78.125 kHz	100	110	640	15.625 kHz
000	111	256	39.06 kHz	100	111	1280	7.81 kHz
001	000	4	2.5 MHz	101	000	12	833.33 kHz

Table 23-6: SPI Baud Rate Selection (10-MHz Module Clock)

SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate	SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate
001	001	8	1.25 MHz	101	001	24	416.67 kHz
001	010	16	0.625 MHz	101	010	48	208.33 kHz
001	011	32	0.31 MHz	101	011	96	104.17 kHz
001	100	64	156.25 kHz	101	100	192	52.08 kHz
001	101	128	78.125 kHz	101	101	384	26.04 kHz
001	110	256	39.06 kHz	101	110	768	13.02 kHz
001	111	512	19.53 kHz	101	111	1536	6.51 kHz
010	000	6	1.67 MHz	110	000	14	714.29 kHz
010	001	12	0.83 MHz	110	001	28	357.14 kHz
010	010	24	0.42 MHz	110	010	56	178.57 kHz
010	011	48	208.33 kHz	110	011	112	89.29 kHz
010	100	96	104.17 kHz	110	100	224	44.64 kHz
010	101	192	52.08 kHz	110	101	448	22.32 kHz
010	110	384	26.04 kHz	110	110	896	11.16 kHz
010	111	768	13.02 kHz	110	111	1792	5.58 kHz
011	000	8	1.25 MHz	111	000	16	0.625 MHz
011	001	16	0.625 MHz	111	001	32	0.31 MHz
011	010	32	0.31 MHz	111	010	64	156.25 kHz
011	011	64	156.25 kHz	111	011	128	78.125 kHz

SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate	SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate
011	100	128	78.125 kHz	111	100	256	39.06 kHz
011	101	256	39.06 kHz	111	101	512	19.53 kHz
011	110	512	19.53 kHz	111	110	1024	9.77 kHz
011	111	1024	9.77 kHz	111	111	2048	4.88 kHz

23.6.4. SPI Frame Register

Offset Address: 0x0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	CONT	GTE	LBM	FFSEL	FMSZ3	FMSZ2	FMSZ1	FMSZ0
RESET:	0	1	0	0	0	1	1	1

Figure 23-5: SPI Frame Register (SPIFR)

Read: Anytime

Write: Anytime

CONT — Continuous peripheral chip select enable

1 = Keep peripheral chip select signal low between transfers until EOTF is set

0 = Return peripheral chip select signal to high between transfers

GTE — Guard Time Enable

1 = Guard Time is Enabled

0 = Guard Time is Disabled

LBM — Loop Back Mode

1 = Loop Back Mode

0 = Normal Mode

FFSEL — Frame Format Select

1 = TI frame format is selected

0 = FREESCALE frame format is selected

FMSZ[3:0] — Frame Size

The FMSZ[3:0] bits control the frame data length from 4 to 16bits. 0x3 sets the frame length to 4bits and 0xF sets it to 16bits. meanwhile case 0x0~0x2 will set the frame to 4bit long automatically.

23.6.5. SPI RX FIFO Control Register

Offset Address: 0x0004

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RXFCLR	RXFOVIE	RXFUDIE	RXFSTHIE	0	RXFSTH2	RXFSTH1	RXFSTH0
RESET:	0	0	0	0	0	0	0	0

Figure 23-6: SPI RX FIFO Control Register (SPIRXFCR)

Read: Anytime

Write: Anytime

RXFCLR — RX FIFO Clear

Write 1 to the bit will reset RX FIFO.

RXFOVIE — RX FIFO Overflow Interrupt Enable

1 = RX FIFO Overflow Interrupt Enabled

0 = RX FIFO Overflow Interrupt Disabled

RXFUDIE — RX FIFO Underflow Interrupt Enable

1 = RX FIFO Underflow Interrupt Enabled

0 = RX FIFO Underflow Interrupt Disabled

RXFSTHIE — RX FIFO Service Threshold Interrupt Enable

1 = RX FIFO Service Threshold Interrupt Enabled

0 = RX FIFO Service Threshold Interrupt Disabled

RXFSTH[2:0] — RX FIFO Service Threshold

Once effective data number in RX FIFO is above or equal the threshold specified, RX FIFO Service Flag asserted.

data number = RXFSTH[2:0] + 1

23.6.6. SPI TX FIFO Control Register

Offset Address: 0x0005

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TXFCLR	TXFOVIE	TXFUDIE	TXFSTHIE	0	TXFSTH2	TXFSTH1	TXFSTH0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 23-7: SPI TX FIFO Control Register (SPITXFCR)

Read: Anytime

Write: Anytime

TXFCLR — TX FIFO Clear
Write 1 to the bit will reset TXFIFO

TXFOVIE — TX FIFO Overflow Interrupt Enable
1 = TX FIFO Overflow Interrupt Enabled
0 = TX FIFO Overflow Interrupt Disabled

TXFUDIE — TX FIFO Underflow Interrupt Enable
1 = TX FIFO Underflow Interrupt Enabled
0 = TX FIFO Underflow Interrupt Disabled

TXFSTHIE — TX FIFO Service Threshold Interrupt Enable
1 = TX FIFO Service Threshold Interrupt Enabled
0 = TX FIFO Service Threshold Interrupt Disabled

TXFSTH[2:0] — TX FIFO Service Threshold

Once effective data number in TX FIFO is below or equal the threshold specified, TX FIFO Service Flag asserted.

data number = TXFSTH[2:0]

23.6.7. SPI RX FIFO TimeOut Counter Register

Offset Address: 0x0006

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RXFTOIE	RXFTOE	5	4	3	2	1	0
RESET:	0	0	1	0	0	0	0	0

Figure 23-8: SPI RX FIFO TimeOut Counter Register (SPIRXFTOCTR)

Read: Anytime;

Write: Anytime;

SPIRXFTOCTR[5:0] sets the SPI RX FIFO Timeout counter number. Once RX FIFO is not empty, the counter is on. If there is no operation to RX FIFO till the counter counts down to 0, the RXF_TIMEOUT flag in SPISR will set.

RXFTOIE — RX FIFO TimeOut Interrupt Enable
1 = RX FIFO TimeOut Interrupt Enabled
0 = RX FIFO TimeOut Interrupt Disabled

RXFTOE — RX FIFO TimeOut Function Enable
1 = RX FIFO TimeOut Function Enabled
0 = RX FIFO TimeOut Function Disabled

23.6.8. SPI TX FIFO TimeOut Counter Register

Offset Address: 0x0007

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TXFTOIE	TXFTOE	5	4	3	2	1	0
RESET:	0	0	1	0	0	0	0	0

Figure 23-9: SPI TX FIFO TimeOut Counter Register (SPITXFTOCTR)

Read: Anytime;

Write: Anytime;

SPITXFTOCTR[5:0] sets the SPI TX FIFO Timeout counter number. Once TX FIFO is not empty, the counter is on. If there is no operation to TX FIFO till the counter counts down to 0,the TXF_TIMEOUT flag in SPISR will set.

TXFTOIE — TX FIFO TimeOut Interrupt Enable

1 = TX FIFO TimeOut Interrupt Enabled

0 = TX FIFO TimeOut Interrupt Disabled

TXFTOE — TX FIFO TimeOut Function Enable

1 = TX FIFO TimeOut Function Enabled

0 = TX FIFO TimeOut Function Disabled

23.6.9. SPI Port Data Direction Register

Offset Address: 0x0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RSVD7	RSVD6	RSVD5	RSVD4	DDRSP3	DDRSP2	DDRSP1	DDRSP0
RESET:	0	0	0	0	0	0	0	0
Pin function:					SS	SCK	MOSI/ MOMI	MISO/ SISO

Figure 23-10: SPI Port Data Direction Register (SPIDDR)

Read: Anytime

Write: Anytime

RSVD[7:4] — Reserved

Writing to these read/write bits updates their values but has no effect on functionality.

DDRSP[3:0] — Data Direction Bits

The DDRSP[3:0] bits control the data direction of SPIPORT pins. Reset clears DDRSP[3:0].

1 = Corresponding pin configured as output

0 = Corresponding pin configured as input

In slave mode, DDRSP3 has no meaning or effect. In master mode, the DDRSP3 and the SSOE bits determine whether SPI port pin 3 is a mode-fault input, a general-purpose input, a general-

purpose output, or a slave-select output.

NOTE: *When the SPI is enabled (SPE = 1), the MISO, MOSI, and SCK pins:*

• Are inputs if their SPI functions are input functions regardless of the state of their DDRSP bits.

• Are outputs if their SPI functions are output functions only if their DDRSP bits are set

23.6.10. SPI Pullup and Reduced Drive Register

Offset Address: 0x0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	HS	PSW	RESERVED		MSPD1	MSPD0	PDPSP	PUPSP
RESET:	0	0	0	0	0	0	0	1

Figure 23-11: SPI Pullup and Reduced Drive Register (SPIPURD)

Read: Anytime;

Write: Anytime; writing to unimplemented bits has no effect

HS — High Speed Mode Enable Bit

Set HS to 1 in master mode makes MSPD[1:0] to take effect.

1 = High Speed Mode is enabled

0 = High Speed Mode is disabled

PSW — Pin Switch Bit

Switch MOSI to MISO and MISO to MOSI.

1 = Switch enabled

0 = Switch disabled

MSPD[1:0] — SPI Master Sample Point Delay

Specify the number of system clock cycles delayed for SCK sample edge

0x0=no delay;

0x1=1 cycle delay;

0x2=2 cycle delay;

0x3=2 cycle delay;

PDPSP — SPI Port Pulldown Enable Bit

1 = Pulldown devices enabled for SPIPORT bits [3:0]

0 = Pulldown devices disabled for SPIPORT bits [3:0]

PUPSP — SPI Port Pullup Enable Bit

1 = Pullup devices enabled for SPIPORT bits [3:0]

0 = Pullup devices disabled for SPIPORT bits [3:0]

23.6.11. SPI After SCK Delay Register

Offset Address: 0x000A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:					0			
Write:	ASCDE	PASCD6	PASCD5	PASCD4		ASCD2	ASCD1	ASCD0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 23-12: SPI After SCK Delay Register (SPIASCDR)

Read: Anytime

Write: Anytime

ASCDE — After SCK Delay Enable

1 = After SCK Delay is Enabled

0 = After SCK Delay is Disabled

PASCD[6:4] — SPI After SCK Delay Preselection Bits

The PASCD[6:4] and ASCD[2:0] bits select the SPI after SCK delay divisor.

ASCD[2:0] — SPI After SCK Delay Bits

$ASCD=(PASCD[6:4]+1)*2^{(ASCD[2:0]+1)}$; and $t=0.5*SCK+ASCD$

23.6.12. SPI Before SCK Delay Register

Offset Address: 0x000B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:					0			
Write:	BSCDE	PBSCD6	PBSCD5	PBSCD4		BSCD2	BSCD1	BSCD0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 23-13: SPI Before SCK Delay Register (SPIBSCDR)

Write: Anytime

BSCDE — Before SCK Delay Enable

1 = Before SCK Delay is Enabled

0 = Before SCK Delay is Disabled

PBSCD[6:4] — SPI Before SCK Delay Preselection Bits

The PBSCD[6:4] and BSCD[2:0] bits select the SPI before SCK delay divisor.

BSCD[2:0] — SPI Before SCK Delay Bits

$BSCD = (PBSCD[6:4]+1)*2^{(BSCD[2:0]+1)}$; and $t_L = 0.5*SCK+BSCD$

23.6.13. SPI Port Data Register

Offset Address: 0x000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RSVD7	RSVD6	RSVD5	RSVD4	PORTSP3	PORTSP2	PORTSP1	PORTSP0
RESET:	0	0	0	0	0	0	0	0
Pin function:					SS	SCK	MOSI/ MOMI	MISO/ SISO

Figure 23-14: SPI Port Data Register (SPIPORT)

Read: Anytime

Write: Anytime

RSVD[7:4] — Reserved

Writing to these read/write bits updates their values but has no effect on functionality.

PORTSP[3:0] — SPI Port Data Bits

Data written to SPIPORT drives pins only when they are configured as general-purpose outputs.

Reading an input (DDRSP bit clear) returns the pin level; reading an output (DDRSP bit set) returns the pin driver input level.

Writing to any of the PORTSP[3:0] pins does not change the pin state when the pin is configured for SPI output.

SPIPORT I/O function depends upon the state of the SPE bit in SPICR1 and the state the DDRSP bits in SPIDDR.

23.6.14. SPI Transmit Counter Register

Offset Address: 0x000D ~ 0x000F

	Bit 31	30	29	28	27	26	25	Bit 24
Read: Write:								
RESET:	0	0	0	0	0	0	0	0
	Bit 23	22	21	20	19	18	17	Bit 16
Read: Write:								
RESET:	0	0	0	0	0	0	0	0
	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 23-15: SPI Transmit Counter Register (SPITCNT)

Read: Anytime

Write: Anytime

SPITCNT[31:8] — SPI Transmit Counter Register

The counter is counted down by 1 when a frame is sent in master mode when CONT bit in SPIFR is asserted. Once the counter is counted down to 0 and another frame is sent, pin SS will set to high and the counter value will reload.

23.6.15. SPI Data Register

Offset Address: 0x0010

	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	15	14	13	12	11	10	9	8
RESET:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	7	6	5	4	3	2	1	0
RESET:	0	0	0	0	0	0	0	0

Figure 23-16: SPI Data Register (SPIDR)

Read: Anytime;

Write: Anytime;

SPIDR is both the input and output register for SPI data. Writing to SPIDR will fill the TX FIFO while reading from SPIDR will drain the RX FIFO.

23.6.16. SPI Status Register

Offset Address: 0x0014

	Bit 15	14	13	12	11	10	9	Bit 8
Read:	TXFTO	TXFOVF	TXFUDF	TXFSER	RXFTO	RXFOVF	RXFUDF	RXFSER
Write:	w1c	w1c	w1c		w1c	w1c	w1c	
RESET:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SPIF	FLOST	EOTF	MODF	TXFFULL	TXFEMP	RXFFULL	RXFEMP
Write:		w1c	w1c					
RESET:	0	0	0	0	0	1	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 23-17: SPI Status Register (SPISR)

Read: Anytime

Write: w1c means write 1 to the bit will clear the corresponding flag

SPIF — SPI Finish Flag

The SPIF flag is set after each single transfer. Clear SPIF by read SPIF and then access to SPIDR.

1 = a single transfer has finished

0 = a single transfer has not finished or no transfer

EOTF — End of Transmission Flag

The EOTF flag is set when all the data in TX FIFO is transmitted.

1 = End of Transmission

0 = Transmission not end or no transmission write 1 to the bit or write to SPIDR will clear this flag

FLOST — Frame Lost

When SPI is in slave mode and no valid data is in the TX FIFO, if the master start a transfer at this time, then SPI will return last received data to the master and FLOST is set. If the FLOSTIE bit is also set, FLOST generates an interrupt request.

1 = Frame lost has occurred

0 = No frame lost

write 1 to the bit will clear this flag

MODF — Mode Fault Flag

The MODF flag is set when the SS pin of a master SPI is driven low and the SS pin is configured as a mode-fault input. If the SPIE bit is also set, MODF generates an interrupt request. A mode fault clears the SPE, MSTR, and DDRSP[2:0] bits. Clear MODF by reading SPISR with MODF set and then writing to SPICR1. Reset clears MODF.

1 = Mode fault

0 = No mode fault

TXFFULL — TX FIFO Full Flag

1 = TX FIFO Full

0 = TX FIFO not Full

TXFEMP — TX FIFO empty Flag

1 = TX FIFO Empty

0 = TX FIFO not Empty

RXFFULL — RX FIFO Full Flag

1 = TX FIFO Full

0 = TX FIFO not Full

RXFEMP — RX FIFO empty Flag

1 = RX FIFO Empty

0 = RX FIFO not Empty

TXFTO — TX FIFO TimeOut

1 = TX FIFO TimeOut has occurred

0 = TX FIFO TimeOut has not occurred

write 1 to the bit or write to SPIDR will clear this flag

TXFOVF — TX FIFO Overflow Flag

1 = TX FIFO Overflow has occurred

0 = TX FIFO Overflow has not occurred

TXFUDF — TX FIFO Underflow Flag

1 = TX FIFO Underflow has occurred

0 = TX FIFO Underflow has not occurred

TXFSER — TX FIFO Service Flag

1 = TX FIFO data number below or equal TXFSTH

0 = TX FIFO data number above TXFSTH

RXFTO — RX FIFO TimeOut

1 = RX FIFO TimeOut has occurred

0 = RX FIFO TimeOut has not occurred

write 1 to the bit or read from SPIDR will clear this flag

RXFOVF — RX FIFO Overflow Flag

1 = RX FIFO Overflow has occurred

0 = RX FIFO Overflow has not occurred

RXFUDF — RX FIFO Underflow Flag

1 = RX FIFO Underflow has occurred

0 = RX FIFO Underflow has not occurred

RXFSER — RX FIFO Service Flag

1 = RX FIFO data number above RXFSTH

0 = RX FIFO data number below or equal RXFST

23.6.17. SPI RX FIFO Status Register

Offset Address: 0x0016

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	RXNXTP2	RXNXTP1	RXNXTP0	RXFCTR3	RXFCTR2	RXFCTR1	RXFCTR0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 23-18: SPI RX FIFO Status Register (SPIRXFSR)

Read: Anytime

Write: Has no meaning or effect

RXNXTP[2:0] — RX Next Pointer

Indicates the RX FIFO pointer to the data which is to be read out.

RXFCTR[3:0] — RX FIFO Counter

Indicates the RX FIFO data counter.

23.6.18. SPI TX FIFO Status Register

Offset Address: 0x0017

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	TXNXTP2	TXNXTP1	TXNXTP0	TXFCTR3	TXFCTR2	TXFCTR1	TXFCTR0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 23-19: SPI TX FIFO Status Register (SPITXFSR)

Read: Anytime

Write: Has no meaning or effect

TXNXTP[2:0] — TX Next Pointer
Indicates the TX FIFO pointer to the data which is to be transmitted.

TXFCTR[3:0] — TX FIFO Counter
Indicates the TX FIFO data counter.

23.6.19. SPI DMA Control Register

Offset Address: 0x0018

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	TXDMAE	RXDMAE
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 23-20: SPI DMA Control Register (SPIDMACR)

Read: Anytime;

Write: Anytime;

TXDMAE — TX FIFO DMA request Enable
1 = TX DMA request Enabled
0 = TX DMA request Disabled

RXDMAE — RX FIFO DMA request Enable
1 = RX DMA request Enabled
0 = RX DMA request Disabled

23.6.20. SPI DMA Threshold Register

Offset Address: 0x0019

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	TXDMATH	TXDMATH	TXDMATH	0	RXDMATH	RXDMATH	RXDMATH
Write:		2	1	0		2	1	0
RESET:	0	0	0	0	0	0	0	0

Figure 23-21: SPI DMA Threshold Register (SPIDMATHR)

Read: Anytime;

Write: Anytime;

TXDMATH[2:0] — TX DMA Threshold

Specify the data number threshold of TX FIFO. Once the data number in TX FIFO is less than the threshold and bit TXDMAE in SPIDMACR is set, SPI will send TX DMA request to the DMA.

data number = TXFDMATH[2:0]

RXDMATH[2:0] — RX DMA Threshold

Specify the data number threshold of RX FIFO. Once the data number in RX FIFO is more than the threshold and bit RXDMAE in SPIDMACR is set, SPI will send RX DMA request to the DMA.

data number = RXFDMATH[2:0]+1

23.6.21. SPI FIFO Debug Control Register

Offset Address: 0x001A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	TXFIDX2	TXFIDX1	TXFIDX0	0	RXFIDX2	RXFIDX1	RXFIDX0
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 23-22: SPI FIFO Debug Control Register (SPIFDCR)

Read: Anytime;

Write: Anytime;

TXFIDX[2:0] — TX FIFO Index

Specify the data index to be read from TX FIFO to SPITXFDBGR.

RXFIDX[2:0] — RX FIFO Index

Specify the data index to be read from RX FIFO to SPIRXFDBGR.

23.6.22. SPI Interrupt Control Register

Offset Address: 0x001B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	FLOSTIE	0	MODFIE	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 23-23: SPI Interrupt Control Register (SPIICR)

Read: Anytime;

Write: Anytime;

MODFIE — MODF Interrupt Enable.
1 = MODF Interrupt Enabled
0 = MODF Interrupt Disabled

FLOSTIE — FLOST Interrupt Enable.
1 = FLOST Interrupt Enabled
0 = FLOST Interrupt Disabled

23.6.23. SPI RX FIFO Debug Register

Offset Address: 0x001C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	7	6	5	4	3	2	1	0
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 23-24: SPI RX FIFO Debug Register (SPIRXFDBGR)

Read: Anytime;

Write: has no effect;

SPIRXFDBGR provides visibility into the RX FIFO for debugging purposes. The register is read-only and cannot be modified. Reading the SPITXRDBGR does not alter the state of the RX FIFO.

23.6.24. SPI TX FIFO Debug Register

Offset Address: 0x001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	7	6	5	4	3	2	1	0
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 23-25: SPI TX FIFO Debug Register (SPITXFDBGR)

Read: Anytime;
Write: has no effect;

SPITXFDBGR provides visibility into the TX FIFO for debugging purposes. The register is read-only and cannot be modified. Reading the SPITXFDBGR does not alter the state of the TX FIFO.

23.7. Functional Description.

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or SPI operation can be interrupt driven.

Setting the SPE bit in SPICR1 enables the SPI and dedicates four SPI port pins to SPI functions:

- Slave select (SS)
- Serial clock (SCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

When the SPE bit is clear, the SS, SCK, MOSI, and MISO pins are general-purpose I/O pins controlled by SPIDDR.

The shift register in a master SPI is linked by the MOSI and MISO pins to the shift register in the slave. The linked shift registers form a distributed register. In an SPI transmission, the SCK clock from the master shifts the data, and the master and slave exchange data. Data written to the master SPIDR register is the output data to the slave. After the exchange, data read from the master SPIDR is the input data from the slave.

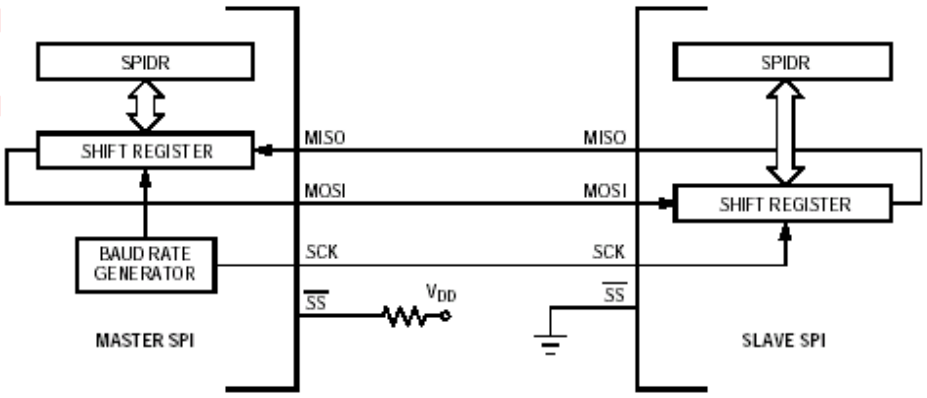


Figure 23-26: Full-Duplex Operation

23.7.1. Master Mode

Setting the MSTR bit in SPICR1 puts the SPI in master mode. Only a master SPI can initiate a transmission. Writing to the master SPIDR begins a transmission. If the shift register is empty, the byte transfers to the shift register and begins shifting out on the MOSI pin under the control of the master SCK clock. The SCK clock starts one-half SCK cycle after writing to SPIDR.

The SPR[2:0] and SPPR[6:4] bits in SPIBR control the baud rate generator and determine the speed of the shift register. The SCK pin is the SPI clock output. Through the SCK pin, the baud rate generator of the master controls the shift register of the slave.

The MSTR bit in SPICR1 and the SPC0 bit in SPICR2 control the function of the data pins, MOSI and MISO.

The SS pin is normally an input that remains in the inactive high state. Setting the DDRSP3 bit in SPIDDR configures SS as an output. The DDRSP3 bit and the SSOE bit in SPICR1 can configure SS for general-purpose I/O, mode fault detection, or slave selection. See **Table 23-3**

The SS output goes low during each transmission and is high when the SPI is in the idle state. Driving the master SS input low sets the MODF flag in SPISR, indicating a mode fault. More than one master may be trying to drive the MOSI and SCK lines simultaneously. A mode fault clears the data direction bits of the MISO, MOSI (or MOMI), and SCK pins to make them inputs. A mode fault also clears the SPE and MSTR bits in SPICR1. If the SPIE bit is also set, the MODF flag generates an interrupt request.

23.7.2. Slave Mode

Clearing the MSTR bit in SPICR1 puts the SPI in slave mode. The SCK pin is the SPI clock input from the master, and the SS pin is the slave-select input. For a transmission to occur, the SS pin must be driven low and remain low until the transmission is complete.

The MSTR bit and the SPC0 bit in SPICR2 control the function of the data pins, MOSI and MISO. The SS input also controls the MISO pin. If SS is low, the MSB in the shift register shifts out on the MISO pin. If SS is high, the MISO pin is in a high impedance state, and the slave ignores the SCK input.

NOTE: *When using peripherals with full-duplex capability, do not simultaneously enable two receivers that drive the same MISO output line.*

As long as only one slave drives the master input line, it is possible for several slaves to receive the same transmission simultaneously.

If the CPHA bit in SPICR1 is clear, odd-numbered edges on the SCK input latch the data on the MOSI pin. Even-numbered edges shift the data into the LSB position of the SPI shift register and shift the MSB out to the MISO pin.

If the CPHA bit is set, even-numbered edges on the SCK input latch the data on the MOSI pin. Odd-numbered edges shift the data into the LSB position of the SPI shift register and shift the MSB out to the MISO pin.

The transmission is complete after the eighth shift. The received data transfers to SPIDR, setting the SPIF flag in SPISR.

23.7.3. FIFO Operation

When a data size of less than 16 or 8bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 or 8bits is automatically right-justified in the receive buffer.

23.7.3.1. Transmit FIFO

The common transmit FIFO is an 8bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the SPI Data (SPIDR) register, and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the SPI Tx pin. In slave mode, the SPI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits received data in last transmit. Care should be taken to ensure that valid data is in the FIFO as needed. The SPI can be configured to generate an interrupt or a DMA request when the FIFO data number is less than the setting threshold.

23.7.3.2. Receive FIFO

The common receive FIFO is an 8bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the SPIDR register.

When configured as a master or slave, serial data received through the SPI Rx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

23.7.4. Transmission Formats

The CPHA and CPOL bits in SPICR1 select one of four combinations of serial clock phase and polarity. Clock phase and polarity must be identical for the master SPI device and the communicating slave device.

23.7.4.1. Transfer Format When CPHA = 1

Some peripherals require the first SCK edge to occur before the slave MSB becomes available at its MISO pin. When the CPHA bit is set, the master SPI waits for a synchronization delay of one-half SCK clock cycle. Then it issues the first SCK edge at the beginning of the transmission. The first edge causes the slave to transmit its MSB to the MISO pin of the master. The second edge and the following even-numbered edges latch the data. The third edge and the following odd-numbered edges shift the latched slave data into the master shift register and shift master data out on the master MOSI pin.

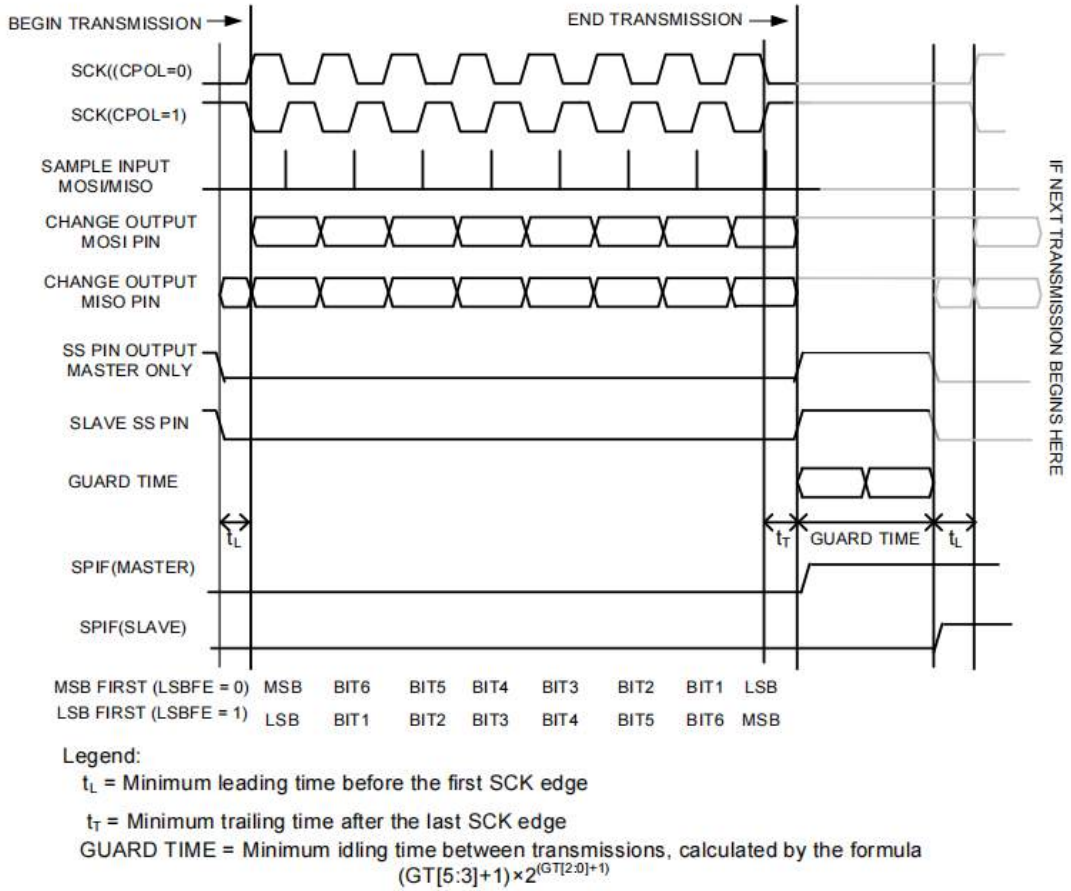
After the 16th and final SCK edge:

- Data that was in the master SPIDR register is in the slave SPIDR.
  Data that was in the slave SPIDR register is in the master SPIDR.
- The SCK clock stops and the SPIF flag in SPISR is set, indicating that the transmission is complete. If the SPIE bit in SPCR1 is set, SPIF generates an interrupt request.

Figure 23-27 shows the timing of a transmission with the CPHA bit set. The SS pin of the master must be either high or configured as a general-purpose output not affecting the SPI.

When CPHA = 1, the slave SS line can remain low between bytes. This format is good for systems with a single master and a single slave driving the MISO data line.

The SPIF interrupt request comes at the end of each single transfer.



Legend:

t_L = Minimum leading time before the first SCK edge

t_T = Minimum trailing time after the last SCK edge

GUARD TIME = Minimum idling time between transmissions, calculated by the formula

$$(GT[5:3]+1)\times 2^{(GT[2:0]+1)}$$

Figure 23-27: SPI Clock Format 1 (CPHA = 1)

23.7.4.2. Transfer Format When CPHA = 0

In some peripherals, the slave MSB is available at its MISO pin as soon as the slave is selected. When the CPHA bit is clear, the master SPI delays its first SCK edge for half a SCK cycle after the transmission starts. The first edge and all following odd-numbered edges latch the slave data. Even-numbered SCK edges shift slave data into the master shift register and shift master data out on the master MOSI pin.

After the 16th and final SCK edge:

- Data that was in the master SPIDR is in the slave SPIDR. Data that was in the slave SPIDR is in the master SPIDR.
- The SCK clock stops and the SPIF flag in SPISR is set, indicating that the transmission is complete. If the SPIE bit in SPCR1 is set, SPIF generates an interrupt request.

Figure 23-28 shows the timing of a transmission with the CPHA bit clear. The SS pin of the master must be either high or configured as a general-purpose output not affecting the SPI.

When CPHA = 0, the slave SS pin must be negated and reasserted between bytes.

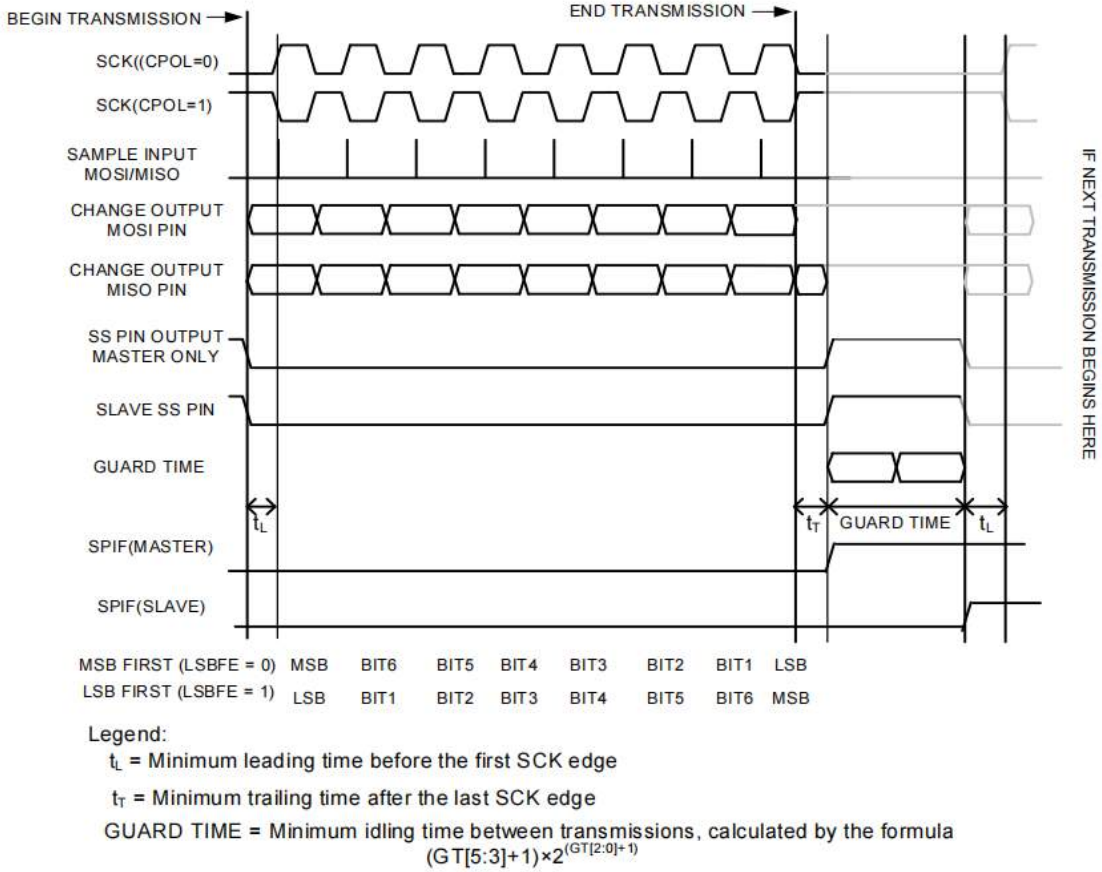


Figure 23-28: SPI Clock Format 0 (CPHA = 0)

NOTE:

Clock skew between the master and slave can cause data to be lost when:

- *CPHA = 0, and,*
- *The baud rate is the SPI clock divided by two, and*
- *The master SCK frequency is half the slave SPI clock frequency, and*
- *Software writes to the slave SPIDR just before the synchronized SS signal goes low.*

The synchronized SS signal is synchronized to the SPI clock. **Figure 23-29** shows an example with the synchronized SS signal almost a full SPI clock cycle late. While the synchronized SS of the slave is high, writing is allowed even though the SS pin is already low. The write can change the MISO pin while the master is sampling the MISO line. The first bit of the transfer may not be stable when the master samples it, so the byte sent to the master may be corrupted.

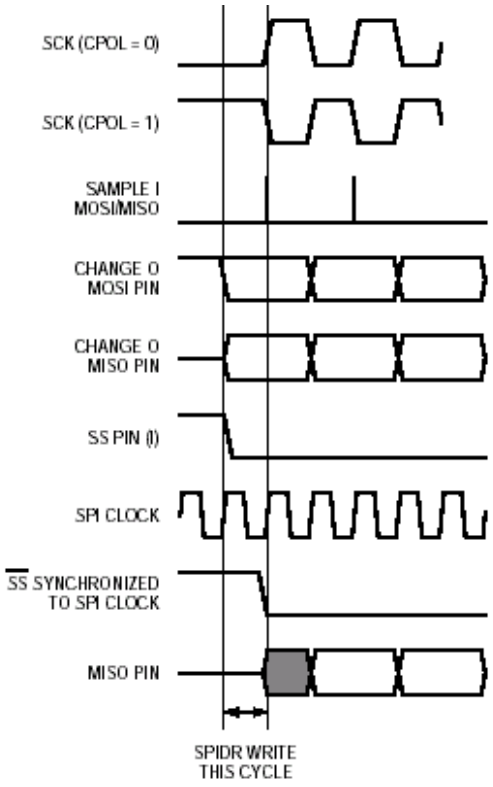


Figure 23-29: Transmission Error Due to Master/Slave Clock Skew

Also, if the slave generates a late write, its state machine may not have time to reset, causing it to incorrectly receive a byte from the master.

This error is most likely when the SCK frequency is half the slave SPI clock frequency. At other baud rates, the SCK skew is no more than one SPI clock, and there is more time between the synchronized SS signal and the first SCK edge. For example, with a SCK frequency one-fourth the slave SPI clock frequency, there are two SPI clocks between the fall of SS and the SCK edge.

As long as another late SPIDR write does not occur, the following bytes to and from the slave are correctly transmitted.

23.7.4.3. Texas Instruments Synchronous Serial Frame Format

In this mode, SCK and SS are forced Low, and the transmit data line SPI Tx is tristate whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SS is pulsed High for one SCK period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSIClk, the MSB of the 4 to 16bit data frame is shifted out on the SSITx pin. Likewise, the MSB of the received data is shifted onto the SSIRx pin by the off-chip serial slave device.

Both the SPI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of SCK. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SCK after the LSB has been latched.

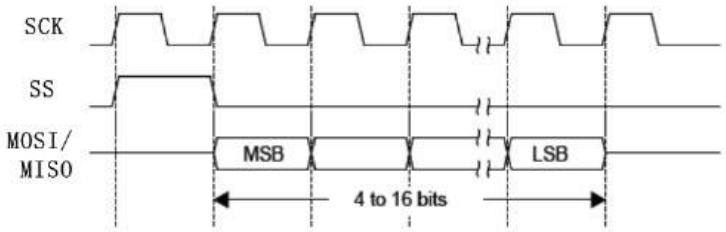


Figure 23-30: TI Single Transfer

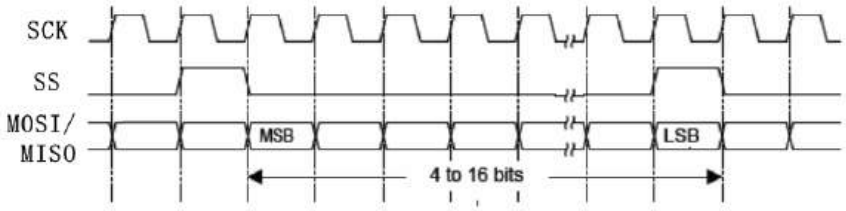


Figure 23-31: TI Continuous Transfer

23.7.5. SPI Baud Rate Generation

The baud rate generator divides the SPI clock to produce the SPI baud clock. The SPPR[6:4] and SPR[2:0] bits in SPIBR select the SPI clock divisor:

$$\text{SPI clock divisor} = (\text{SPPR} + 1) \times 2^{(\text{SPR} + 1)}$$

where:

SPPR = the value written to bits SPPR[6:4]

SPR = the value written to bits SPR[2:0]

The baud rate generator is active only when the SPI is in master mode and transmitting. Otherwise, the divider is inactive to reduce IDD current.

23.7.6. Slave-Select Output

The slave-select output feature automatically drives the SS pin low during transmission to select external devices and drives it high during idle to deselect external devices. When SS output is selected, the SS output pin is connected to the SS input pin of the external device.

In master mode only, setting the SSOE bit in SPICR1 and the DDRSP[3] bit in SPIDDR configures the SS pin as a slave-select output. Setting the SSOE bit disables the mode fault feature.

NOTE: *Be careful when using the slave-select output feature in a multimaster system. The mode fault feature is not available for detecting system errors between masters.*

23.7.7. Bidirectional Mode

Setting the SPC0 bit in SPICR1 selects bidirectional mode (see **Table 23-6**). The SPI uses only one data pin for the interface with external device(s). The MSTR bit determines which pin to use. In master mode, the MOSI pin is the master out/master in pin, MOMI. In slave mode, the MISO pin is the slave out/slave in pin, SISO. The MISO pin in master mode and MOSI pin in slave mode are general-purpose I/O pins.

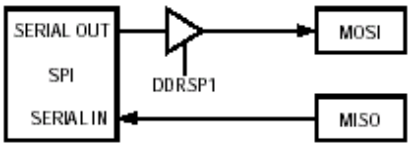
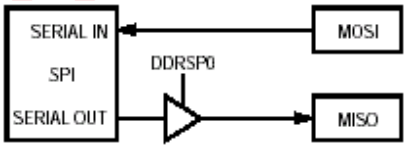
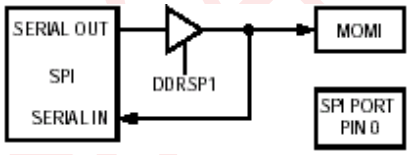
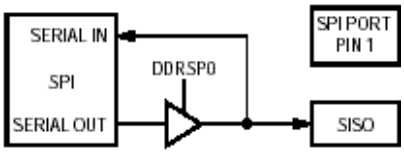
The direction of each data I/O pin depends on its data direction register bit. A pin configured as an output is the output from the shift register. A pin configured as an input is the input to the shift register, and data coming out of the shift register is discarded.

The SCK pin is an output in master mode and an input in slave mode.

The SS pin can be an input or an output in master mode, and it is always an input in slave mode.

In bidirectional mode, a mode fault does not clear DDRSP0, the data direction bit for the SISO pin.

Table 23-7: Normal Mode and Bidirectional Mode

SPE = 1	Master Mode, MSTR = 1	Slave Mode, MSTR = 0
Normal Mode SPC0 = 0	 SWOM enables open drain output.	 SWOM enables open drain output.
Bidirectional Mode SPC0 = 1	 SWOM enables open drain output. SPI port pin 0 is general-purpose I/O.	 SWOM enables open drain output. SPI port pin 1 is general-purpose I/O.

23.7.8. DMA Operation

The SPI peripheral provides an interface to the DMA controller with separate channels for transmit and receive. The DMA operation of the SPI is enabled through the SPI DMA Control (SPIDMACR)register. When DMA operation is enabled, the SPI asserts a DMA request on the receive or transmit channel when the associated FIFO meet transfer requirements. For the receive channel, a DMA request is asserted whenever the data number in the receive FIFO is above or equal to the threshold set by SPIDMATHR or RXF timeout counter set by SPIRXFTOCTR counts down to zero. For the transmit channel, a single transfer request is asserted whenever the data number in the transmit FIFO is below or equal to the threshold set by SPIDMATHR or TXF timeout counter set by SPITXFTOCTR counts down to zero. The requests are handled automatically by the DMA controller depending how the DMA channel is configured. To enable DMA operation for the receive channel, the RXDMAE bit of the DMA Control (SPIDMACR) register should be set. To enable DMA operation for the transmit channel, the TXDMAE bit of SPIDMACR should be set.

23.7.9. High Speed Mode

In high speed mode, both the master sample later and the slave shift data out earlier in the SCK period than in normal SPI mode to allow for delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

For the master, the high speed mode is enabled by HS bit in SPIURD register when configured as a master and the sample point delay is set by the MSPD[1:0] in SPIPURD register.

For the slave, the high speed mode is enabled by HS bit in SPIURD register when confided as a slave. If HS is set, SPI slave will shift out data at shift-in data edge while the shift-in data timing is the same as normal mode.

For the high speed mode to operate correctly, you must thoroughly analyze the SPI link timing budget.

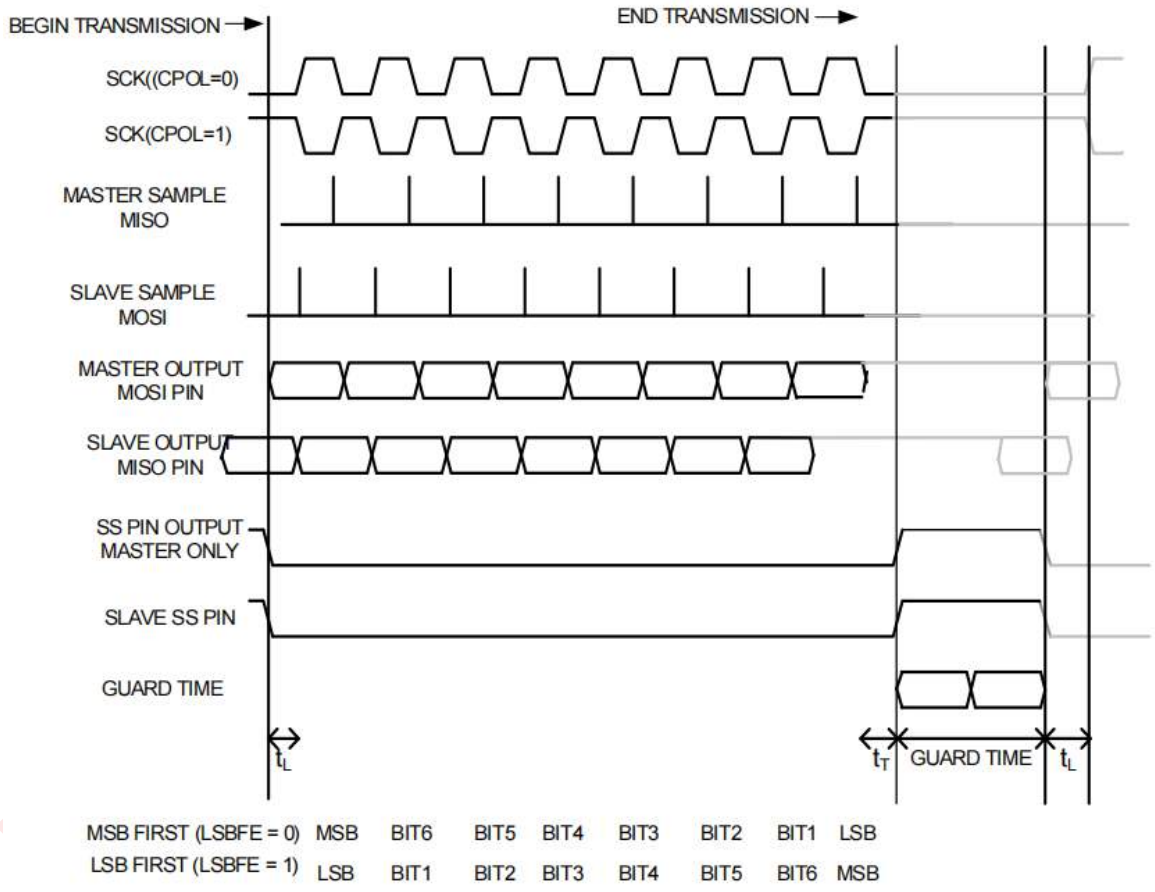


Figure 23-32: High Speed Mode (CPHA = 0)

23.7.10. Low-Power Mode Options

This subsection describes the low-power mode options.

23.7.10.1. Run Mode

Clearing the SPE bit in SPICR1 puts the SPI in a disabled, low-power state. SPI registers are accessible, but SPI clocks are disabled.

23.7.10.2. Doze Mode

SPI operation in doze mode depends on the state of the SPISDOZ bit in SPICR2.

- If SPISDOZ is clear, the SPI operates normally in doze mode.
- If SPISDOZ is set, the SPI clock stops, and the SPI enters a low-power state in doze mode.
  - Any master transmission in progress stops at doze mode entry and resumes at doze mode exit.
  - Any slave transmission in progress continues if a master continues to drive the slave SCK pin. The slave stays synchronized to the master SCK clock.

NOTE: *Although the slave shift register can receive MOSI data, it cannot transfer data to SPIDR or set the SPIF flag in doze or stop mode. If the slave enters doze mode in an idle state and exits doze mode in an idle state, SPIF remains clear and no transfer to SPIDR occurs.*

23.7.10.3. Stop Mode

SPI operation in stop mode is the same as in doze mode with the SPISDOZ bit set.

23.8. Reset

Reset initializes the SPI registers to a known startup state as described in **23.6 Memory Map and Registers**. A transmission from a slave after reset and before writing to the SPIDR register is either indeterminate or the byte last received from the master before the reset. Reading the SPIDR after reset returns 0s.

23.9. Interrupts

Table 23-8: SPI Interrupt Request Sources

Interrupt Request	Flag	Enable Bit
Mode fault	MODF	MODFIE
Transmission complete	EOTF	SPIE
Frame lost	FLOST	FLOSTIE
TXFIFO timeout	TXFTO	TXFTOIE
TXFIFO overflow	TXFOVF	TXFOVIE
TXFIFO underflow	TXFUDF	TXFUDIE
TXFIFO service	TXFSER	TXFSTHIE
RXFIFO timeout	RXFTO	RXFTOIE
RXFIFO overflow	RXFOVF	RXFOVIE
RXFIFO underflow	RXFUDF	RXFUDIE
RXFIFO service	RXFSER	RXFSTHIE

23.9.1. Mode Fault (MODF) Flag

MODF is set when the SS pin of a master SPI is driven low and the SS pin is configured as a mode-fault input. If the MODFIE bit is also set, MODF generates an interrupt request. A mode fault clears the SPE, MSTR, and DDRSP[2:0] bits. Clear MODF by reading SPISR with MODF set and then writing to SPICR1.

23.9.2. EOT Interrupt Flag (EOTF)

EOTF is set when all the data in TX FIFO is transmitted. If the SPIE bit is also set, EOTF generates an interrupt request. Clear EOTF by write 1 to this bit or fill TX FIFO by the CPU or DMA. Reset clears EOTF.

23.9.3. Frame Lost Interrupt Flag (FLOST)

When SPI is in slave mode and no valid data is in the TX FIFO, if the master start a transfer at this time, then SPI will return last received data to the master and FLOST is set. If the FLOSTIE bit is also set, FLOST generates an interrupt request. Clear FLOST by write 1 to this bit. Reset clears EOTF.

23.9.4. TXFIFO Timeout Interrupt Flag (TXFTO)

Once TX FIFO is not empty, the counter is on. If there is no operation to TX FIFO till the counter counts down to 0,the TXFTO is set. The counter is counted by SCK period. Clear TXFTO by write 1 to this bit.

23.9.5. TXFIFO Overflow Interrupt Flag (TXFOVF)

TXFOVF is set when data write operation is attempted to cause the data number of TX FIFO larger than 8.Clear TXFOVF by write 1 to this bit.

23.9.6. TXFIFO Underflow Interrupt Flag (TXFUDF)

TXFUDF is set when data write operation is attempted to cause the data number of TX FIFO smaller than 0.Clear TXFUDF by write 1 to this bit.

23.9.7. TXFIFO Service Interrupt Flag (TXFSER)

TXFSER is set when data number in TX FIFO is below or equal the threshold specified in SPITXFCR.

23.9.8. RXFIFO Timeout Interrupt Flag (RXFTO)

Once RX FIFO is not empty, the counter is on. If there is no operation to RX FIFO till the counter counts down to 0,the RXFTO is set. The counter is counted by SCK period. Clear RXFTO by write 1 to this bit.

23.9.9. RXFIFO Overflow Interrupt Flag (RXFOVF)

RXFOVF is set when data write operation is attempted to cause the data number of RX FIFO larger than 8.Clear RXFOVF by write 1 to this bit.

23.9.10. RXFIFO Underflow Interrupt Flag (RXFUDF)

RXFUDF is set when data write operation is attempted to cause the data number of RX FIFO smaller than 0.Clear RXFUDF by write 1 to this bit.

23.9.11. RXFIFO Service Interrupt Flag (RXFSER)

RXFSER is set when data number in RX FIFO is above or equal the threshold specified in SPIRXFCR.

24. Inter-Integrated Circuit (I2C)

24.1. Introduction

I2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I2C allows additional devices to be connected to the bus for expansion and system development.

The two bus lines provide data transfer rates up to 100Kbits/s IN Standard Mode, data rates up to 400Kbits/s in Fast Mode and data rates up to 3.4Mbits/s in High-Speed Mode.

The I2C system is a true multiple-master bus including arbitration and collision detection that prevents data corruption if multiple devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer.

24.2. Features

- Supports 7 bit addressing.
- Supports Standard Mode, Fast Mode and High-Speed Mode
- Software option to select between High-Speed mode and Standard/Fast mode
- Compatibility with standard and fast-mode of I2C bus version 2.1 standard.
- Multiple-master operation.
- Software-programmable for one of 64 different serial clock frequencies.
- Software-selectable acknowledge bit.
- Interrupt-driven, byte-by-byte data transfer.
- Arbitration-lost interrupt with automatic mode switching from master to slave.
- Transfer completion and read configure interrupt.
- Start and stop signal generation/detection.
- Repeated START signal generation.
- Acknowledge bit generation/detection.
- Bus-busy detection.
- Option slave address receiving enable when system clock stop mode
- SCL or SDA line GPIO function supported

24.3. System and Block Diagram

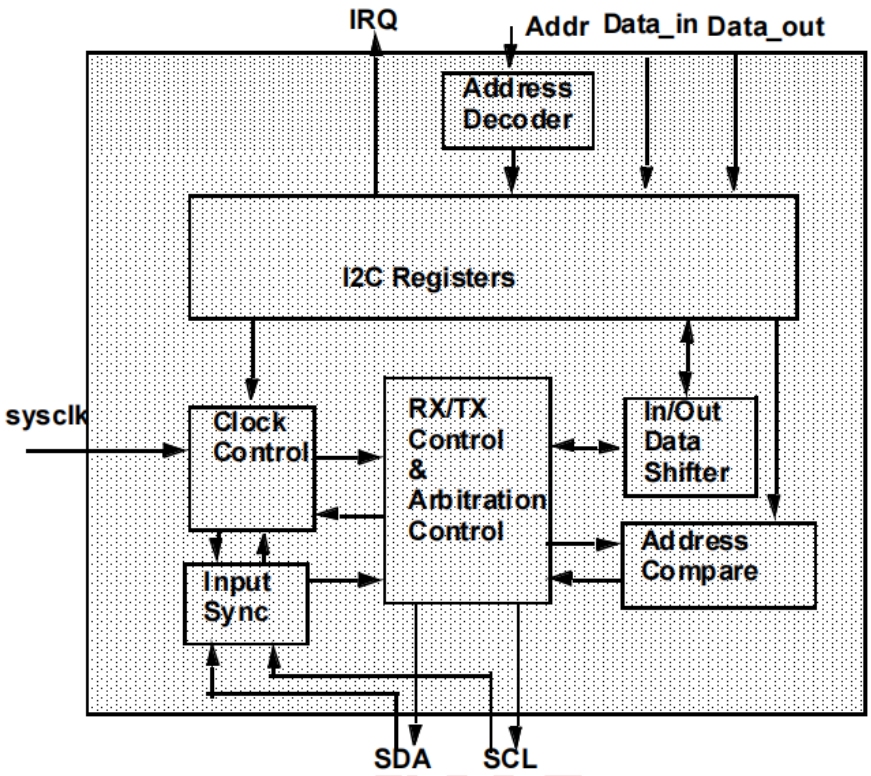


Figure 24-1: I2C Block Diagram

24.4. Memory Map and Registers

There are five registers in the I2C memory map, as shown in **Table 24-1**.

Table 24-1: I2C Memory Map

Offset Address	Bits 7- 0
0x0000	I2C Status Register (I2CS)
0x0001	I2C Clock Prescaler Register (I2CP)
0x0002	I2C Control Register (I2CC)
0x0003	I2C Slave Address Register (I2CSA)
0x0004	I2C Port Control Register (I2CPCR)
0x0005	I2C slave high-speed indicator register
0x0006	I2C slave SDA hold time Register
0x0007	I2C Data Register (I2CD)
0x0008	Reversed
0x0009	Reversed
0x000A	I2C Port Direction Register (I2CDDR)
0x000B	I2C Port Data Register (I2CPDR)

24.4.1. I2C Status Register(I2CS)

Offset Address: 0x0000

	Bit7	6	5	4	3	2	1	Bit0
Read:	AACK	DACK	RXTX	ARBL	BBUSY	AASLV	RC	TF
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 24-2: I2C Status Register(I2CS)

The I2CS register shows the status of I2C module.

TF - Transfer Complete Flag

Indicates there is data transmitted or received. For receiver, it is set by the falling edge of the ninth clock of a data(not address) byte received regardless of acknowledge bit detected or not. For master transmitter, it is set by the falling edge of the ninth clock of a data or address byte send regardless of acknowledge bit detected or not. For slave transmitter, it is set by the falling edge of the ninth clock of address or data byte transferred and acknowledge bit must be detected. If the IEN bit in I2CC is also set, an interrupt will be generated. Clear TF by reading I2CS with TF set and then accessing I2CD or master-transmit mode writing I2CC.

0 = meaningless,

1 = Data or address transfer complete.

RC - Receive Complete

Indicates it is time to configure the receiver. For master receiver, it is set by the falling edge of the ninth clock of data or address byte transferred regardless of acknowledge bit detected or not. For slave receiver, it is set by the falling edge of the ninth clock of address or data byte received and acknowledgement bit must be received. If the IEN bit in I2CC is also set, an interrupt will be generated. Clear RC by reading I2CS with RC set and then writing I2CC.

0 = meaningless,

1 = Indicates it is time to configure the receiver.

AASLV - Addressed as a slave

It is set by the falling edge of the eighth clock if I2C module is addressed as a slave and its own slave address matches the calling address received on SDL. It is clear by START or STOP bit detected

0 = Not as a slave.

1 = Addressed as a slave.

BBUSY - I2C bus busy

Shows the bus status.

0 = Bus is idle. It is cleared from STOP bit.

1 = Bus is busy. It is set from START bit.

ARBL - Arbitration lost

Shows the arbitration status of the bus. It will be set in the following cases during SCL high:

1. SDA is sampled low when the master drives high during START condition, address cycle, data-transmit cycle or STOP condition.
2. SDA is sampled low when the master drives high during the acknowledge bit of a data-

receive cycle.

3. A start cycle is attempted when the bus is busy.

ARBL must be cleared by software by reading the I2CS register.

0 = Arbitration not lost.

1 = Arbitration lost.

RXTX - Receive or transmit.

Indicates the I2C module function as receiver or transmitter. It is valid of falling edge of the eighth clock.

0 = Receive, receive data.

1 = Transmit, transmit data.

DACK - Data acknowledge received.

Indicates whether the acknowledge bit is detected during address or data transfer. It is valid of rising edge of the ninth clock.

0 = No acknowledge bit received.

1 = Acknowledge bit received.

AACK - Address acknowledge error.

Indicates whether the acknowledge bit was detected or not by master during address phase. It is set by the rising edge of ninth clock of the address phase and clear by changing MSMOD from 1 to 0 or repeat start condition.

0 = No address acknowledge error.

1 = Address acknowledge error.

24.4.2. I2C Clock Prescalar Register (I2CP)

Offset Address: 0x0001

	Bit7	6	5	4	3	2	1	Bit0
Read:	0	TEST	PRE5	PRE4	PRE3	PRE2	PRE1	PRE0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 24-3: I2C Clock Prescalar Register (I2CP)

I2CP is a prescaler to generate a bit-rate clock for the data transceiver. Due to potentially slow SCL and SDA rise and fall times, bus signals are sampled at the prescaler frequency. The serial bit clock frequency is equal to the OPB clock divided by the divider.

PRE[5:0] - Prescaler Divider Value

TEST - Clock Test Enable

It enables test mode for the I2C clock. In normal mode, its frequency is $f_{ipgclk}/(396x(PRE[5:0]+1)+1)$. In test mode, the frequency is $f_{ipgclk}/(8x(PRE[5:0]+1)+1)$.

1 = Clock Test Mode Enable

0 = Normal Mode

24.4.3. I2C Control Register (I2CC)

Offset Address: 0x0002

	Bit7	6	5	4	3	2	1	Bit0
Read: Write:	SLV_HSIE	HMS_EN	AMIE	REPSTA	ACKEN	MSMOD	IEN	EN
RESET:	0	0	0	0	1	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 24-4: I2C Control Register (I2CC)

I2CC is used to control the working of I2C module. EN - I2C

enable/disable control.

1 = module is enabled.

0 = module is disabled.

It enables/disables the module. Also controls the software reset of the entire I2C module. Setting the bit generates an internal reset to the module which gets asserted after 2 clocks of setting the bit and remain asserted for 3 clocks. Thus, reset gets negated after 5 clocks of setting EN bit.

If the module is enabled in the middle of a byte transfer, slave mode ignores the current I2C bus transfer and starts operating when the next start condition is detected. Master mode is not aware that the bus is busy; so, initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I2C module to lose arbitration, after which bus operation returns to normal.

IEN - I2C interrupt enable control.

It enables the I2C interrupt when it is set.

1 = I2C interrupt enabled.

0 = I2C interrupt disabled.

MSMOD - I2C master/slave mode selection control.

Changing MSMOD from 1 to 0 generates a STOP on the bus and selects slave mode. Changing MSMOD from 0 to 1 generates a START on the bus and selects master mode.

1 = Master mode.

0 = Slave mode.

ACKEN - Acknowledge enable control.

Specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. Note that ACKEN bit applies only when the I2C bus receives data byte. During receiving address, if the received address is matched with the slave address, the Acknowledge bit will be sent regardless of the ACKEN bit.

1 = An acknowledge signal is sent to the SDA at the ninth clock bit after receiving one byte of data.

0 = No acknowledge signal is sent to the SDA at the ninth clock bit after receiving one byte of data.

REPSTA - Repeat Start

In these case: 1) The master receiver wasn't acknowledged after sending slave address or data byte, 2) the master transmitter has sent an address or data byte regardless of

acknowledged or not, the master can repeat the START signal, followed by a new slave address. The repeat start bit will be send when configure slave address(by writing I2CD) after REPSTA bit set.

1 = Generate repeat start.

0 = No repeat start.

AMIE - Address Match Interrupt Enable

Selects whether Slave address match will generate interrupt request if I2C interrupt enable control is set to 1. AMIE should be set before entering stop mode to wakeup the system when slave address matched and must be clear when normal work mode.

1 = Enable address match interrupt request.

0 = Disable address match interrupt request. HSM_EN - High Speed Mode Enable

Selects the High Speed mode or Fast/Standard mode operation for the module in master mode. The HSM_EN bit should be set to select the High Speed mode operation. After the transmission of the master code in F/S mode and the HS_I2C does not loose the arbitration, HSM_EN should be set by software, and if you prepare to clear MSMOD to transfer STOP, HSM_EN should be clear by software after MSMOD.

0 = Fast/Standard mode operation (Default)

1 = High-Speed Mode operation

SLV_HSIE - Slave High-Speed Mode Interrupt Enable

Selects whether Slave High-Speed Mode status will generate interrupt request if 2C interrupt enable control is set to 1.

1 = Enable Slave High-Speed Mode status request.

0 = Disable Slave High-Speed Mode status request.

24.4.4. I2C Slave Address Register (I2CSA)

Offset Address: 0x0003

	Bit7	6	5	4	3	2	1	Bit0
Read:	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	0
Write:								
RESET:	0	1	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 24-5: I2C Slave Address Register (I2CSA)

The I2CSA stores the address when the I2C works as slave and responds to the address sent by a master.

ADDR[7:1] - Module Slave Address.

Slave address when the I2C module is in slave mode. (I2C is slave by default).

24.4.5. I2C Port Control Register (I2CPCR)

Offset Address: 0x0004

	Bit7	6	5	4	3	2	1	Bit0
Read: Write:	SDAPA	SCLPA	WOMI2C[1:0]		PDI2C[1:0]		PUI2C[1:0]	
RESET:	0	0	0	0	0	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 24-6: I2C Port Control Register (I2CPCR)

SDAPA - SDA Port Assignment Bit.

The read/write bit selects the SDA's function mode.

1 = Pin configured as GPIO.

0 = Pin configured as primary function.

SCLPA - SCL Port Assignment Bit.

The read/write bit selects the SCL's function mode.

1 = Pin configured as GPIO.

0 = Pin configured as primary function.

WOMI2C[1:0] — Wired-OR mode Bits

The read/write bits set the corresponding I2C pins to open-drain drive mode. WOMI2C[1] is for SDA pin and WOMI2C[0] is for SCL pin. These bits are available only in GPIO mode.

1 = Open-drain when output.

0 = CMOS drive when output.

PDI2C[1:0] — Pull-down Enable Bits

The read/write bits enable the pull-downs of corresponding I2C pins. PDI2C[1] is for SDA pin and PDI2C[0] is for SCL pin. These bits are available both in GPIO and main function mode.

1 = Pullup Enabled.

0 = Pullup Disabled.

PUI2C[1:0] — Pull-up Enable Bits

The read/write bits enable the pullups of corresponding I2C pins. PUI2C[1] is for SDA pin and PUI2C[0] is for SCL pin. These bits are available both in GPIO and main function mode.

1 = Pullup Enabled.

0 = Pullup Disabled.

24.4.6. I2C Slave High-Speed Mode Indicator Register(I2CSHIR)

Offset Address: 0x0005

	Bit7	6	5	4	3	2	1	Bit0
Read:	0	0	0	0	0	0	0	SLV_HS
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 24-7: I2C Slave High-Speed Mode Indicator Register(I2CSHIR)

SLV_HS -Slave high speed mode.

When I2C is selected as slave, this bit indicates whether the module is selected for High-Speed mode or Fast/Standard mode data transfer. This bit is set on the ninth SCL rising edge of Master code and Not Acknowledgement bit is received. This bit must be cleared by writing 1 to it otherwise the SCL line will be force to LOW state.

0 = I2C is selected for Fast/Standard data transfer (Default).

1 = I2C is selected for High-Speed mode data transfer.

24.4.7. I2C Slave SDA Hold Time Register(I2CSHT)

Offset Address: 0x0006

	Bit7	6	5	4	3	2	1	Bit0
Read: Write:	SCL_FILTER_EN	SDA_FILTER_EN	SLVHT					
RESET:	0	0	0	0	1	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

Figure 24-8: I2C Slave SDA Hold Time Register(I2CSHT)

SCL_FILTER_EN - SCL Filter enable.

If SCL filter is enabled, when the HS mode transfer is ongoing, the 10ns pulse occurred on SCL line will be filter out. If the F/S mode transfer is ongoing, the 50ns pulse occurred on SCL line will be filter on.

SDA_FILTER_EN - SDA Filter enable.

If SDA filter is enabled, when the HS mode transfer is ongoing, the 10ns pulse occurred on SDA line will be filter out. If the F/S mode transfer is ongoing, the 50ns pulse occurred on SDA line will be filter on.

SLVHT - slave SDA line hold time configuration.

When I2C work as slave output mode, the data will be changed after SCL falling edge and the value of internal SDA hold register equal to SLVHT. Writing value of 0 is not allowed.

24.4.8. I2C Data Register(I2CD)

Offset Address: 0x0007

	Bit7	6	5	4	3	2	1	Bit0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 24-9: I2C Data Register(I2CD)

The I2CD register holds the data to be transmitted (next byte) or data received. In master mode it also holds the slave address and transfer direction to be transmitted. Bits [7:1] form the slave address and bit [0] is the transfer direction (R/W). In master-receiver mode, reading I2CD allows a read to occur and initiates next byte data receiving. In master-transmit mode, writing I2CD will store the byte of the next transmit. In slave mode, the same function is available after it is addressed.

R[7:0] - Receive Bits [7:0]

T[7:0] - Transmit Bits [7:0]

24.4.9. I2C Port Direction Register (I2CDDR)

Offset Address: 0x000A

	Bit7	6	5	4	3	2	1	Bit0
Read:	0	0	0	0	0		DDRI2C[1:0]	
Write:								
RESET:	0	0	0	0	0		0	0

Figure 24-10: I2C Port Direction Register (I2CDDR)

Read: Anytime

Write: Anytime

DDRI2C[1:0] — I2C Data Direction Bits

The read/write bits control the data direction of the I2C pins. These bits are available only in GPIO mode

1 = Corresponding pin configured as output

0 = Corresponding pin configured as input

24.4.10. I2C Port Data Register (I2CPDR)

Offset Address: 0x000B

	Bit7	6	5	4	3	2	1	Bit0
Read:	0	0	0	0	0	0	PORTI2C[1:0]	
Write:								
RESET:	0	0	0	0	0	0	0	0

Figure 24-11: I2C Port Data Register (I2CPDR)

Read: Anytime

Write: Anytime

PORTI2C[1:0] — I2C Port Data Bits

Writes to these bits set the output data for the corresponding I2C pins configured as GPIO. Reading these bits return the I2C pins' level.

24.5. Functional Description

The I2C module is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. Software can poll the I2C status flags or I2C operation can be interrupt driven. When a byte is received or to be transmitted, the TF bit in I2CS will be set and if the IEN bit in I2CC is also set, an interrupt will be generated.

If arbitration is lost during its transfer, the ARBL will be set and if the IEN bit in I2CC is also set, an interrupt will be generated. An interrupt can also be generated if there is no address acknowledge from the slave (AACK set).

The module can clear ACKEN if it does not want to receive next byte.

24.5.1. Master Mode

The I2C module may initial a transfer if the bus is free (the BBUSY bit of I2CS is clear) when it works as a master. Changing the MSMOD bit from 0 to 1 generates a START on the bus and selects master mode. The master controls the direction of transfer, namely the R/W bit. The first byte of a transfer sent by the master is the slave address, the following byte is data. Change the MSMOD bit from 1 to 0 to generate a STOP on the bus if no acknowledge is received after each ninth cycle of clock. The PRE[5:0] bits in I2CP control the bit-rate clock of I2C-bus.

By configuring slave address after setting the REPSTA bit, the master can repeat the START signal instead of signaling a STOP.

24.5.2. Slave Mode

If the MSMOD bit is cleared the module is a slave and can be addressed by other masters. When a winning master is trying to address it, it will release the SDA line and switch over immediately to its slave mode.

NOTE: *the I2C can't work in slave mode and master mode at the same time.*

24.5.3. Protocol

The I2C communication protocol consists of six components: START, Data Source/Recipient, Data Direction, Slave Acknowledge, Data, Data Acknowledge and STOP. These are shown in **Figure 24-2** and described in the following text.

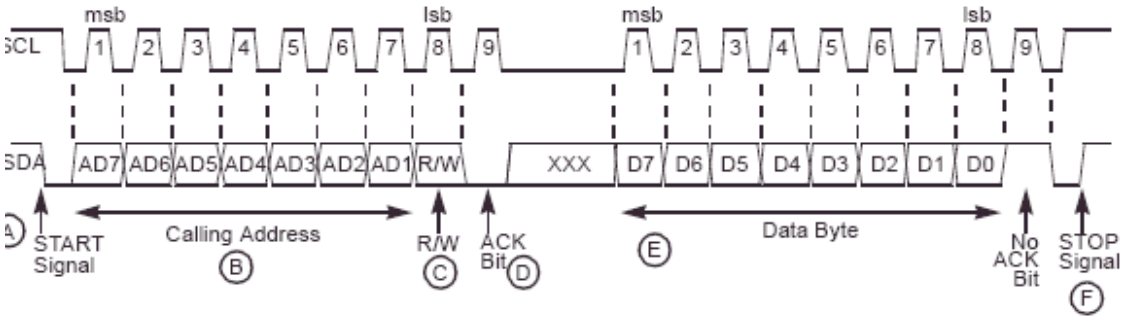


Figure 24-12: I2C Communication Protocol

1. START signal — When no other device is bus master (both SCL and SDA lines are at logic high), a device can initiate communication by sending a START signal (see A in **Figure 24-2**). A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a data transfer (each data transfer can be several bytes long) and awakens all slaves.

2. Slave address transmission — The master sends the slave address in the first byte after the START signal (B). After the seven-bit calling address, it sends the R/W bit (C), which tells the slave the data transfer direction.

Each slave must have a unique address. An I2C master must not transmit an address that is the same as its own slave address; it cannot be master and slave at the same time. The slave whose address matches that sent by the master pulls SDA low at the ninth clock (D) to return an acknowledge bit.

3. Data transfer — When successful slave addressing is achieved, the data transfer can proceed (E) on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master.

Data can be changed only while SCL is low and must be held stable while SCL is high, as **Figure 24-2** shows. SCL is pulsed once for each data bit, with the MSB being sent first. The receiving device must acknowledge each byte by pulling SDA low at the ninth clock; therefore, a data byte transfer takes nine clock pulses.

If it does not acknowledge the master, the slave receiver must leave SDA high. The master can then generate a STOP signal to abort the data transfer or generate a START signal (repeated start, shown in **Figure 24-3**) to start a new calling sequence.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means end-of-data to the slave. The slave releases SDA for the master to generate a STOP or START signal.

4. STOP signal — The master can terminate communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical high (F). Note that a master can generate a STOP even if the slave has made an acknowledgment, at which point the slave must release the bus.

5. Instead of signaling a STOP, the master can repeat the START signal, followed by a calling command, (A in **Figure 24-3**). A repeated START occurs when a START signal is generated without first generating a STOP signal to end the communication. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode), without releasing the bus.

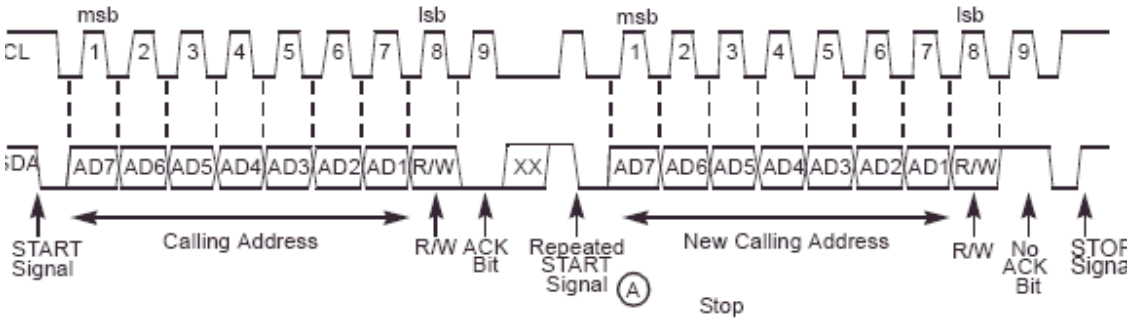


Figure 24-13: Repeat Start Of I2C Protocol

24.5.4. Arbitration Procedure

If multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the low period equals the longest clock-low period among the devices and the high period equals the shortest. A data arbitration procedure determines the relative priority of competing devices. A device loses arbitration if it sends logic high while another sends logic low; it immediately switches to slave-receive mode and stops driving SDA.

A master that loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration. It's possible that the winning master is trying to address it. The losing master must therefore switch over immediately to its slave mode.

In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets the ARBL bit of I2CSR to indicate loss of arbitration.

24.5.5. Clock Synchronization

Because wired-AND logic is used, a high-to-low transition on SCL affects all devices connected to the bus. Devices start counting their low period when the master drives SCL low. When a device clock goes low, it holds SCL low until the clock high state is reached. However, the low-to-high change in this device clock may not change the state of SCL if another device clock is still in its low period.

Therefore, the device with the longest low period holds the synchronized clock SCL low. Devices with shorter low periods enter a high wait state during this time (See **Figure 24-4**). When all devices involved have counted off their low period, the synchronized clock SCL is released and pulled high.

There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

24.5.6. Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold SCL low after completing one byte transfer (9 bits). In such a case, the clock mechanism halts the bus clock and forces the master clock into wait states until the slave releases SCL.

24.5.7. Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is longer than the master SCL low period, the resulting SCL bus signal low period is stretched.

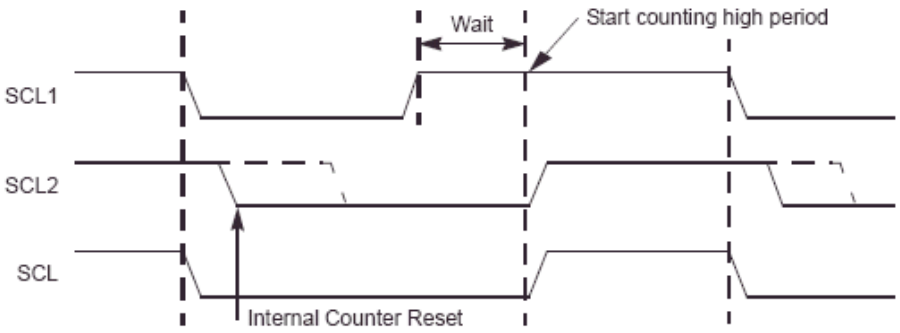


Figure 24-14: SCL Synchronization

24.5.8. High-Speed Mode operation

The I2C module can transfer information at bit rates of up to 3.4 Mbits/s in HS-mode, yet it remains fully downward compatible with Fast or Standard-mode (F/S-mode) devices for bi-directional communication in a mixed-speed bus system. With the exception that arbitration and clock synchronization is not performed during the HS-mode transfer, the same serial bus protocol and data format is maintained as with the F/S-mode system.

Serial data transfer format in HS-mode meets the Standard-mode I2C-bus specification. HS-mode can only commence after the following conditions are met (all of which occur while in F/S-mode):

1. START condition (S)
2. 8bit master code (00001XXX)
3. not-acknowledge bit (A)

(see **Figure 24-5 Data transfer format in HS mode**) and (see **Figure 24-6 A Complete HS mode transfer**) show this in more detail. The HS master code has two main functions:

1. It allows arbitration and synchronization between competing masters at F/S-mode speeds, resulting in one winning master.
2. It indicates the beginning of an HS-mode transfer.

HS-mode master codes are reserved 8bit codes, which are not used for slave addressing or other purposes. Furthermore, as each master has its own unique master code, up to eight HS-mode masters can be present on the one I2C-bus system (although master code 0000 1000 should be reserved for test and diagnostic purposes). The master code for an I2C module is software programmable. Arbitration and clock synchronization only take place during the transmission of the master code and not-acknowledge bit (A), after which one winning master remains active. The master code indicates to other devices that an HS-mode transfer is to begin and the connected devices must meet the HS-mode specification. As no device is allowed to acknowledge the master code, the master code is followed by a not-acknowledge (A). After the not-acknowledge bit (A), and the SCL line has been pulled-up to a HIGH level, the active master switches to HS-mode and enables (at time tH, refer to (see **Figure 24-6 A Complete HS mode transfer**)) the current-source pull-up circuit for the SCL signal. As other devices can delay the serial transfer before tH by stretching the LOW period of the SCL signal, the active master will enable its current-source pull-up circuit when all devices have released the SCL line and the SC signal has reached a HIGH level, thus speeding up the last part of the rise time of the SCL signal. The active master then sends a repeated START condition (Sr) followed by a 7bit slave address with a R/W bit address, and receives an acknowledge bit (A) from the selected slave.

After a repeated START condition and after each acknowledge bit (A) or not-acknowledge bit (A), the active master disables its current-source pull-up circuit. This enables other devices to delay the serial

transfer by stretching the LOW period of the SCL signal. The active master re-enables its current-source pull-up circuit again when all devices have released and the SCL signal reaches a HIGH level, and so speeds up the last part of the SCL signal s rise time.

Data transfer continues in Hs-mode after the next repeated START (Sr), and only switches back to F/S-mode after a STOP condition (P). To reduce the overhead of the master code, it's possible that a master links a number of HS-mode transfers, separated by repeated START conditions (Sr).

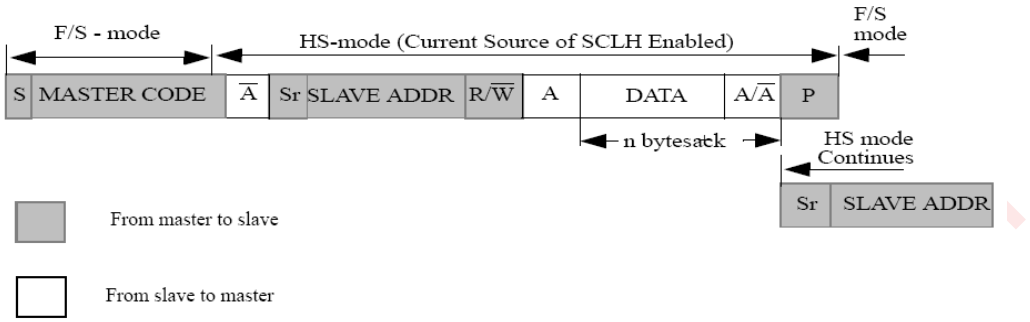


Figure 24-15: Data Transfer Format In HS Mode

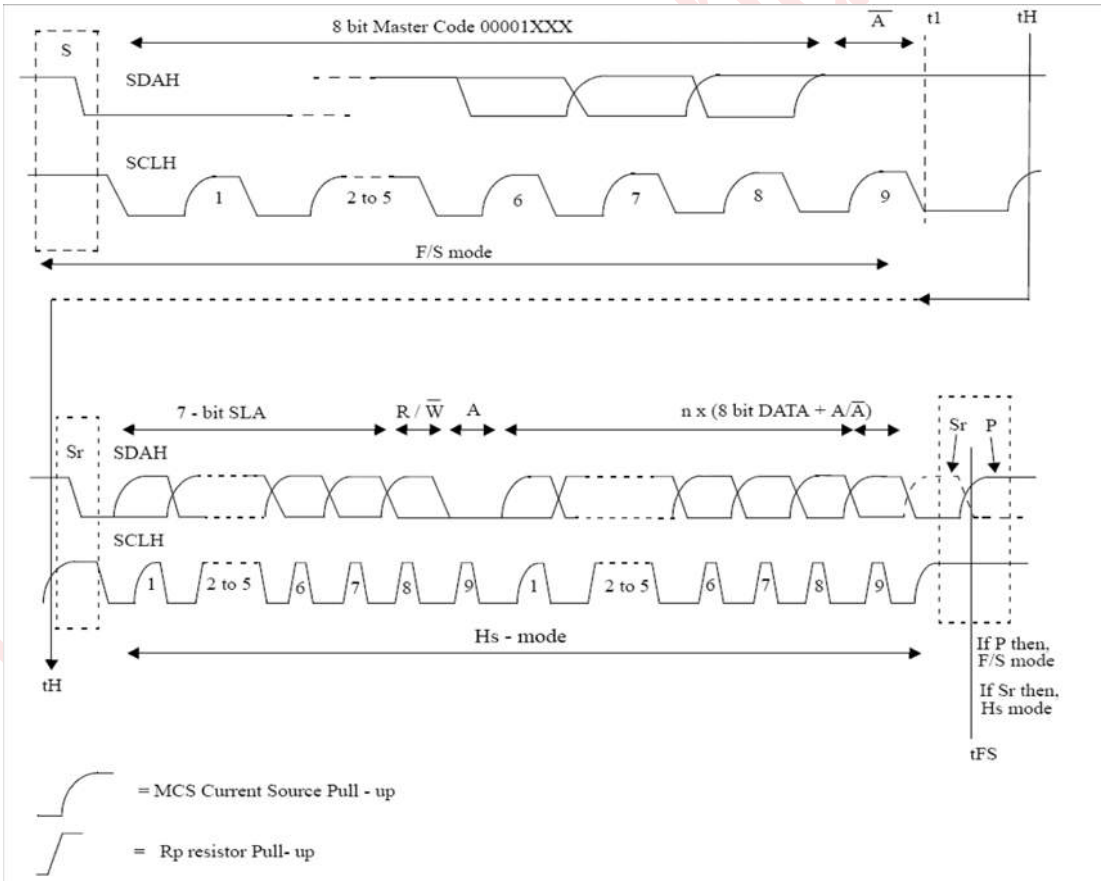


Figure 24-16: A Complete HS Mode Transfer

24.5.9. Software Transaction Flow Diagrams

1. Initialization.

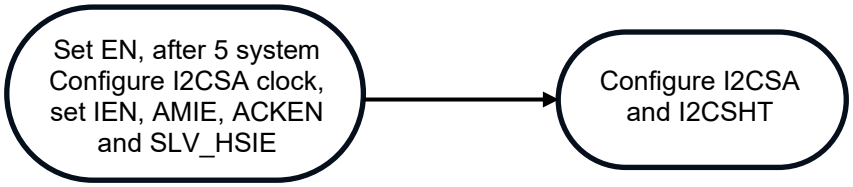


Figure 24-17: Slave Mode Initialization

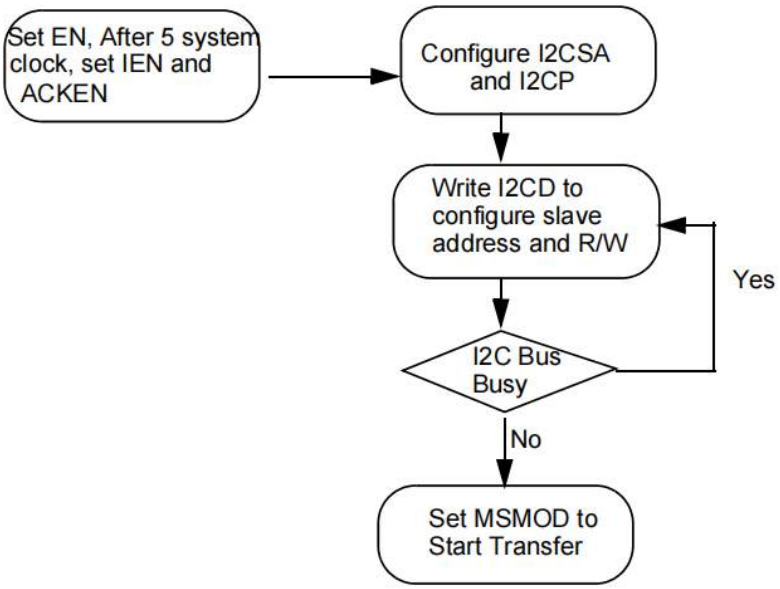


Figure 24-18: Master Mode Initialization

2. Interrupt transaction

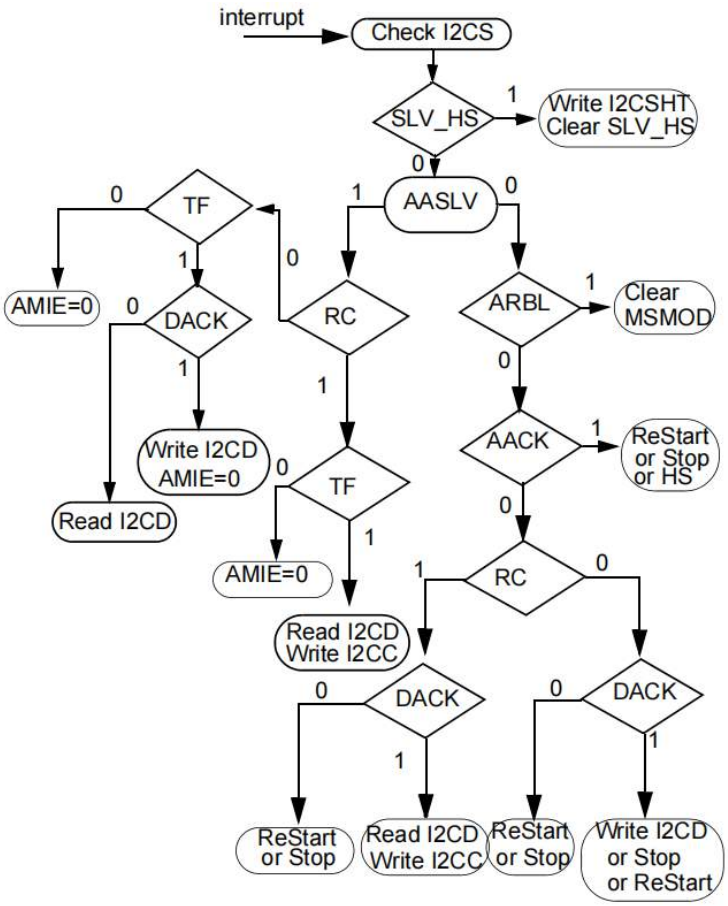


Figure 24-19: Interrupt Transaction

If there is an interruption comes from I2C,first check I2CS to confirm whether the I2C is in the master or slave mode.

First check SLV_HS status, if SLV_HS is set, then write I2CSHT to configure High-Speed Timing and write 1 to I2CSHIR to clear SLV_HS, otherwise:

In the slave mode, if RC has been set, means I2C is in the slave-receiver mode then check TF, if TF has also been set, means data(not address)has been received, then read I2CD and write I2CC to clear RC and TF to receive next byte data. If TF has not been set, means only slave address has been received, then write I2CC to clear RC and AMIE then receive next byte data.

In the slave mode, if RC has not been set, then check TF, if TF has been set, then check DACK, if DACK has also been set, means I2C is in the slave-transmit mode, then write I2CD to clear TF, write I2CC to clear AMIE and transmit next byte data. if DACK has not been set, then read I2CD for last received byte to clear TF.

In the slave mode, if both the TF and RC are not set, then write I2CC to clear AMIE.

In the master mode, if ARBL has been set, means arbitration lost has been occurred, then write I2CC to clear MSMOD. if ARBL has not been set then check AACK, if AACK has been set, means address acknowledge error, then write I2CC to repeat start or stop.

In the master mode, if ARBL and AACK has not been set, then check RC, if RC has been set, means I2C is in the mast-receiver mode, then check DACK, if DACK has also been set, then read I2CD to

clear RC and write I2CC to choice whether acknowledge the data. if DACK has not been set, write I2CC to repeat start or stop.

In the master mode, if ARBL and AACK has not been set, then check RC, if RC has not been set, means I2C is in the master-transmit mode, then check DACK, if DACK has been set, write I2CD to clear TF and transmit next byte data or write I2CC to repeat start or stop. if DACK has not been set, then write I2CC to repeat start or stop.

25. Pulse Width Modulator (PWM)

25.1. Introduction

There are 4 PWM-Timers enclosed. The 4 PWM-Timers has 2 Pre-scale, 2 clock divider, 4 clock selectors, 4 16bit counters, 4 16bit comparators, 2 Dead-Zone generators. Each can be used as a timer and issues interrupt independently.

Each two PWM-Timers share the same pre-scale. Clock divider provides each timer with 5 clock sources (1, 1/2, 1/4, 1/8, 1/16). The 16bit counter in each timer receive clock signal from clock selector and can be used to handle one PWM period. The 16bit comparator compares number in counter with threshold number in register loaded previously to generate PWM duty cycle. The clock signal from clock divider is called PWM clock. Dead-Zone generator utilize PWM clock as clock source. Once Dead-Zone generator is enabled, output of two PWM-Timers are blocked. Two output pins are all used as Dead-Zone generator output signal to control off-chip power device. The value of comparator is used for pulse width modulation. The counter control logic changes the output level when down-counter value matches the value of compare register.

Each PWM-Timer includes a capture channel. The Capture 0 and PWM 0 share a timer that included in PWM 0; and the Capture 1 and PWM 1 share another timer, and etc. Therefore, user must setup the PWM-Timer before turn on Capture feature. After enabling capture feature, the capture always latched PWM-counter to CRLR when input channel has a rising transition and latched PWM-counter to CFLR when input channel has a falling transition. Capture channel 0 interrupt is programmable by setting CCR0[1] (Rising latch Interrupt enable) and CCR0[2] (Falling latch Interrupt enable) to decide the condition of interrupt occur. Capture channel1&2&3 has the same feature. Whenever Capture issues Interrupt 0/1/2/3, the PWM counter 0/1/2/3 will be reload at this moment. The maximal capture frequency should be decided by interrupt process time. If interrupt process time is T0,then the capture channel input signal should not change in T0,the maximal capture frequency is 1/T0.

There are only four interrupts from PWM to interrupt controller (INTC). PWM 0 and Capture 0 share the same interrupt; PWM1 and Capture 1 share the same interrupt and so on. Therefore, PWM function and Capture function in the same channel cannot be used at the same time.

25.2. Features

The Pulse Width Modulator includes these distinctive features:

- Programmable period
- Programmable duty cycle
- Two Dead-Zone generator
- Capture function
- Pins can be configured as general-purpose I/O

25.3. Block Diagram

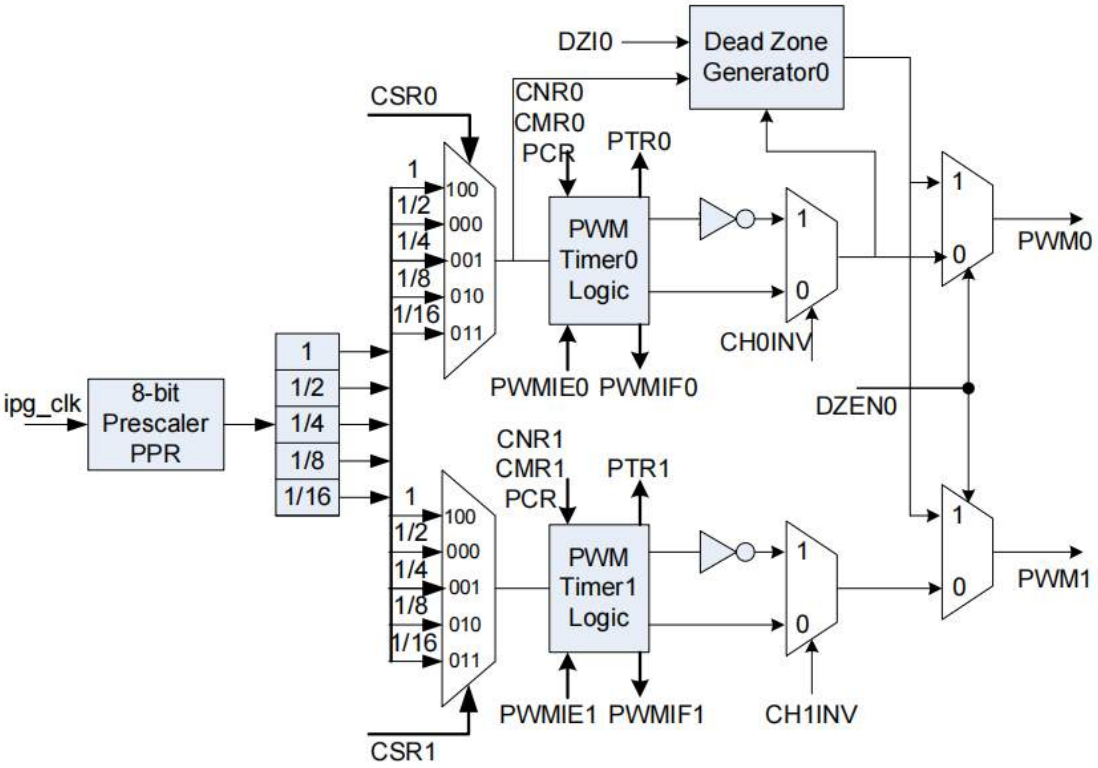


Figure 25-1: PWM Block Diagram

25.4. Signal Description

Table 25-1: PWM Signal Description

Name	I/O	Width	Reset State	Description
PWM0	I/O	1	0	PWM0 pin
PWM1	I/O	1	0	PWM1 pin
PWM2	I/O	1	0	PWM2 pin
PWM3	I/O	1	0	PWM3 pin

PWMx are used as a general-purpose input / output, and also used as the PWM send output or capture input

In default state, it is used as general-purpose input port.

25.5. Memory Map and Registers

This subsection describes the memory map and register structure.

25.5.1. Memory Map

Table 25-2: Module Memory Map

Address	Bit[15:8]	Bit[7:0]	Access(1)
0x0000	PWM Pre-scale Register (PPR)		S/U
0x0004	PWM Clock Select Register (PCSR)		S/U
0x0008	PWM Control Register (PCR)		S/U
0x000C	PWM Counter Register0 (PCNR0)		S/U
0x0010	PWM Comparator Register0 (PCMR0)		S/U
0x0014	PWM Timer Register0 (PTR0)		S/U
0x0018	PWM Counter Register1 (PCNR1)		S/U
0x001C	PWM Comparator Register1 (PCMR1)		S/U
0x0020	PWM Timer Register1 (PTR1)		S/U
0x0024	PWM Counter Register2 (PCNR2)		S/U
0x0028	PWM Comparator Register2 (PCMR2)		S/U
0x002C	PWM Timer Register2 (PTR2)		S/U
0x0030	PWM Counter Register3 (PCNR3)		S/U
0x0034	PWM Comparator Register3 (PCMR3)		S/U
0x0038	PWM Timer Register3 (PTR3)		S/U
0x003C	PWM Interrupt Enable Register (PIER)		S/U
0x0040	PWM Interrupt Flag Register (PIFR)		S/U
0x0044	PWM Capture Control Register0 (PCCR0)		S/U
0x0048	PWM Capture Control Register1 (PCCR1)		S/U
0x004C	PWM Capture Rising Latch Register0 (PCRLR0)		S/U
0x0050	PWM Capture Falling Latch Register0 (PCFLR0)		S/U
0x0054	PWM Capture Rising Latch Register1 (PCRLR1)		S/U
0x0058	PWM Capture Falling Latch Register1 (PCFLR1)		S/U
0x005C	PWM Capture Rising Latch Register2 (PCRLR2)		S/U
0x0060	PWM Capture Falling Latch Register2 (PCFLR2)		S/U
0x0064	PWM Capture Rising Latch Register3 (PCRLR3)		S/U
0x0068	PWM Capture Falling Latch Register3 (PCFLR3)		S/U
0x006C	PWM Port Control Register (PPCR)		S/U

NOTE: *S/U = CPU supervisor or user mode access.*

25.5.2. Registers

25.5.2.1. PWM Pre-scale Register (PPR)

The register*(PPR)* is used to set prescaler and set dead zone length.

Offset Address: 0x0000 and 0x0003

	31	30	29	28	27	26	25	24
R/W	DZI1							
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R/W	DZI0							
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R/W	CP1							
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R/W	CP0							
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 25-2: PWM Pre-scale Register (PPR)

DZI1[7:0] — Dead zone interval register 1 for PWM2 and PWM3

These 8bit determine dead zone length. The 1 unit time of dead zone length is received from clock selector 1.

DZI0[7:0] — Dead zone interval register 0 for PWM0 and PWM1

These 8bit determine dead zone length. The 1 unit time of dead zone length is received from clock selector 0.

CP1[7:0] — Clock pre-scale 1 for PWM Timer 2 & 3

Clock input is divided by (CP1 + 1) before it is fed to the counter of PWM Timer 2 & 3.

CP0[7:0] — Clock pre-scale 0 for PWM Timer 0 & 1

Clock input is divided by (CP0 + 1) before it is fed to the counter of PWM Timer 0 & 1.

25.5.2.2. PWM Clock Select Register (PCSR)

Clock divider provides each timer with 5 clock sources (1, 1/2, 1/4, 1/8, 1/16). Each timer receives its own clock signal from clock divider which receives clock from 8bit pre-scale.

Offset Address: 0x0004 and 0x0007

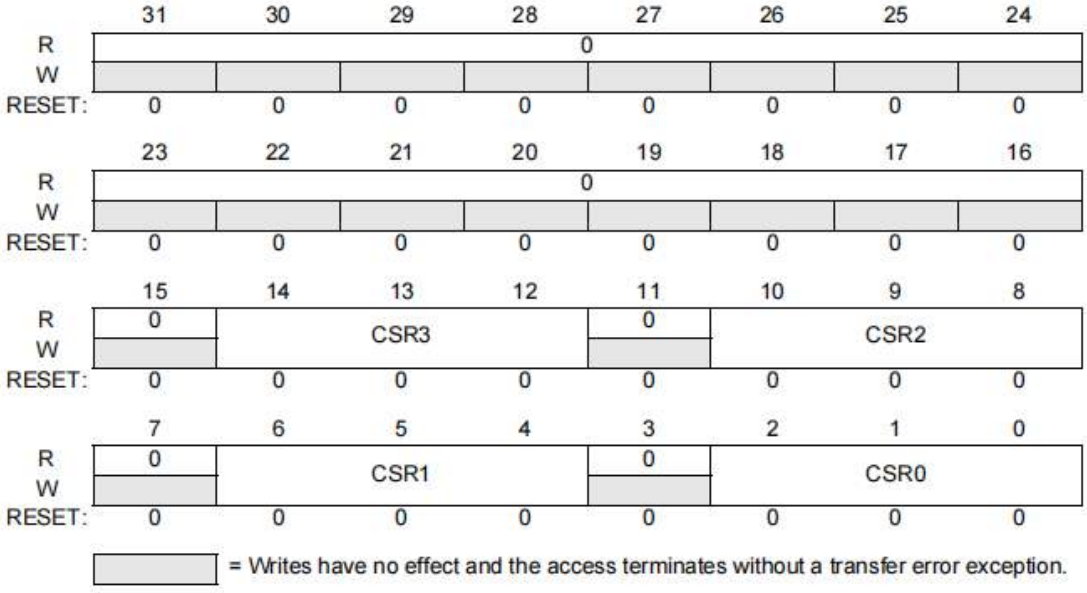


Figure 25-3: PWM Clock Select Register(PCSR)

CSR3[2:0] — Timer 3 Clock Source Selection

Select clock input for timer 3.

Table 25-3: PWM Clock Divider Setting

CSR3[2:0]	Input Clock Divided by
100~111	1
011	16
010	8
001	4
000	2

CSR2[2:0] — Timer 2 Clock Source Selection

Select clock input for timer 2.Same as CSR3.

CSR1[2:0] — Timer 1 Clock Source Selection

Select clock input for timer 1.Same as CSR3

CSR0[2:0] — Timer 0 Clock Source Selection

Select clock input for timer 0. Same as CSR3

25.5.2.3. PWM Control Register (PCR)

This register is the PWM control register.

Offset Address: 0x0008 and 0x000B

	31	30	29	28	27	26	25	24
R	0				CH3MOD	CH3INV	0	CH3EN
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0				CH2MOD	CH2INV	0	CH2EN
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0				CH1MOD	CH1INV	0	CH1EN
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0		DZEN1	DZEN0	CH0MOD	CH0INV	0	CH0EN
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 25-4: PWM Control Register(PCR)

CH3MOD — Timer 3 Auto-load/One-Shot Mode

1 = Auto-load Mode

0 = One-Shot Mode

NOTE: *If there is a rising or falling transition at this bit, it will cause CNR3 and CMR3 be clear.*

CH3INV — Timer 3 Inverter ON/OFF

1 = Inverter ON

0 = Inverter OFF

CH3EN — Timer 3 Enable/Disable

1 = Enable

0 = Disable

CH2MOD — Timer 2 Auto-load/One-Shot Mode

1 = Auto-load Mode

0 = One-Shot Mode

NOTE: *If there is a rising or falling transition at this bit, it will cause CNR2 and CMR2 be clear.*

CH2INV — Timer 2 Inverter ON/OFF

1 = Inverter ON

0 = Inverter OFF

CH2EN — Timer 2 Enable/Disable

1 = Enable

0 = Disable

CH1MOD — Timer 1 Auto-load/One-Shot Mode

1 = Auto-load Mode

0 = One-Shot Mode

NOTE: *If there is a rising or falling transition at this bit, it will cause CNR1 and CMR1 be clear.*

CH1INV — Timer 1 Inverter ON/OFF

1 = Inverter ON

0 = Inverter OFF

CH1EN — Timer 1 Enable/Disable

1 = Enable

0 = Disable

CH0MOD — Timer 0 Auto-load/One-Shot Mode

1 = Auto-load Mode

0 = One-Shot Mode

NOTE: *If there is a rising or falling transition at this bit, it will cause CNR0 and CMR0 be clear.*

CH0INV — Timer 0 Inverter ON/OFF

1 = Inverter ON

0 = Inverter OFF

CH0EN — Timer 0 Enable/Disable

1 = Enable

0 = Disable

DZEN1 — Dead-Zone 1 Generator Enable/Disable

1 = Enable

0 = Disable

NOTE: *When DZEN1 is enable,CH3EN should be set disable. Because channel3 and channel2 outputs both decided by channel2.*

DZEN0 — Dead-Zone 0 Generator Enable/Disable

1 = Enable

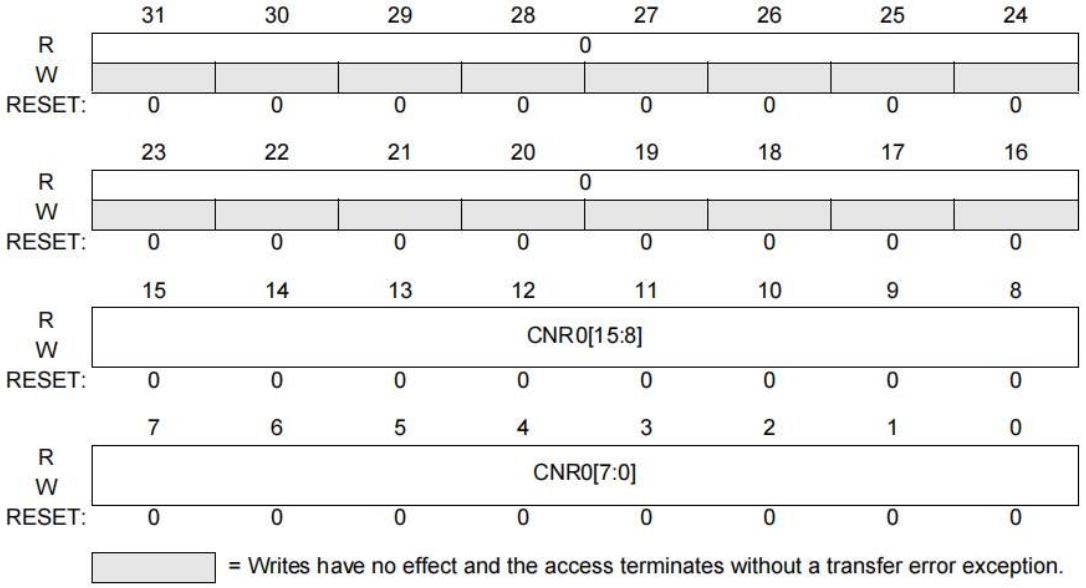
0 = Disable

NOTE: *When DZENO is enableCH1 EN should be set disable. Because channel1 and channel 0 outputs both decided by channel 0.*

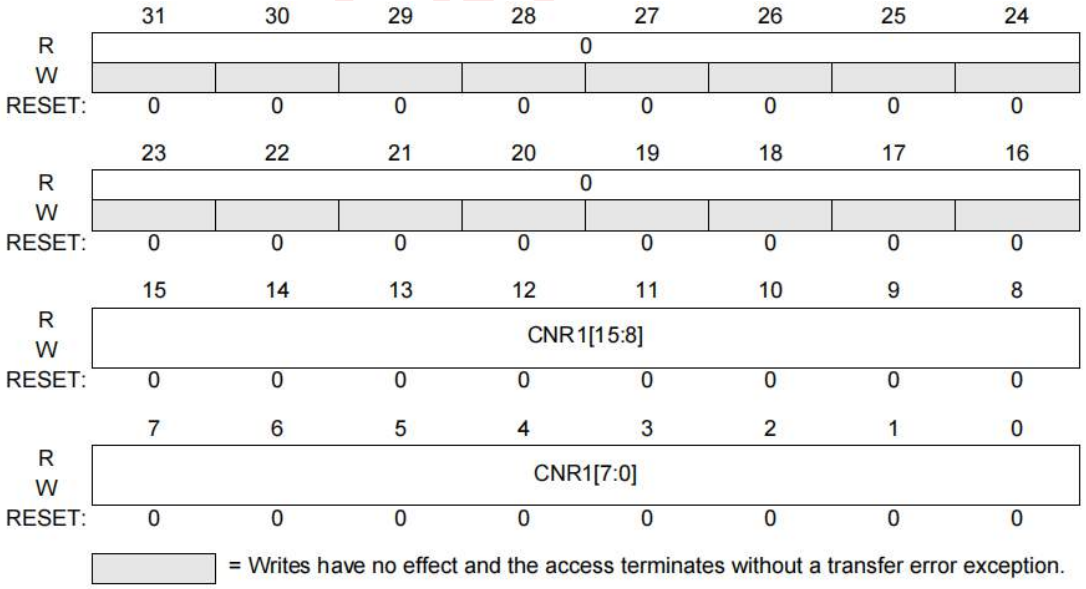
25.5.2.4. PWM Counter Register (PCNR0/1/2/3)

These registers control the period of the PWM by defining the number of the count_pulse in the period.

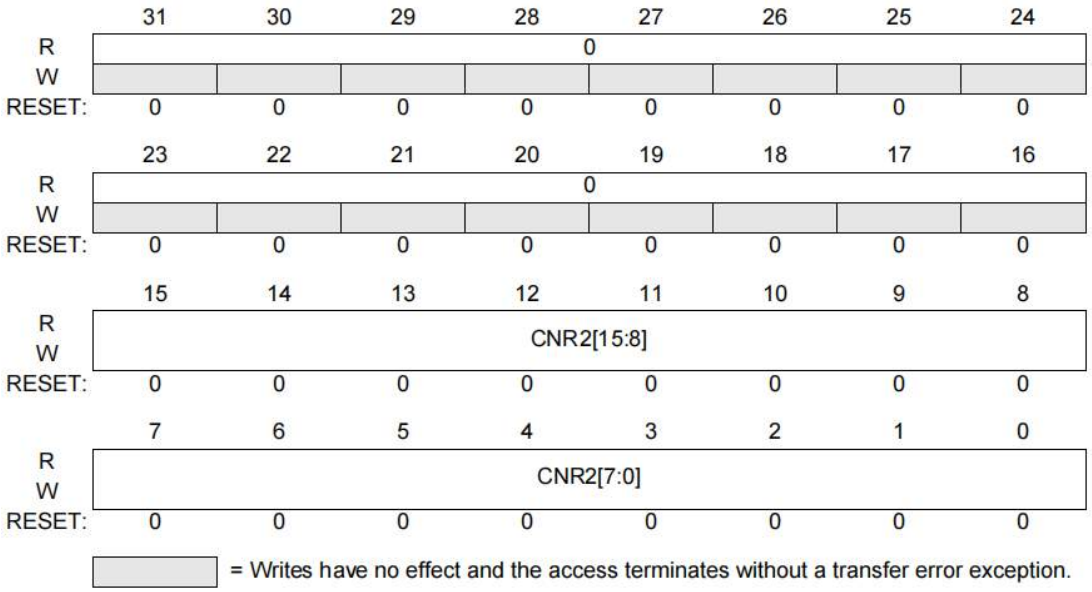
Offset Address: 0x000C and 0x000F



Offset Address: 0x0018 and 0x001B



Offset Address: 0x0024 and 0x0027



Offset Address: 0x0030 and 0x0033

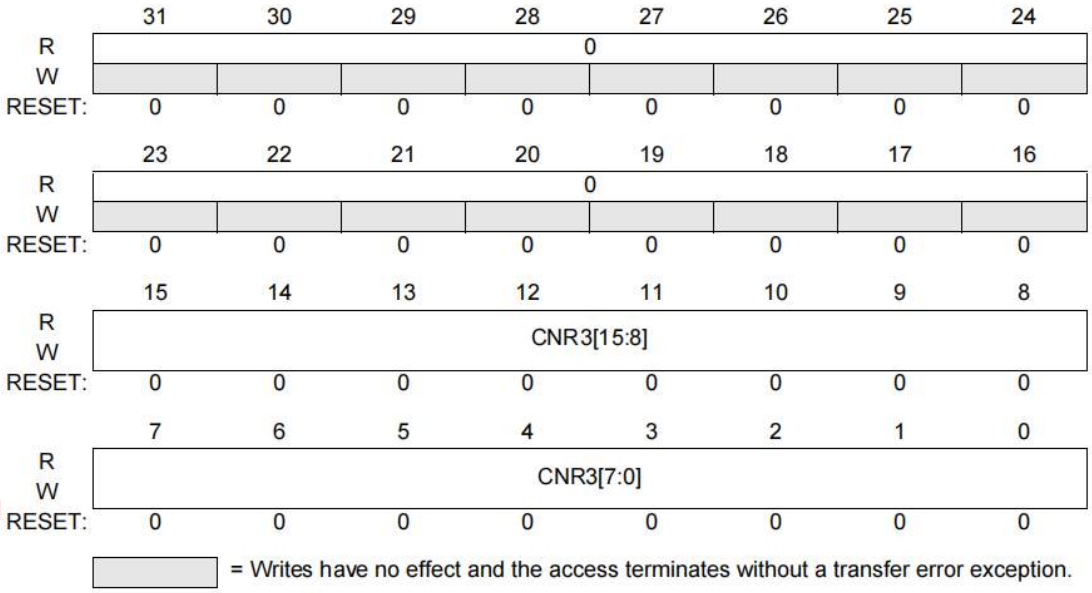


Figure 25-5: PWM Counter Register (PCNR)

CNR[15:0] — PWM Counter/Timer Loaded Value

Inserted data range : 65535~0 (Unit : 1 PWM clock cycle)

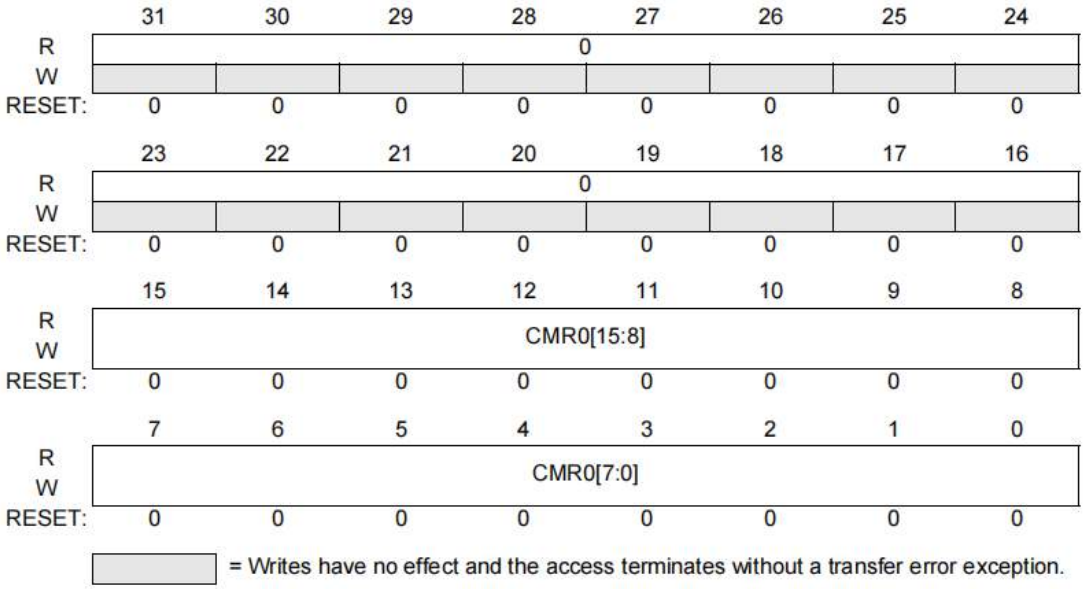
NOTE:

1. *One PWM cycle width = CNR + 1. If CNR equal zero, PWM counter/timer will be stopped.*
2. *:Programmer can feel free to write a data to CNR at any time, and it will take effect in next PWM Counter cycle.*

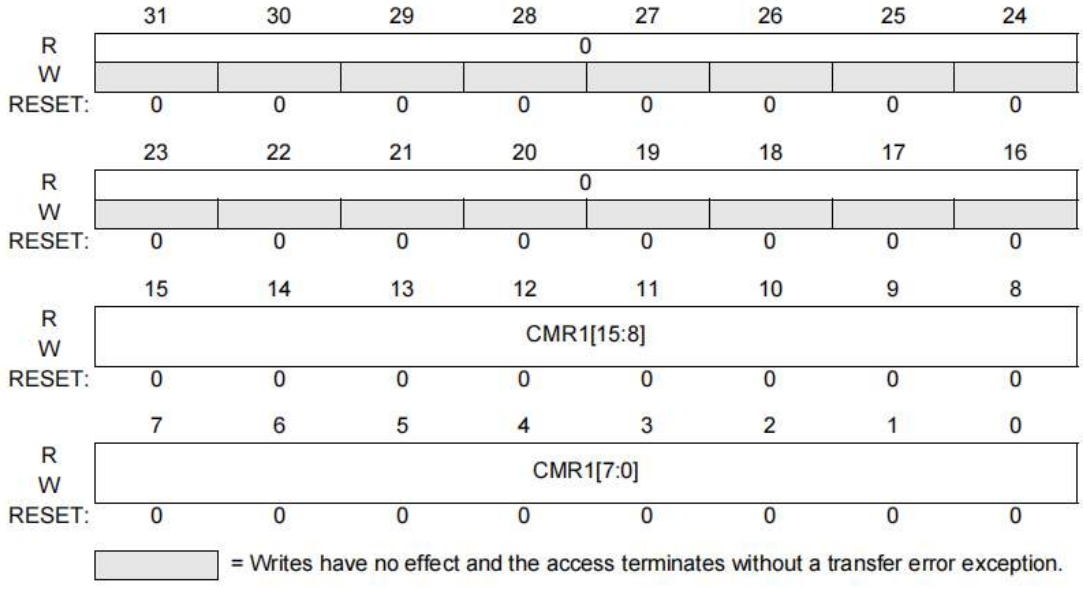
25.5.2.5. PWM Comparator Register (PCMR0/1/2/3)

These registers define the width of the pulse. When the counter matches the value in this register, the output is reset for the duration of the period.

Offset Address: 0x0010 and 0x0013



Offset Address: 0x001C and 0x001F



Offset Address: 0x0028 and 0x002B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CMR2[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CMR2[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x0034 and 0x0037

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CMR3[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CMR3[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 25-6: PWM Comparator Register (PCMR)

CMR[15:0] — PWM Comparator Register

Inserted data range : 65535~0 (Unit : 1 PWM clock cycle)

CMR are used to determine PWM output duty ratio.

Assumption : PWM output initial : high

CMR >= CNR : PWM output is always high

CMR < CNR : PWM output high = (CMR + 1) unit

CMR = 0 : PWM output high = 1 unit

NOTE:

1. *PWM duty = CMR + 1. If CMR equal zero, PWM duty = 1*
2. *Programmer can feel free to write a data to CMR at any time, and it will take effect in next PWM Counter cycle.*

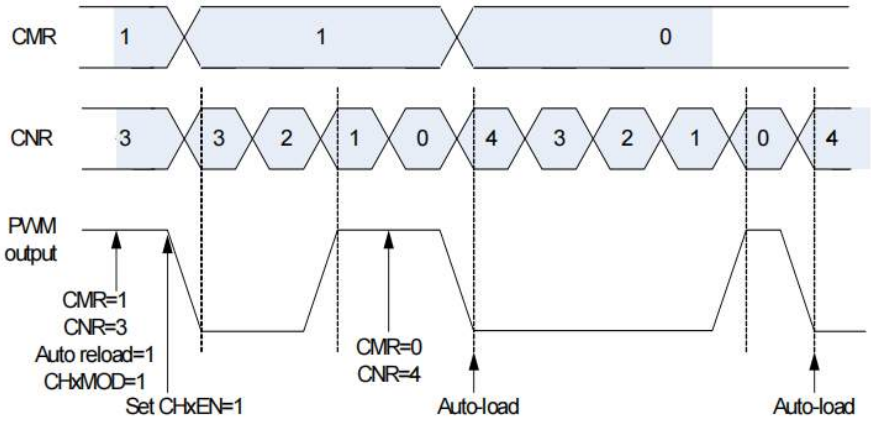


Figure 25-7: PWM Comparator Register Setting Timing

Modulate PWM Controller output duty ratio (CNR=150)

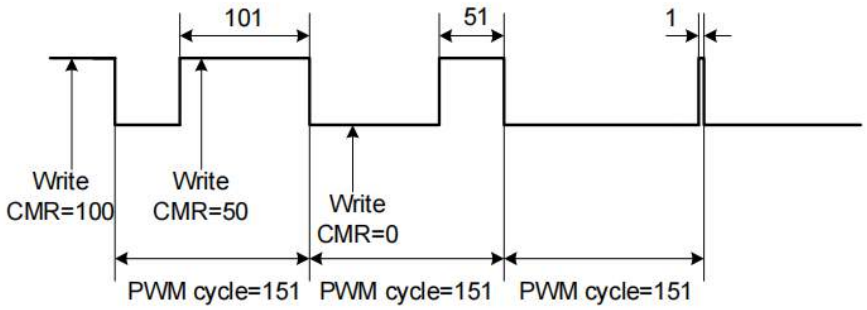


Figure 25-8: PWM Duty

25.5.2.6. PWM Timer Register (PTR0/1/2/3)

The read-only PWM timer registers hold the current count value. It can be read at any time without disturbing the counter.

Offset Address: 0x0014 and 0x0017

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	PTR0[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PTR0[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x0020 and 0x0023

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	PTR1[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PTR1[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x002C and 0x002F

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	PTR2[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PTR2[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x0038 and 0x003B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	PTR3[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PTR3[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 25-9: PWM Timer Register (PTR)

PTR[15:0] — PWM Timer value

The read-only PTR bits holds the current count value. User can monitor PTR to know current value in 16bit down counter.

25.5.2.7. PWM Interrupt Enable Register (PIER)

This register is used to enable PWM timer interrupt.

Offset Address: 0x003C and 0x003F

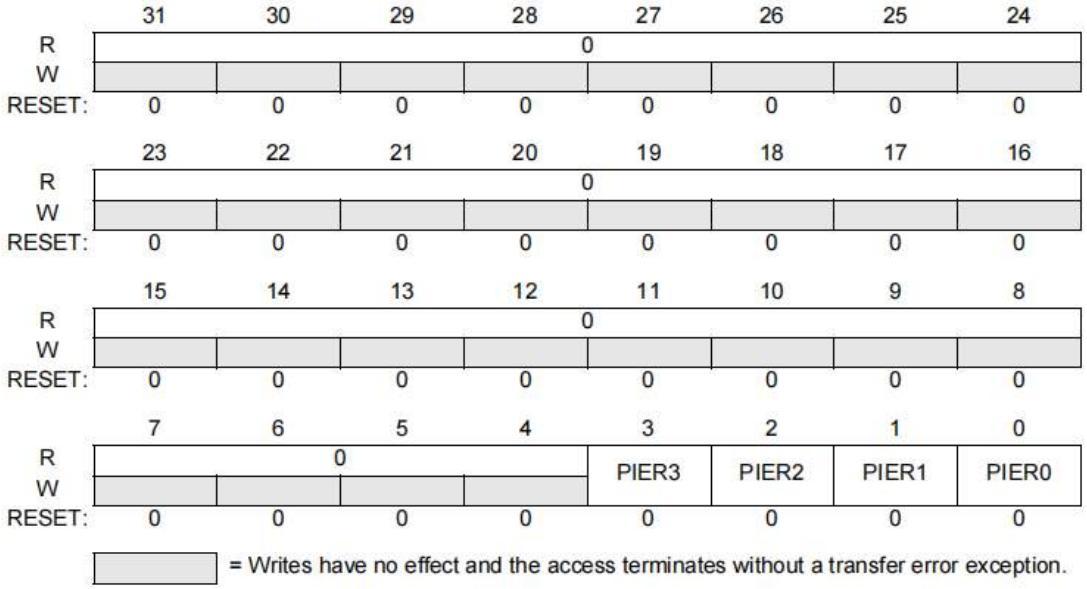


Figure 25-10: PWM Interrupt Enable Register (PIER)

PIER3 — PWM Timer 3 Interrupt Enable
1 = Enable
0 = Disable

PIER2 — PWM Timer 2 Interrupt Enable
1 = Enable
0 = Disable

PIER1 — PWM Timer 1 Interrupt Enable
1 = Enable
0 = Disable

PIER0 — PWM Timer 0 Interrupt Enable
1 = Enable
0 = Disable

25.5.2.8. PWM Interrupt Flag Register (PIFR)

This register is used to indicate PWM timer interrupt flag.

Offset Address: 0x0040 and 0x0043

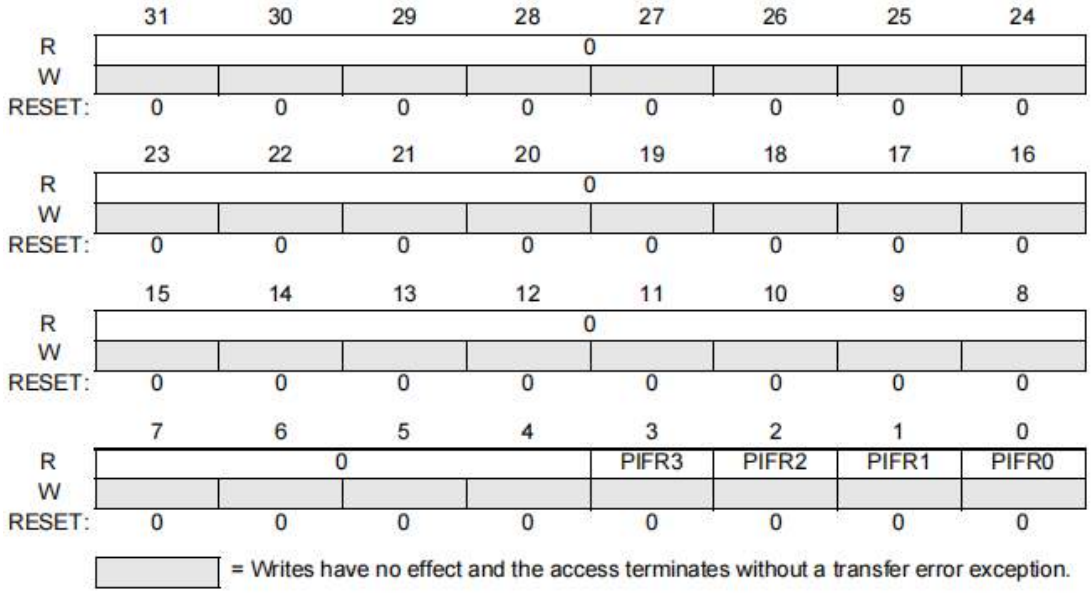


Figure 25-11: PWM Interrupt Flag Register (PIFR)

PIFR3 — PWM Timer 3 Interrupt Flag.

When PWM timer3 count to 0,and PIER3 =1,PIFR3 will be set to 1.Write 1 to this bit will clear PIFR3.

1 = Interrupt Flag on

0 = Interrupt Flag off

PIFR2 — PWM Timer 2 Interrupt Flag.

When PWM timer2 count to 0,and PIER2 =1,PIFR2 will be set to 1.Write 1 to this bit will clear PIFR2.

1 = Interrupt Flag on

0 = Interrupt Flag off

PIFR1 — PWM Timer 1 Interrupt Flag.

When PWM timer1 count to 0,and PIER1 =1,PIFR1 will be set to 1.Write 1 to this bit will clear PIFR1.

1 = Interrupt Flag on

0 = Interrupt Flag off

PIFR0 — PWM Timer 0 Interrupt Flag.

When PWM timer0 count to 0,and PIER0 =1,PIFR0 will be set to 1.Write 1 to this bit will clear PIFR0.

1 = Interrupt Flag on

0 = Interrupt Flag off

25.5.2.9. PWM Capture Control Register (PCCR0/1)

These registers are used to control capture function.

Offset Address: 0x0044 and 0x0047

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	CFLRD1	CRLRD1	0	CAPIF1	CAPCH1EN	FL_IE1	RL_IE1	INV1
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLRD0	CRLRD0	0	CAPIF0	CAPCH0EN	FL_IE0	RL_IE0	INV0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x0048 and 0x004B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	CFLRD3	CRLRD3	0	CAPIF3	CAPCH3EN	FL_IE3	RL_IE3	INV3
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLRD2	CRLRD2	0	CAPIF2	CAPCH2EN	FL_IE2	RL_IE2	INV2
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 25-12: PWM Capture Control Register (PCCR0/1)

CFLRDx — Capture Falling Latch Register load flag

1 = When input channel x has a falling transition, CFLRx was updated and this bit was "1".

0 = When input channel x doesn't have a falling transition. write 1 clear this bit.

CRLRDx — Capture Rising Latch Register load flag

1 = When input channel x has a rising transition, CRLRx was updated and this bit was "1".

0 = When input channel x doesn't have a rising transition. write 1 clear this bit.

CAPIFx — Capture Channel x interrupt flag

1 = When input channel x has a falling transition, and FL&IEx bit enable, this interrupt flag set. When input channel x has a rising transition, and RL&IEx bit enable, this interrupt flag also set.

0 = Capture channel x interrupt flag not set. write 1 clear this bit.

CAPCHxEN — Capture Channel x Enable/Disable

1 = Enable

0 = Disable

When Enable, Capture latched the PMW-counter and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disable, Capture does not update CRLR and CFLR, and disable Channel x Interrupt.

FL_IEx — Channel x Falling Interrupt Enable ON/OFF

1 = Enable

0 = Disable

When Enable, if Capture detects Channel x has falling transition, Capture issues an Interrupt.

RL_IEx — Channel x Rising Interrupt Enable ON/OFF

1 = Enable

0 = Disable

When Enable, if Capture detects Channel x has rising transition, Capture issues an Interrupt.

INVx — Channel x Inverter ON/OFF

1 = Inverter ON

0= Inverter OFF

25.5.2.10. PWM Capture Rising Latch Register (PCRLR0/1/2/3)

These registers are used to latch the PWM counter when capture rising transition.

Offset Address: 0x004C and 0x004F

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CRLR0[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CRLR0[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x0054 and 0x0057

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

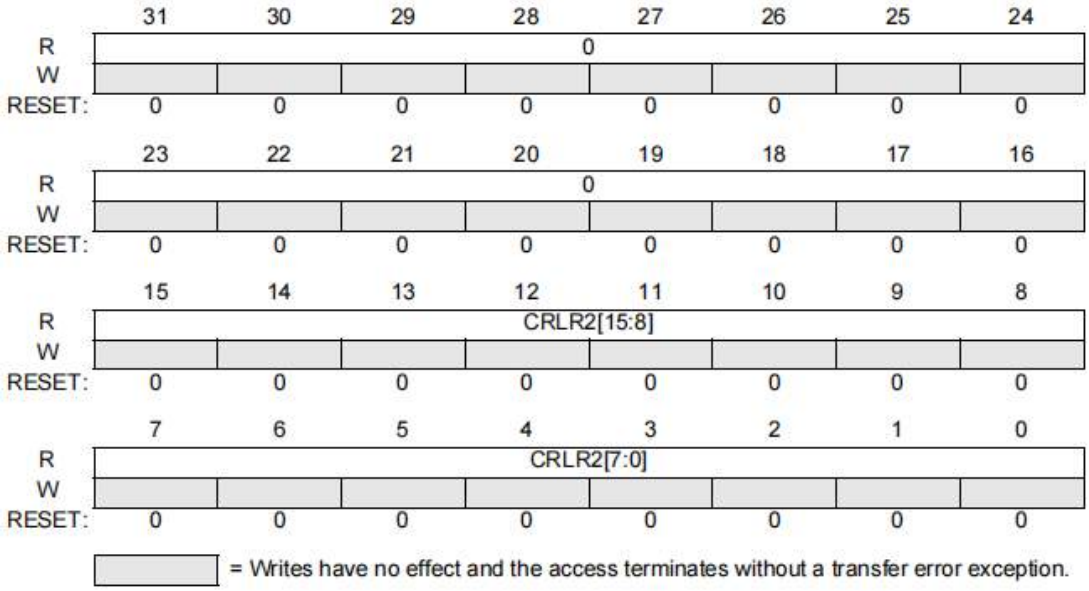
	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CRLR1[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CRLR1[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x005C and 0x005F



Offset Address: 0x0064 and 0x0067

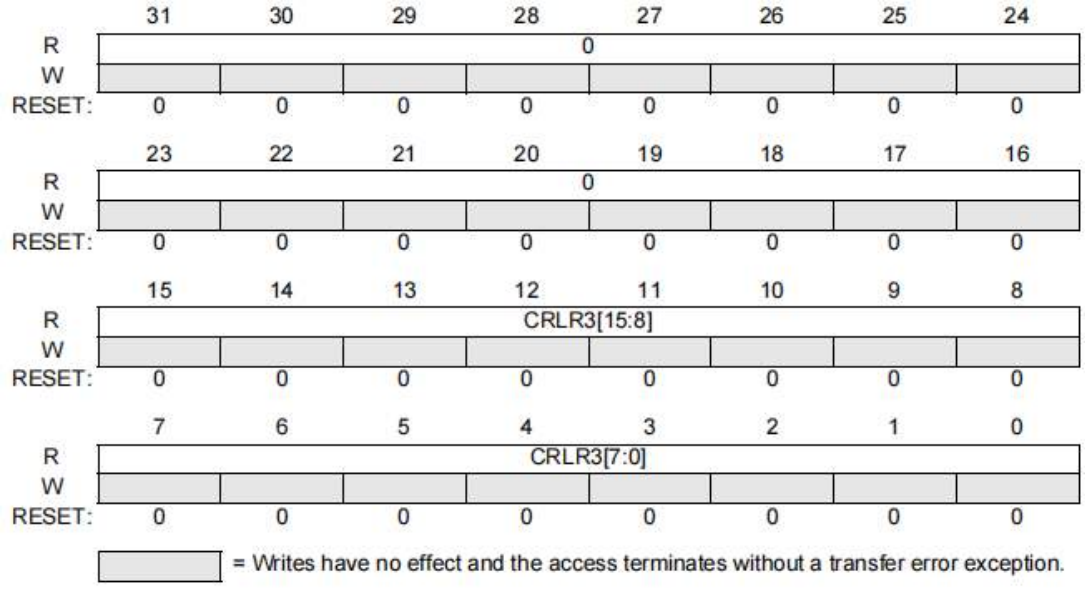


Figure 25-13: PWM Capture Rising Latch Register (PCRLR0/1/2/3)

CRLRx[15:0] — Capture Rising Latch Registerx

Latch the PWM counter when Channel x has rising transition.

25.5.2.11. PWM Capture Falling Latch Register (PCFLR0/1/2/3)

These registers are used to latch the PWM counter when capture falling transition.

Offset Address: 0x0050 and 0x0053

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CFLR0[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLR0[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x0058 and 0x005B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CFLR1[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLR1[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x0060 and 0x0063

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CFLR2[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLR2[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Offset Address: 0x0068 and 0x006B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CFLR3[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLR3[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 25-14: PWM Capture Falling Latch Register (PCFLR0/1/2/3)

CFLRx[15:0] — Capture Falling Latch Registerx

Latch the PWM counter when Channel x has falling transition.

25.5.2.12. PWM Port Control Register (PPCR)

The register*(PPCR)* is used to control PWMx pin direction and pin status.

Offset Address: 0x006C and 0x006F

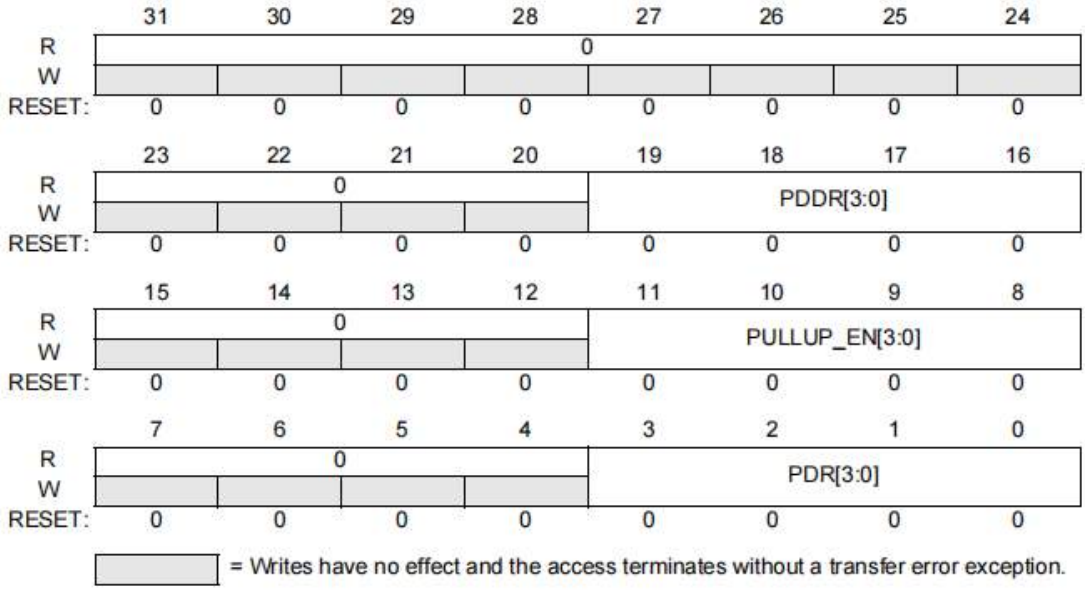


Figure 25-15: PWM Port Control Register (PPCR)

NOTE: *PULLUP_EN is meaningless for this device.*

PDDR[3:0] — Port Data Direction Register

The PDDR[3:0] bits control the direction of PWM Pins. Reset clear PDDR[3:0].

1 = Corresponding pin configured as output

0 = Corresponding pin configured as input

PULLUP_EN[3:0] — Port Pull up Enable(These bits are not implemented in this chip)

The PULLUP_EN[3:0] bits control the pull-up characteristic of PWM Pins.

1 = Enable pull up

0 = Disable pull up

PDR[3:0] — Port Data Register

Data written to PDR[3:0] drives pins only when they are configured as general-purpose outputs. Reading an input (PDDR bit clear) returns the pin level.

25.6. Functional Descriptions

This subsection describes the PWM functional operation.

25.6.1. PWM Double Buffering and Automatic Reload

PWM-Timers have a double buffering function, enabling the reload value changed for next timer operation without stopping current timer operation. Although new timer value is set, current timer operation still operate successfully. The counter value can be written into CNR0~3 and current counter value can be read from PTR0~3. The auto-reload operation will copy from CNR0~3 to down-counter when down-counter reaches zero. If CNR0~3 are set as zero, counter will be halt when counter count to zero. If auto-reload bit is set as zero, counter will be stopped immediately.

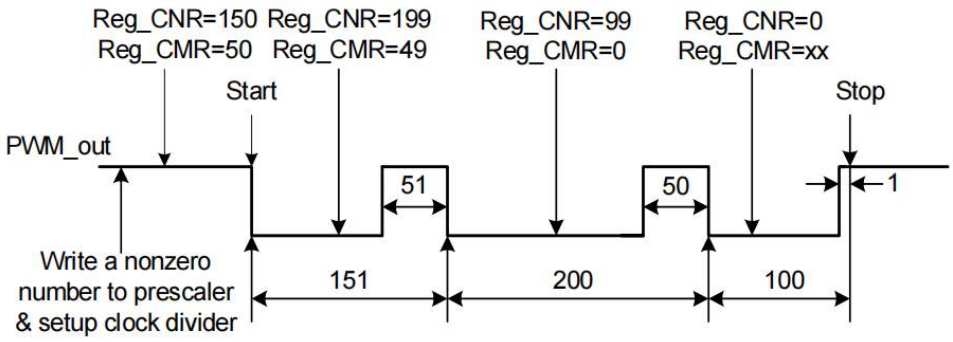


Figure 25-16: PWM Double Buffering Illustration

25.6.2. Modulate Duty Ratio

The double buffering function allows CMR written at any point in current cycle. The loaded value will take effect from next cycle.

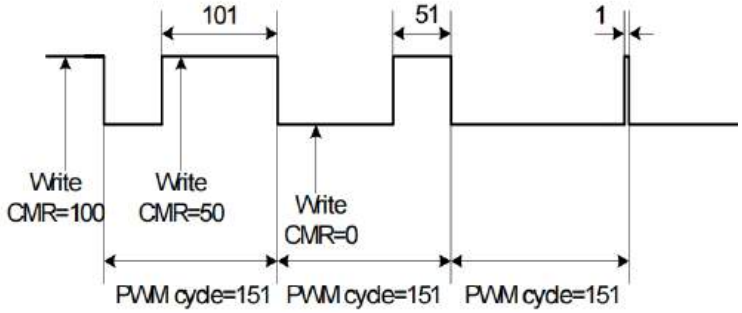


Figure 25-17: PWM Controller Output Duty Ratio

25.6.3. Dead-Zone Generator

PWM is implemented with Dead Zone generator. They are built for power device protection. This function enables generation of a programmable time gap at the rising of PWM output waveform. User can program PPR [31:24] and PPR [23:16] to determine the two Dead Zone interval respectively.

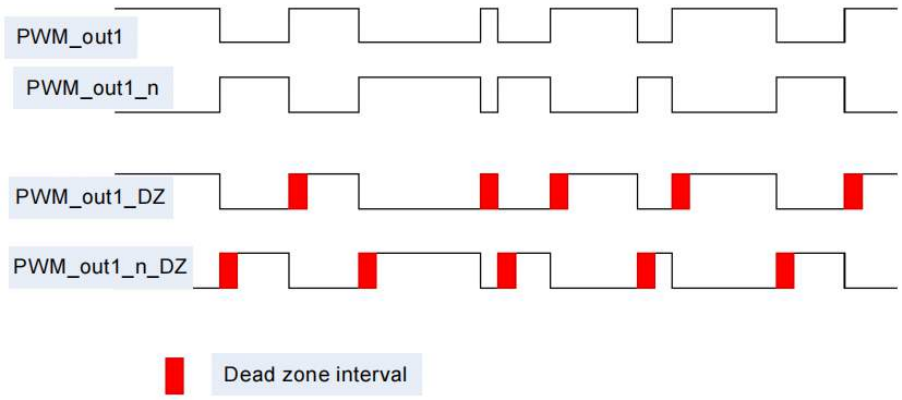


Figure 25-18: Dead Zone Generation Operation

25.6.4. PWM Timer Start Procedure

1. Setup clock selector (CSR)
2. Setup prescaler & dead zone interval (PPR)
3. Setup inverter on/off, dead zone generator on/off, toggle mode /one-shot mode, and PWM timer off. (PCR)
4. Setup the comparator register (CMR)
5. Setup the counter register (CNR)
6. Setup the interrupt enable register (PIER)
7. Setup PWMx as output pin (PPCR)
8. Enable PWM timer (PCR)

25.6.5. PWM Timer Stop Procedure

Method 1:

Set 16bit down counter (CNR) as 0, and monitor PTR. When PTR reaches to 0, disable PWM timer (PCR). (Recommended)

Method 2:

Set 16bit down counter (CNR) as 0. When interrupt request happen, disable PWM timer (PCR). (Recommended)

Method 3:

Disable PWM timer directly (PCR). (Not recommended)

25.6.6. Capture Start Procedure

1. Setup clock selector (CSR)
2. Setup pre-scale (PPR)
3. Setup inverter on/off, dead zone generator on/off, auto-load mode/one-shot mode, and PWM timer off. (PCR)
4. Setup the counter register (CNR)
5. Setup the capture register (CCR)
6. Setup PWMx as input pin (PPCR)
7. Enable PWM timer (PCR)

25.6.7. Capture Basic Timer Operation

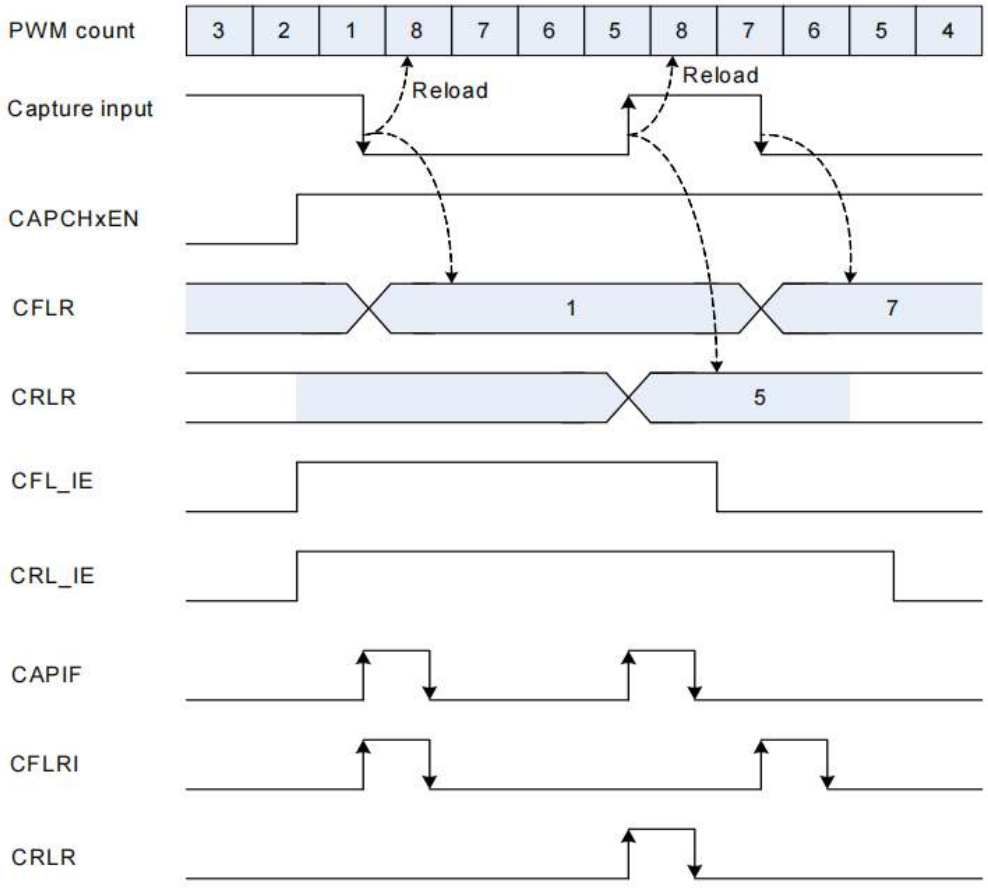


Figure 25-19: Capture Basic Timer Operation

At this case, the CNR is 8:

1. When CAPIFx set 1, the PWM counter CNRx will be reload.
2. The channel low pulse width is (CNR +1 – CRLR).
3. The channel high pulse width is (CNR+1 – CFLR)

26. Comparator Modules (COMP)

26.1. Introduction

The Comparator offers programmable response time, hysteresis, an analog input multiplexer, and two outputs that are optionally available at the Port pins: a synchronous “Filtered” output (CP), or an asynchronous “raw” output (CPA). The asynchronous CPA signal is available even when in when the system clock is not active. This allows the Comparator to operate and generate an output with the device in STOP mode.

26.2. Block Diagram

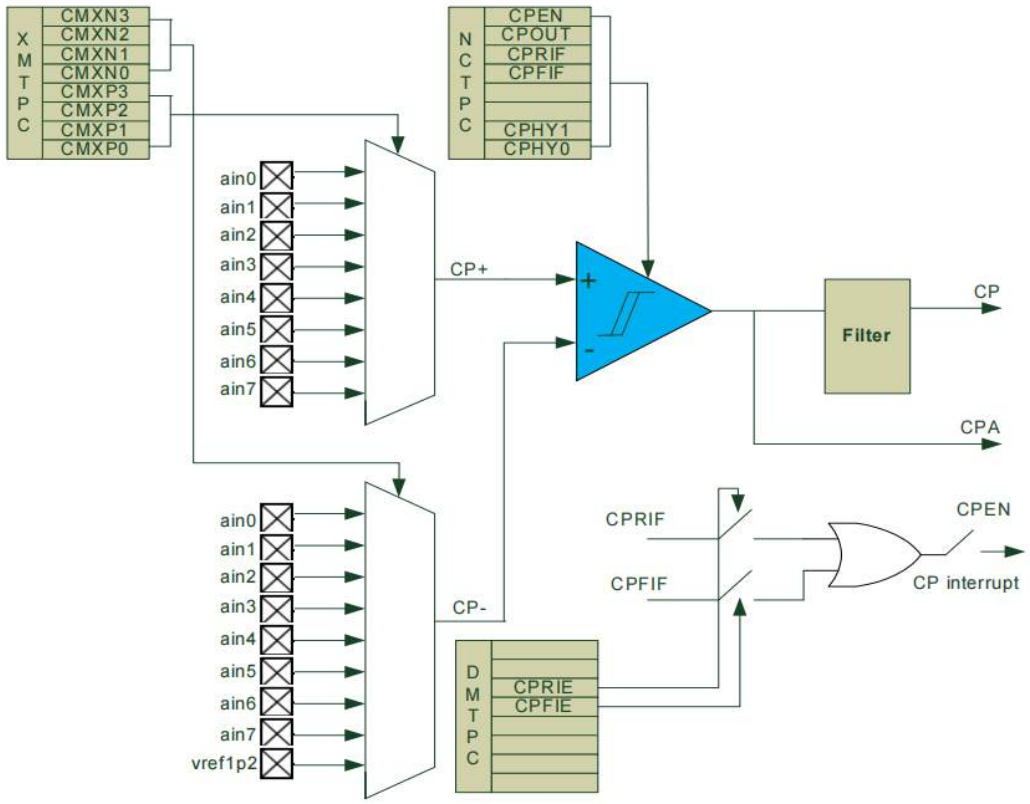


Figure 26-1: Comparator Block Diagram

26.3. Modes of Operation

This subsection describes the three low-power modes.

26.3.1. Wait Mode

In wait mode, the Comparator module can be continued to operate normally by setting CPEN bit and can be configured to exit the low-power mode by generating an interrupt request.

26.3.2. Doze Mode

In wait mode, the Comparator module can be continued to operate normally by setting CPEN bit and can be configured to exit the low-power mode by generating an interrupt request.

26.3.3. Stop Mode

In stop mode, the system clock is absent, and the Comparator module can be continued to operate normally by setting CPEN bit and can be configured to exit the low-power mode by generating an asynchronous “raw” output (CPA) wakeup signal.

26.4. Memory Map and Registers

This subsection describes the memory map and register structure for Comparator.

26.4.1. Memory Map

Refer to **Table 26-1** for a description of the memory map. This device has two Comparator modules.

Table 26-1: Comparator Module Memory Map

Offset Address	Bits 7-0	Access(1)
0x0003	Comparator Control Register(CPTCN)	S/U
0x0002	Comparator0 Mode Selection Register(CPTMD)	S/U
0x0001	Comparator MUX Selection Register(CPTMX)	S/U
0x0000	Comparator Output Filter Selection Register(CPTFS)	S/U

NOTE:

S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.

26.4.2. Registers

The Comparator programming model consists of these registers:

- **CPTCN:** Comparator Control Register.
- **CPTMD:** Comparator Mode Selection Register.
- **CPTMX:** Comparator MUX Selection Register.
- **CPTFLS:** Comparator Output Filter Selection Register.

26.4.2.1. Comparator Control Register

Offset Address: 0x0003

	7	6	5	4	3	2	1	0
R	CPEN	CPOUT	CPRIF	CPFIF	0	0	CPHY[1:0]	
W								
RESET:	0	0	0	0	0	0	0	1

Figure 26-2: Comparator Control Register (CPTCN)

CPEN — Comparator Enable Bit

The read/write CPEN bit enables Comparator operation. When the Comparator is disabled, CPOUT is low state.

1 = Comparator enabled

0 = Comparator disabled

CPOUT — Comparator Output State Flag.

1 = Voltage on CP+ > CP–.

0 = Voltage on CP+ < CP–.

CPRIF — Comparator Rising-Edge Flag. Must be cleared by software writing one to this bit.

1 = Comparator Rising Edge has occurred.

0 = No Comparator Rising Edge has occurred since this flag was last cleared.

CPFIF — Comparator Falling-Edge Flag. Must be cleared by software writing one to this bit.

1 = Comparator Falling-Edge has occurred.

0 = No Comparator Falling-Edge has occurred since this flag was last cleared.

CPHY[1:0] — Comparator Hysteresis Control Bits.

00: Hysteresis Disabled.

01: Hysteresis = 8 mV.

10: Hysteresis = 12 mV.

11: Hysteresis = 15.4 mV.

26.4.2.2. Comparator Mode Selection Register

Offset Address: 0x0002

	7	6	5	4	3	2	1	0
R	CPOEN	0	CPRIE	CPFIE	0	WKUP-LEVEL	0	CPMD
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 26-3: Comparator Mode Selection Register(CPTMD)

CPOEN — Comparator Output to pad control bit.

1 = Comparator output can be observed in pwm0[0] for COMP0 and pwm0[1] for COMP1(COMP1 is not implemented in this device).

0 = Comparator output is disabled.

CPRIE — Comparator Rising-Edge Interrupt Enable.

1 = Comparator Rising-edge interrupt enabled.

0 = Comparator Rising-edge interrupt disabled.

CPFIE — Comparator Falling-Edge Interrupt Enable.

1 = Comparator Falling-edge interrupt enabled.

0 = Comparator Falling-edge interrupt disabled.

WKUPLEVEL — Comparator Wake Up level control bit.

1 = Voltage on CP+ > CP– will generate a wakeup request during stop mode.

0 = Voltage on CP+ < CP– will generate a wakeup request during stop mode.

CPMD — Comparator Mode Select. These bits select the response time for Comparator.

Table 26-2: Comparator Mode Selection

Mode	CPMD	Response Time	Power Consumption
Low-speed	0	500ns	3.3uA
High-speed	1	80ns	22uA

26.4.2.3. Comparator MUX Selection Register

Offset Address: 0x0001

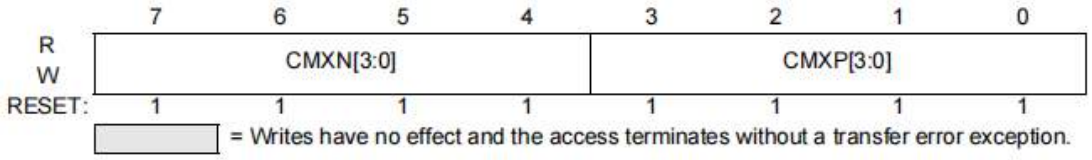


Figure 26-4: Comparator MUX Selection Register(CPTMX)

CMXN[3:0] — Comparator Negative Input MUX Select. These bits select which Port pin is used as the Comparator negative input.

Table 26-3: Comparator Negative Input MUX Selection

CMXN[3]	CMXN[2]	CMXN[1]	CMXN[0]	Negative Input
0	0	0	0	ain0
0	0	0	1	ain1
0	0	1	0	ain2
0	0	1	1	ain3
0	1	0	0	ain4

CMXN[3]	CMXN[2]	CMXN[1]	CMXN[0]	Negative Input
0	1	0	1	ain5
0	1	1	0	ain6
0	1	1	1	ain7
1	0	0	0	vref1p2
Other value				None

CMXP[3:0] — Comparator Positive Input MUX Select. These bits select which Port pin is used as the Comparator positive input.

Table 26-4: Comparator Positive Input MUX Selection

CMXN[3]	CMXN[2]	CMXN[1]	CMXN[0]	Positive Input
0	0	0	0	ain0
0	0	0	1	ain1
0	0	1	0	ain2
0	0	1	1	ain3
0	1	0	0	ain4
0	1	0	1	ain5
0	1	1	0	ain6
0	1	1	1	ain7
1	x	x	x	None

26.4.2.4. Comparator Output Filter Selection Register

Offset Address: 0x0000

	7	6	5	4	3	2	1	0
R W	CMFLEN	CMFL_WIDTH[6:0]						
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

Figure 26-5: Comparator Output Filter Selection Register(CPTFLS)

CMFLEN — Comparator Output Digital Filter Enable

1 = Comparator Output digital filter enable and the CPMRIF and CPMFIF is generated by filter output;

0 = Comparator Output digital filter disable and the CPMRIF and CPMFIF is generated by raw output;

CMFL_WIDTH[6:0] — Comparator Output Digital Filter Pulse Width Selection.

CMFL_WIDTH[6:0] determine the width of input pulse will be filtered. If the input pulse width less than (CMFL_WIDTH[6:0]+2) * Period of f_{ips}, the pulse will be filtered.

26.5. Function Description

The Comparator inputs are selected in the CPTMX register. The CMXP3 ~ CMXP0 bits select the Comparator positive input; the CMXN3–CMXN0 bits select the Comparator negative input. **Important Note About Comparator Inputs:** The Port pins selected as comparator inputs should be configured as analog inputs in their associated Port configuration register and the input, output, pullup and pulldown control should be disabled.

The Comparator output can be polled in software, used as an interrupt source, and/or routed to a Port pin. When routed to a Port pin, the Comparator output is available asynchronous or synchronous to the system clock; the asynchronous output is available even in STOP mode (with no system clock active). When disabled, the Comparator output defaults to the logic low state, and its supply current falls to less than 100nA.

The Comparator hysteresis is software-programmable via its Comparator Control register CPTCN. The user can program the amount of hysteresis voltage. The Comparator hysteresis is programmed using Bits1 ~ 0 in the Comparator Control Register CPTCN.

Comparator interrupts can be generated on both rising-edge and falling-edge output transitions. The CPFIF flag is set to logic 1 upon a Comparator falling-edge occurrence, and the CPRIF flag is set to logic 1 upon the Comparator rising-edge occurrence. Once set, these bits remain set until cleared by software. The Comparator rising-edge interrupt mask is enabled by setting CPRIE to a logic 1. The Comparator falling-edge interrupt mask is enabled by setting CPFIE to a logic 1.

The output state of the Comparator can be obtained at any time by reading the CPOUT bit. The Comparator is enabled by setting the CPEN bit to logic 1, and is disabled by clearing this bit to logic 0.

Note that false rising edges and falling edges can be detected when the comparator is first powered on or if changes are made to the hysteresis control bits. Therefore, it is recommended that the rising-edge and falling-edge flags be explicitly cleared to logic 0 a short time after the comparator is enabled or its mode bits have been changed. This least start-up time of the comparator is less than 5us.

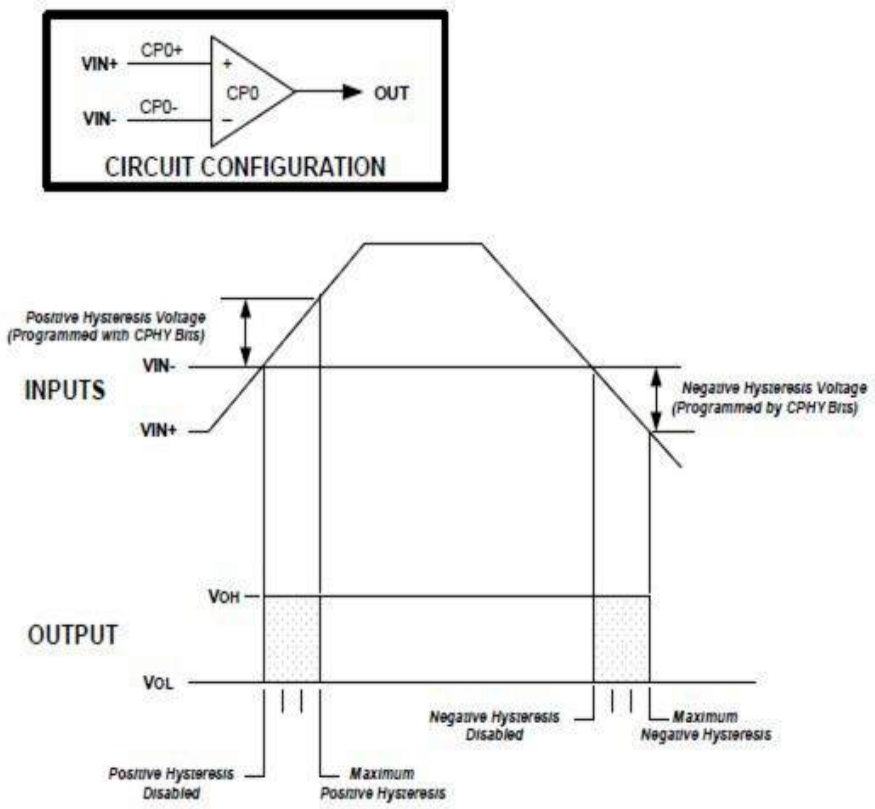


Figure 26-6: Comparator Hysteresis Plot

27. Analog-to-Digital Converter (ADC)

27.1. Introduction

The 12bit ADC is a successive approximation analog-to-digital converter. It has up to 4 channels allowing it to measure signals from 3 external and 1 internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The results of the ADC are stored in a 12bit x 8depth FIFO, and the data format can be left-aligned or right-aligned.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low power mode is implemented to allow very low consumption at low frequency.

27.2. ADC Main Features

- High performance
  - 12bit, 10bit, 8bit or 6bit configurable resolution
  - ADC conversion time: 1.0 μs for 12bit resolution (1 MHz), 0.88 μs conversion time for 10 bit resolution, faster conversion times can be obtained by lowering resolution.
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support
- Low power
  - Application can reduce PLCK frequency for low power operation while still keeping optimum ADC performance. For example, 1.0 μs conversion time is kept, whatever the frequency of PCLK.
  - Wait mode: prevents ADC overrun in applications with low frequency PLCK
  - Auto off mode: ADC is automatically powered off except during the active conversion phase. This dramatically reduces the power consumption of the ADC.
- Analog input channels
  - 8 external analog inputs
  - 1 channel for internal reference voltage
  - 1 channel for internal temperature sensor
- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity
- Conversion modes
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events.

- Analog watchdog
- Single-ended and differential-input configurations
- Converter uses an internal reference or an external reference

27.3. ADC Functional Description

Figure 27-1 shows the ADC block diagram.

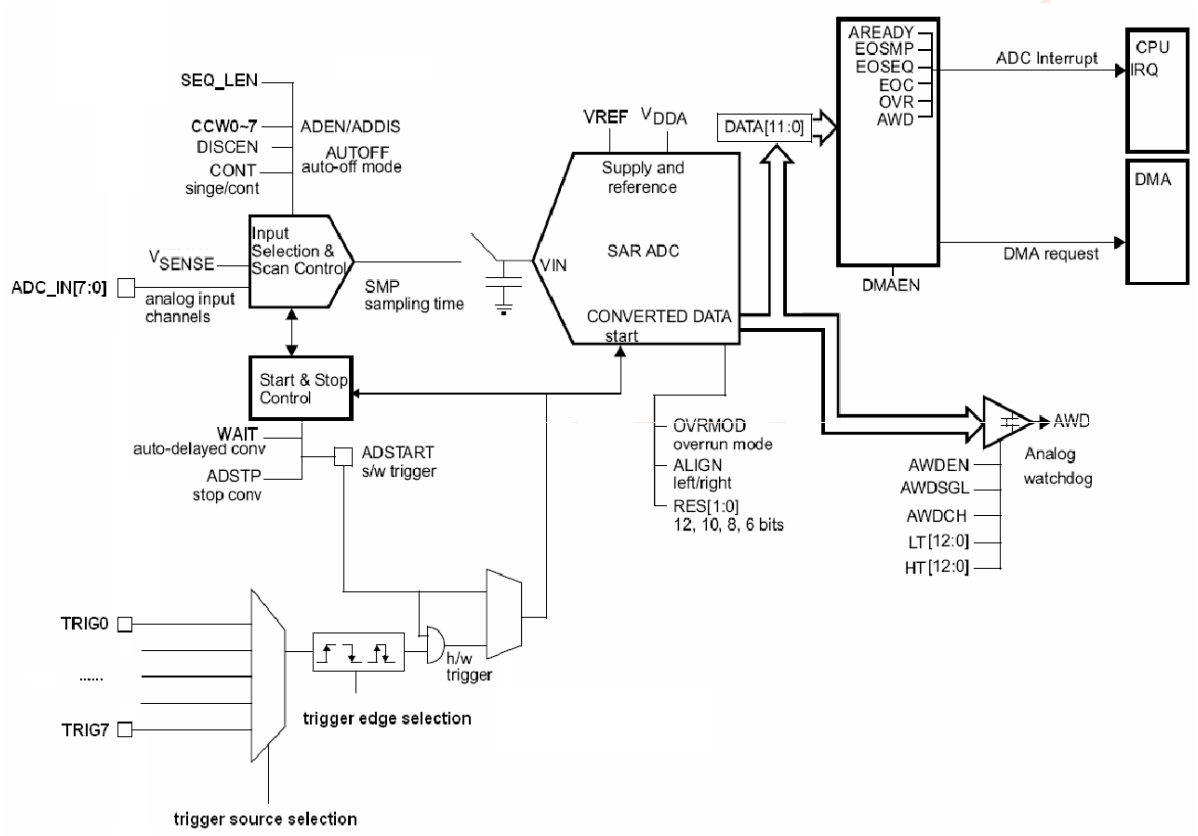


Figure 27-1: ADC Block Diagram

27.3.1. ADC On-Off Control (ADEN, ADDIS, ADRDY)

At MCU power-up, the ADC is disabled and put in power-down mode (ADEN=0). As shown in **Figure 27-2**, the ADC needs a stabilization time of tSTAB(~2.0 μs) before it starts converting accurately.

Two control bits are used to enable or disable the ADC:

- Set ADEN=1 to enable the ADC. The ADRDY flag is set as soon as the ADC is ready for operation.
- Set ADDIS=1 to disable the ADC and put the ADC in power down mode. The ADEN and ADDIS bits are then automatically cleared by hardware as soon as the ADC is fully disabled.

Conversion can then start either by setting ADSTART=1 or when an external trigger event occurs if triggers are enabled.

Follow this procedure to enable the ADC:

- Set ADEN=1 in the ADC_CR register.
- Wait until ADRDY=1 in the ADC_ISR register (ADRDY is set after the ADC startup time). This can be handled by interrupt if the interrupt is enabled by setting the ADRDYIE bit in the ADC_IER register.

Follow this procedure to disable the ADC:

- Check that ADSTART=0 in the ADC_CR register to ensure that no conversion is ongoing. If required, stop any ongoing conversion by writing 1 to the ADSTP bit in the ADC_CR register and waiting until this bit is read at 0.
- Set ADDIS=1 in the ADC_CR register.
- If required by the application, wait until ADEN=0 in the ADC_CR register, indicating that the ADC is fully disabled (ADDIS is automatically reset once ADEN=0).

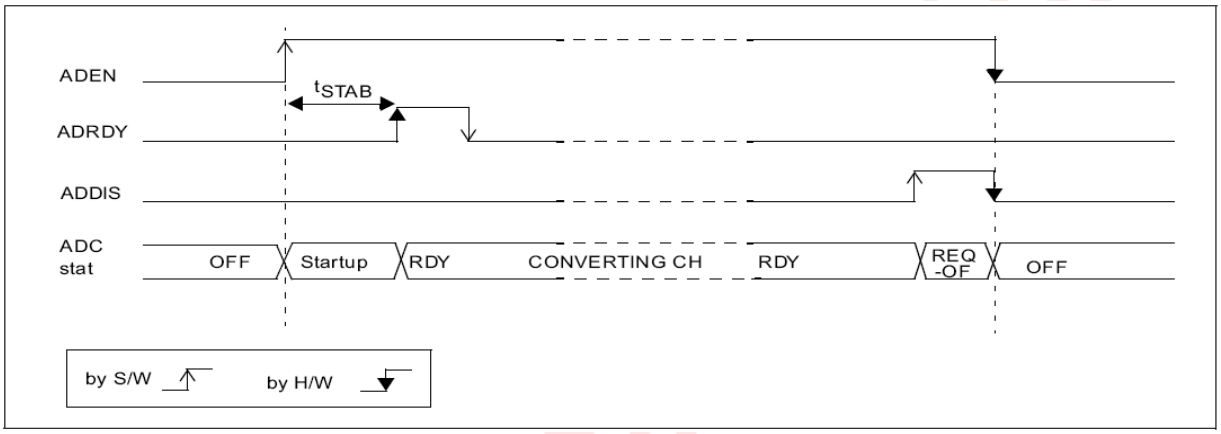


Figure 27-2: Enabling/Disabling the ADC

NOTE: *In auto-off mode (AUTOFF=1) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.*

27.3.2. ADC Clock

The ADC has a dual clock-domain architecture, as show in **Figure 27-3**, this has the advantage of reaching the maximum ADC clock frequency whatever the IPG clock scheme selected.

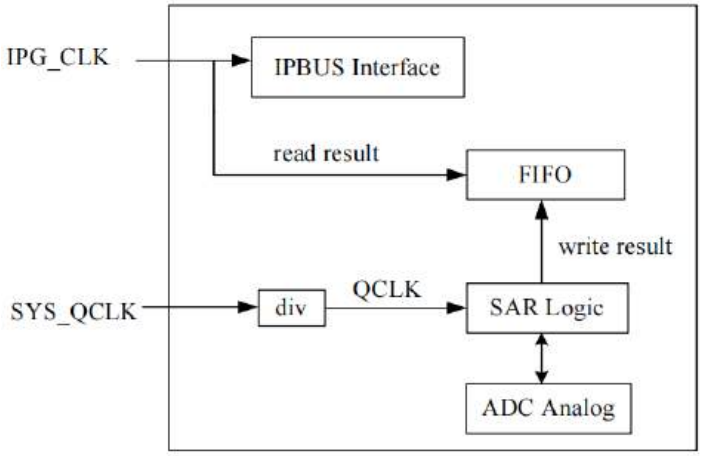


Figure 27-3: ADC Clock Scheme

27.3.3. Configuring the ADC

Software must write to the ADEN bit in the ADC_CR register if the ADC is disabled (ADEN must be 0).

Software must only write to the ADSTART and ADDIS bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC_IER, ADC_CFGRi, ADC_SMPR, ADC_TR, ADC_CHSELRi and ADC_WDG registers, software must only write to the configuration control bits if there is no conversion ongoing (ADSTART = 0).

Software must only write to the ADSTP bit in the ADC_CR register if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (ADSTART = 1 and ADDIS = 0).

NOTE: *There is no hardware protection preventing software from making write operations forbidden by the above rules. If such a forbidden write access occurs, the ADC may enter an undefined state. To recover correct operation in this case, the ADC must be disabled(ADDIS = 1).*

27.3.4. Channel Selection (CCWi)

There are up to 10 multiplexed channels:

- 8 analog inputs from pins (ADC_IN0...ADC_IN7)
- 2 internal analog input (VREFINT & Temperature Sensor)

It is possible to convert a single channel or to automatically scan a sequence of channels.

The sequence of the channels to be converted is organized as CCW[0], then CCW[1], ... ,then CCW[7]. The CCWi are programed in ADC_CHSELRi. The sequence length is programmed in SEQ_LEN[2:0] of ADC_CFGR2. For example, if sequence length is set as 3, then the sequence is organized as CCW[0], then CCW[1], then CCW[2].

The channel decode is shown in **Table 27-1**.

Table 27-1: Channel Decode

CCWi[3:0]	Channel Select
4'b0000	ADC_IN0
4'b0001	ADC_IN1
4'b0010	ADC_IN2
4'b0011	ADC_IN3
4'b0100	ADC_IN4
4'b0101	ADC_IN5
4'b0110	ADC_IN6
4'b0111	ADC_IN7
4'b1110	VREFINT
4'b1111	Temperature Sensor

27.3.5. Programmable Sampling Time (SMP)

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level.

Having a programmable sampling time allows to trim the conversion speed according to the input resistance of the input voltage source. The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP[3:0] bits in the ADC_SMPR register. This programmable sampling time is common to all channels.

The ADC indicates the end of the sampling phase by setting the EOSMP flag.

27.3.6. Single Conversion Mode (CONT=0)

In Single conversion mode, the ADC converts the sequence once. This mode is selected when CONT=0 in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the FIFO
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOSEQ (end of sequence) flag is set
- An interrupt is generated if the EOSEQIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

NOTE: *To convert a single channel once, program a sequence with a length of 1.*

27.3.7. Continuous Conversion Mode (CONT=1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting the sequence once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT=1 in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the FIFO
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOSEQ (end of sequence) flag is set
- An interrupt is generated if the EOSEQIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

NOTE: *It is not possible to have both discontinuous mode and continuous mode enabled:*

it is forbidden to set both bits DISCEN=1 and CONT=1.

27.3.8. Starting Conversions (ADSTART)

Software starts ADC conversions by setting ADSTART=1. When ADSTART is set, the conversion:

- Starts immediately if TRIGMODE = 0x0 (software trigger)
- At the next active edge of the selected hardware trigger if TRIGMODE ≠ 0x0

The ADSTART bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0, indicating that the ADC is idle.

The ADSTART bit is cleared by hardware:

- In single mode with software trigger
  - At any end of conversion sequence (EOSEQ=1)
- In discontinuous mode with software trigger
  - At any end of conversion
- In all cases
  - After execution of the ADSTP procedure invoked by software

NOTE:

- *In continuous mode (CONT=1), the ADSTART bit is not cleared by hardware when the EOSEQ flag is set because the sequence is automatically relaunched.*
- *When hardware trigger is selected in single mode, ADSTART is not cleared by hardware when the EOSEQ flag is set. This avoids the need for software having to set the ADSTART bit again and ensures the next trigger event is not missed.*

27.3.9. Timings

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

tADC = tSMPL + tSAR = [4|min + 12|12bit] x tQCLK = 1µs |min (for fQCLK = 16MHz)

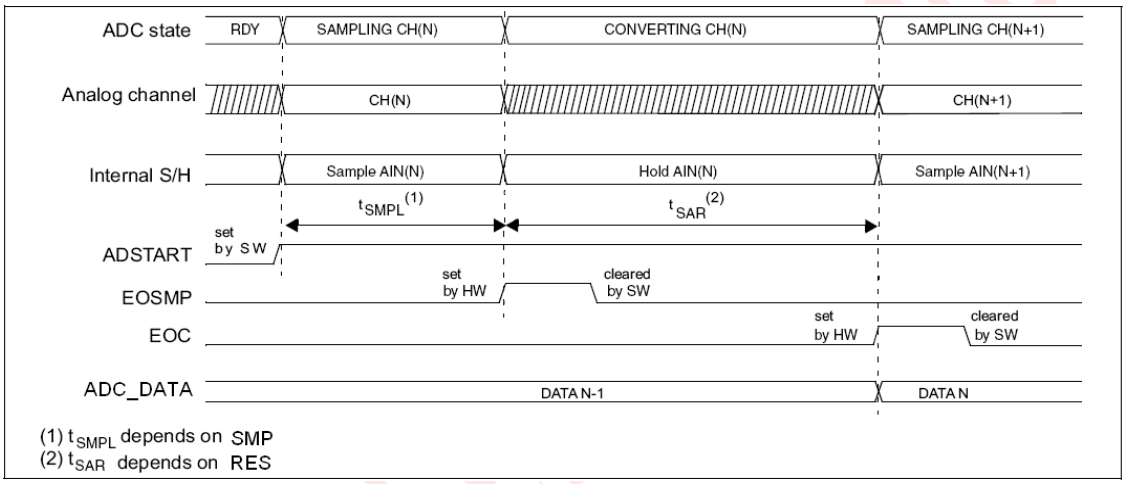


Figure 27-4: Analog to Digital Conversion Time

27.3.10. Stopping an Ongoing Conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP=1 in the ADC_CR register.

This will reset the ADC operation and the ADC will be idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted and the result is discarded (FIFO is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that restarting the ADC would re- start a new sequence).

Once this procedure is complete, the ADSTP and ADSTART bits are both cleared by hardware and the software must wait until ADSTART=0 before starting new conversions.

NOTE: *The flags in QADC_ISR are not cleared by STOP command, and the data in FIFO are not lost.*

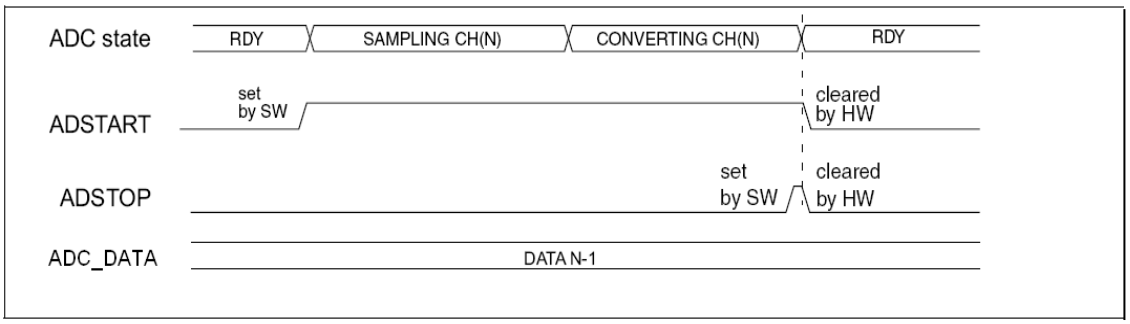


Figure 27-5: Stopping an Ongoing Conversion

27.4. Conversion on External Trigger and Trigger Polarity (TRIGMODE, TRIGSCR)

A conversion or a sequence of conversion can be triggered either by software or by an external event. If the TRIGMODE control bits are not equal to "0", then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART=1.

Any hardware triggers which occur while a conversion is ongoing are ignored. If bit ADSTART=0, any hardware triggers which occur are ignored.

Table 27-2 provides the correspondence between the TRIGMODE values and the trigger polarity.

Table 27-2: Configuring the Trigger Polarity

TRIGMODE[2:0]	Source
3'b000	Trigger detection disabled, software trigger
3'b001	Detection on rising edge
3'b010	Detection on falling edge
3'b011	Detection on both rising and falling edges
3'b100	Detection on high level voltage
3'b101	Detection on low level voltage
3'b110	Detection on PIT 0
3'b111	Detection on PWM 0

NOTE: *The polarity of the external trigger can be changed only when the ADC is not converting (ADSTART= 0).*

The TRIGSCR control bits are used to select which of 8 possible events can trigger conversions. **Table 27-3** gives the possible external trigger for regular conversion. Software source trigger events can be generated by setting the ADSTART bit in the ADC_CR register.

Table 27-3: External Triggers

TRIGSCR[2:0]	Name	Source
3'b000	TRG0	eport1[0]
3'b001	TRG1	eport1[1]
3'b010	TRG2	eport1[2]
3'b011	TRG3	eport1[3]

TRIGSCR[2:0]	Name	Source
3'b100	TRG4	eport1[4]
3'b101	TRG5	eport1[5]
3'b110	TRG6	eport1[6]
3'b111	TRG7	eport1[7]

NOTE: *The trigger selection can be changed only when the ADC is not converting (ADSTART= 0).*

27.4.1. Discontinuous Mode (DISCEN)

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

In this mode (DISCEN=1), a hardware or software trigger event is required to start each conversion defined in the sequence. On the contrary, if DISCEN=0, a single hardware or software trigger event successively starts all the conversions defined in the sequence.

Example:

- DISCEN=1, channels to be converted = 0, 3, 7, 10
  - 1st trigger: channel 0 is converted and an EOC event is generated
  - 2nd trigger: channel 3 is converted and an EOC event is generated
  - 3rd trigger: channel 7 is converted and an EOC event is generated
  - 4th trigger: channel 10 is converted and both EOC and EOSEQ events are generated.
  - 5th trigger: channel 0 is converted an EOC event is generated
  - 6th trigger: channel 3 is converted and an EOC event is generated
  - ...
- DISCEN=0, channels to be converted = 0, 3, 7, 10
  - 1st trigger: the complete sequence is converted: channel 0, then 3, 7 and 10. Each conversion generates an EOC event and the last one also generates an EOSEQ event.
  - Any subsequent trigger events will restart the complete sequence.

27.4.2. Programmable Resolution (RES) - Fast Conversion Mode

It is possible to obtain faster conversion times (tSAR) by reducing the ADC resolution. The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

NOTE: *The RES[1:0] bit must only be changed when the ADEN bit is reset.*

The result of the conversion is always 13 bits wide and any unused LSB bits are read as zeroes.

Lower resolution reduces the conversion time needed for the successive approximation steps.

27.4.3. End of Conversion, End Of Sampling Phase (EOC, EOSMP Flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC_ISR register as soon as a new conversion data result is available. An interrupt can be generated if the EOCIE bit is set in the ADC_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the FIFO.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC_ISR register. The EOSMP flag is cleared by software by writing1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC_IER register.

27.4.4. End of Conversion Sequence (EOSEQ Flag)

The ADC notifies the application of each end of sequence (EOSEQ) event.

The ADC sets the EOSEQ flag in the ADC_ISR register as soon as the last data result of a conversion sequence is available in the FIFO. An interrupt can be generated if the EOSEQIE bit is set in the ADC_IER register. The EOSEQ flag is cleared by software by writing 1 to it.

27.4.5. Example Timing Diagrams (Single/Continuous Modes Hardware / Software Triggers)

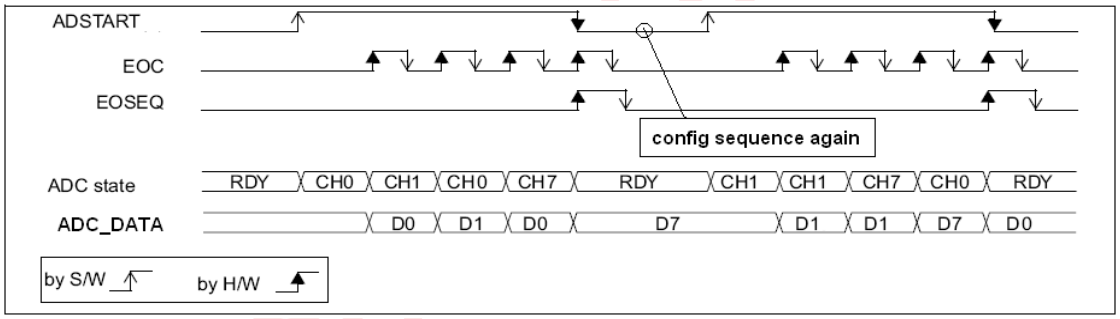


Figure 27-6: Single Conversions of A Sequence, Software Trigger

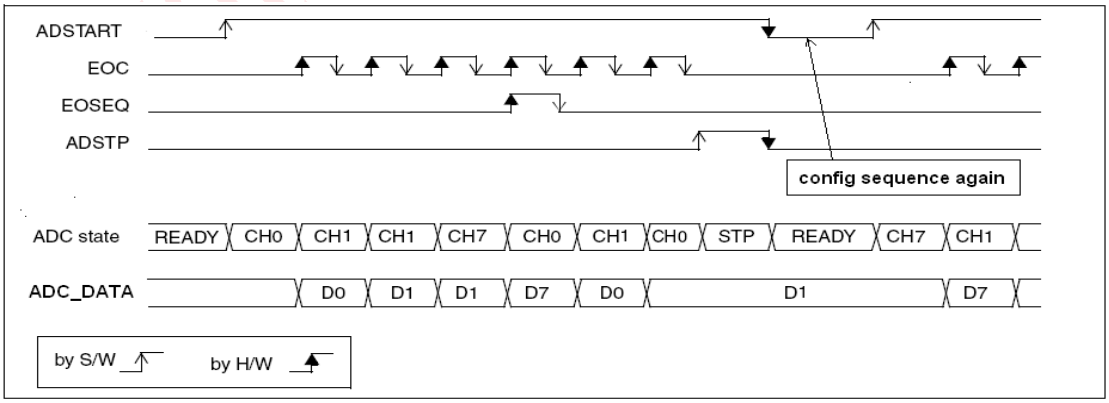


Figure 27-7: Continuous Conversion of A Sequence, Software Trigger

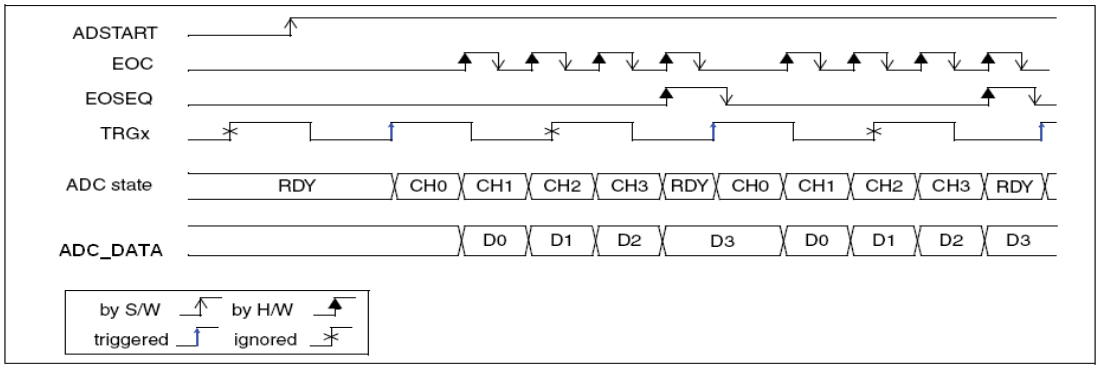


Figure 27-8: Single Conversions of A Sequence, Hardware Trigger

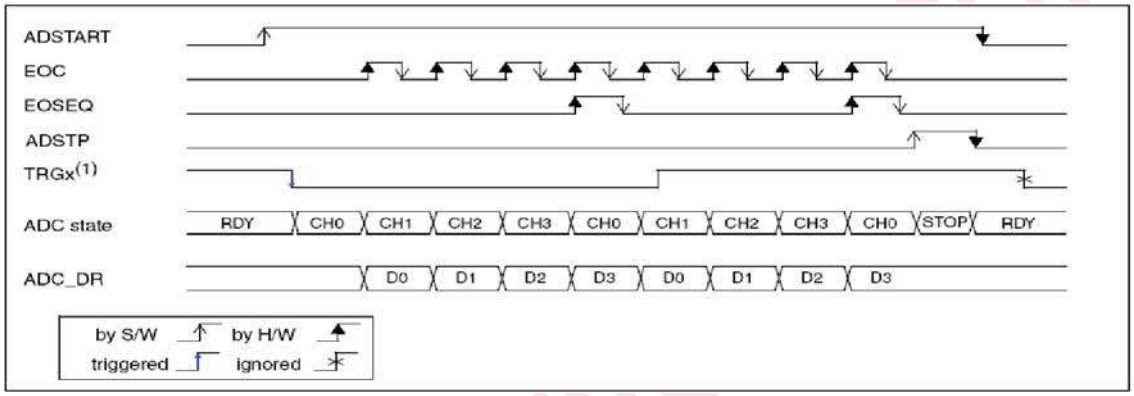


Figure 27-9: Continuous Conversions of A Sequence, Hardware Trigger

27.5. Data Management

27.5.1. Data FIFO & Data Alignment (ADC_FIFO, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC_FIFO which is 13bit wide x 8 depth.

The format of the read out data depends on the configured data alignment and resolution.

The ALIGN bit in the ADC_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN=0) or left-aligned (ALIGN=1) as shown in **Figure 27-10**.

ALIGN	RES[1:0]	31	30	…	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x0			…					data[11:0]											
	0x1			…							data[9:0]									
	0x2			…									data[7:0]							
	0x3			…											data[5:0]					
1	0x0			…	data[11:0]															
	0x1			…					data[9:0]											
	0x2			…									data[7:0]							
	0x3			…									data[5:0]							

Figure 27-10: Data Alignment And Resolution

The FIFO supports byte, half-word and word read, but the address offset should be always 0x004C.

For different data alignments and resolutions, users should note:

- If read by word, then the data format is as data[31:0] as **Figure 27-10** shows
- if read by half word, then the data format is as data[15:0] as **Figure 27-10** shows
- If read by byte when the data is longer than 8bit, then the high byte is first read out, the low byte should read again.
- If read by byte when the data is no longer than 8bit, then the data format is as data[7:0] as **Figure 27-10** shows

27.5.2. ADC Overrun (OVR, OVRMOD)

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA, before the FIFO is full.

The OVR flag is set in the ADC_ISR register if the FULL flag is still at '1' at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC_IER register.

When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CR register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the OVRMOD bit in the ADC_CFGR1 register:

- OVRMOD=0
  - An overrun event preserves the data register from being overwritten: the old data is maintained and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is discarded.
- OVRMOD=1
  - The data register is overwritten with the last conversion result. If OVR remains at 1, further conversions can be performed and the FIFO always contains the data from the latest conversion.

27.5.3. Managing a Sequence of Data Converted Without Using The DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software can use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC_ISR register and the FIFO register can be read.

Software can also use FIFO EMPTY flag to handle each data result. If EMPTY is not "0", this means FIFO has new data.

The OVRMOD bit in the ADC_CFGR1 register should be configured to 0 to manage overrun events as an error.

27.5.4. Managing Converted Data Without Using The DMA Without Overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag should be ignored by the software. When OVRMOD=1, an overrun event does not prevent the ADC from continuing to convert and the FIFO always contains the latest conversion data.

27.5.5. Managing Converted Data Using The DMA

Once the data number in the FIFO is not empty and bit DMAEN is set, QADC will send request to the DMA. This allows the transfer of the converted data from the FIFO to the destination location selected by the software.

Despite this, if an overrun occurs (OVR=1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten.

The DMA transfer requests are blocked until the software clears the OVR bit.

27.6. Low power Features

27.6.1. Wait Mode Conversion

Wait mode conversion can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring. When the WAIT bit is set to 1 in the ADC_CFGR1 register, a new conversion can start only if the FIFO is not full.

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

NOTE: *Any hardware triggers which occur while a conversion is ongoing or during the wait time preceding the read access are ignored.*

27.6.2. Auto-off Mode (AUTOFF)

The ADC has an automatic power management feature which is called auto-off mode, and is enabled by setting AUTOFF=1 in the ADC_CFGR1 register.

When AUTOFF=1, the ADC is always powered off when not converting and automatically wakes-up when a conversion is started (by software or hardware trigger). A startup-time is automatically inserted between the trigger event which starts the conversion and the sampling time of the ADC. The ADC is then automatically disabled once the sequence of conversions is complete.

Auto-off mode can cause a dramatic reduction in the power consumption of applications which need relatively few conversions or when conversion requests are timed far enough apart (for example with a low frequency hardware trigger) to justify the extra power and extra time used for switching the ADC on and off.

Auto-off mode can be combined with the wait mode conversion (WAIT=1) for applications clocked at low frequency. This combination can provide significant power savings if the ADC is automatically powered-off during the wait phase and restarted as soon as the FIFO is read by the application

27.7. Analog Window Watchdog

The AWD analog watchdog feature is enabled by setting the AWDEN bit in the ADC_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels remain within a configured voltage range (window) as shown in **Figure 27-11**.

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in the ADC_TR register. An interrupt can be enabled by setting the AWDIE bit in the ADC_IER register.

The AWD flag is cleared by software by writing 1 to it.

When converting a data with a resolution of less than 12bit (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12bit raw converted data
(left aligned).

Table 27-4 shows how to configure the AWDSGL and AWDEN bits in the ADC_CFGR1 register to enable the analog watchdog on one or more channels.

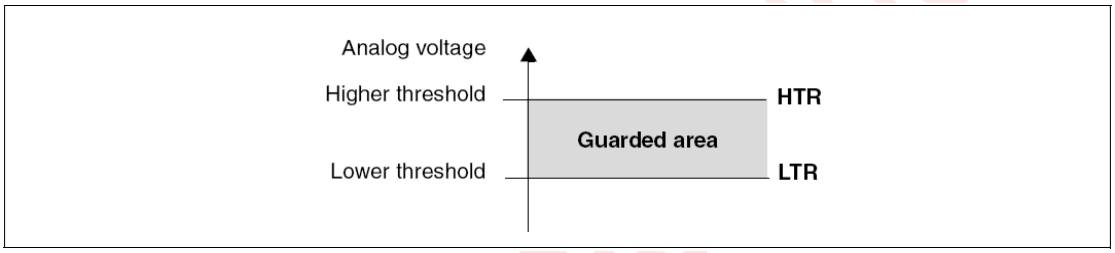


Figure 27-11: Analog Watchdog Guarded Area

Table 27-4: Analog Watchdog Channel Selection

Channels Guarded by the Analog Watchdog	AWDSGL Bit	AWDEN Bit
None	x	0
All Channels	0	1
Signal(1) Channel	1	1

(1) Selected by the AWDCH

27.8. Temperature Sensor

The temperature sensor is internally connected to the ADC1_IN15 input channel which is used to convert the sensor's output voltage to a digital value.

27.9. ADC Interrupts

An interrupt can be generated by any of the following events:

- ADC power-up, when the ADC is ready (ADRDY flag)
- End of any conversion (EOC flag)
- End of a sequence of conversions (EOSEQ flag)
- When an analog watchdog detection occurs (AWD flag)
- When the end of sampling phase occurs (EOSMP flag)
- when a data overrun occurs (OVR flag) Separate interrupt enable bits are available for flexibility.

Table 27-5: ADC Interrupts

Interrupt Event	Event Flag	Enable Control Bit
ADC Ready	ADRDY	ADRDYIE
End of Conversion	EOC	EOCIE
End of Sequence of Conversions	EOSEQ	EOSEQIE
Analog Watchdog Bit is Set	AWD	AWDIE
End of Sampling Phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE

27.10. Memory Map and Registers

This subsection describes the memory map and register structure.

27.10.1. Memory Map

Refer to **Table 27-6** for a description of the QADC memory map.

Table 27-6: QADC Memory Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_ISR																									AWD	Empty	FULL	OVR	EOSE	EOC	EOSM	ADRD
0x04	ADC_IER																									AWDIE			OVRIE	EOSEQIE	EOCIE	EOSMPI	ADRDYIE
0x08	ADC_CR																													ADSTP	ADSTAR	ADDIS	ADEN
0x0C	ADC_CFGR1	DIFF	OVRMOD				SEQ_LEN [2:0]			DISCEN	AUTOFF	WAIT	CONT		TRIGSCR [2:0]					TRIGMODE		ALIGN	RES [1:0]									DMAEN
0x10	ADC_CFGR2																					QPR [3:0]				STCNT [7:0]							
0x14	ADC_SMPR																									SMP [7:0]							
0x18	ADC_WDG																									AWDEN	AWDSG			AWDCH [3:0]			
0x1C	ADC_TR					HT [11:0]																LT [11:0]											
0x2C	ADC_CHSELR1					CCW3 [3:0]								CCW2 [3:0]								CCW1 [3:0]								CCW0 [3:0]			
0x30	ADC_CHSELR2					CCW7								CCW6								CCW5 [3:0]								CCW4 [3:0]			
0x4C	ADC_FIFO																	DATA [15:0]															

NOTE:

1. All the registers are CPU supervisor or user mode accessible.

2. The darked bits are reserved, and must be kept at reset value.

27.10.2. Registers

27.10.2.1. ADC Interrupt And Status Register (ADC_ISR)

Offset Address: 0x0000

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	AWD	EMPTY	FULL	OVR	EOSEQ	EOC	EOSMP	ADRDY
W	w1c			w1c	w1c	w1c	w1c	w1c
RESET:	0	1	0	0	0	0	0	0

	= Writes have no effect and the access terminates without a transfer error excepiton.
w1c	= .Write 1 to the bit will clear it

Figure 27-12: ADC Interrupt And Status Register (ADC_ISR)

Read: Anytime

Write: Anytime

AWD — Analog watchdog flag

This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_TR register. It is cleared by software writing 1 to it.

1 = Analog watchdog event occurred

0 = No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

EMPTY — FIFO empty status

This bit is set by hardware when the FIFO is empty. It is cleared by hardware when FIFO is not empty.

1 = FIFO is empty

0 = FIFO is not empty

FULL — FIFO full status

This bit is set by hardware when the FIFO is full. It is cleared by hardware when FIFO is not full.

1 = FIFO is full

0 = FIFO is not full

OVR — ADC overrun

This bit is set by hardware when an overrun occurs, meaning that a new conversion has complete while the FULL flag was already set. It is cleared by software writing 1 to it.

1 = Overrun has occurred

0 = No overrun occurred (or the flag event was already acknowledged and cleared by software)

EOSEQ — End of sequence flag

This bit is set by hardware at the end of the conversion of a sequence. It is cleared by software writing 1 to it.

1 = Conversion sequence complete

0 = Conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

EOC — End of conversion flag

This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the ADC_FIFO register. It is cleared by software writing 1 to it or by reading the ADC_FIFO register.

1 = Channel conversion complete

0 = Channel conversion not complete (or the flag event was already acknowledged and cleared by software)

EOSMP — End of sampling flag

This bit is set by hardware during the conversion, at the end of the sampling phase.

1 = End of sampling phase reached

0 = Not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

ADRDY — ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN=1) and when the ADC reaches a state where it is ready to accept conversion requests. It is cleared by software writing 1 to it.

1 = ADC is ready to start conversion

0 = ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

27.10.2.2. ADC Interrupt Enable Register (ADC_IER)

Offset Address: 0x0004

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	AWDIE	0	0	OVRIE	EOSEQIE	EOCIE	EOSMPIE	ADRDYIE
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

Figure 27-13: ADC Interrupt Enable Register (ADC_IER)

Read: Anytime

Write: Before ADC start

AWDIE — Analog watchdog interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

1 = Analog watchdog interrupt enabled

0 = Analog watchdog interrupt disabled

OVRIE — Overrun interrupt enable

This bit is set and cleared by software to enable/disable the overrun interrupt.

1 = Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

0 = Overrun interrupt disabled

EOSEQIE — End of conversion sequence interrupt enable

This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt.

1 = EOSEQ interrupt enabled. An interrupt is generated when the EOSEQ bit is set.

0 = EOSEQ interrupt disabled

EOCIE — End of conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of conversion interrupt.

1 = EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

0 = EOC interrupt disabled

EOSMPIE — End of sampling flag interrupt enable

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt.

1 = EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

0 = EOSMP interrupt disabled.

ADRDYIE — ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

1 = ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

0 = ADRDY interrupt disabled.

27.10.2.3. ADC Control Register (ADC_CR)

Offset Address: 0x0008

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADSTP	ADSTART	ADDIS	ADEN
W					rs	rs	rs	rs
RESET:	0	0	0	0	0	0	0	0

	= Writes have no effect and the access terminates without a transfer error excepiton.
rs	= Software can read and set, write 0 has no effect

Figure 27-14: ADC Control Register (ADC_CR)

Read: Anytime

Write: See each bit description

ADSTP — ADC stop conversion command

This bit is set by software to stop and discard an ongoing conversion (ADSTP Command). It is cleared by hardware when the conversion is effectively discarded and the ADC is ready to accept a new start conversion command.

1 = Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.

0 = No ADC stop conversion command ongoing

NOTE: *Software is allowed to set ADSTP only when ADSTART=1 and ADDIS=0 (ADC is enabled and may be converting and there is no pending request to disable the ADC)*

ADSTART — ADC start conversion command

This bit is set by software to start ADC conversion. Depending on the TRIGMODE configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– In single conversion mode when software trigger is selected : at the assertion of the

End of Conversion Sequence (EOSEQ) flag.

– In discontinued conversion mode when software trigger is selected: at the assertion of the End of Conversion (EOC) flag.

– In all cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.

1 = Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting.

0 = No ADC conversion is ongoing.

NOTE: *Software is allowed to set ADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC)*

ADDIS — ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state). It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

1 = Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

0 = No ADDIS command ongoing

NOTE: *Software is allowed to set ADDIS only when ADEN=1 and ADSTART=0 (which ensures that no conversion is ongoing)*

ADEN — ADC enable command

This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the ADRDY flag has been set. It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

1 = Write 1 to enable the ADC.

0 = ADC is disabled (OFF state).

NOTE: *Software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADSTP=0, ADSTART=0, ADDIS=0 and ADEN=0)*

27.10.2.4. ADC Configuration Register 1 (ADC_CFGR1)

Offset Address: 0x000C

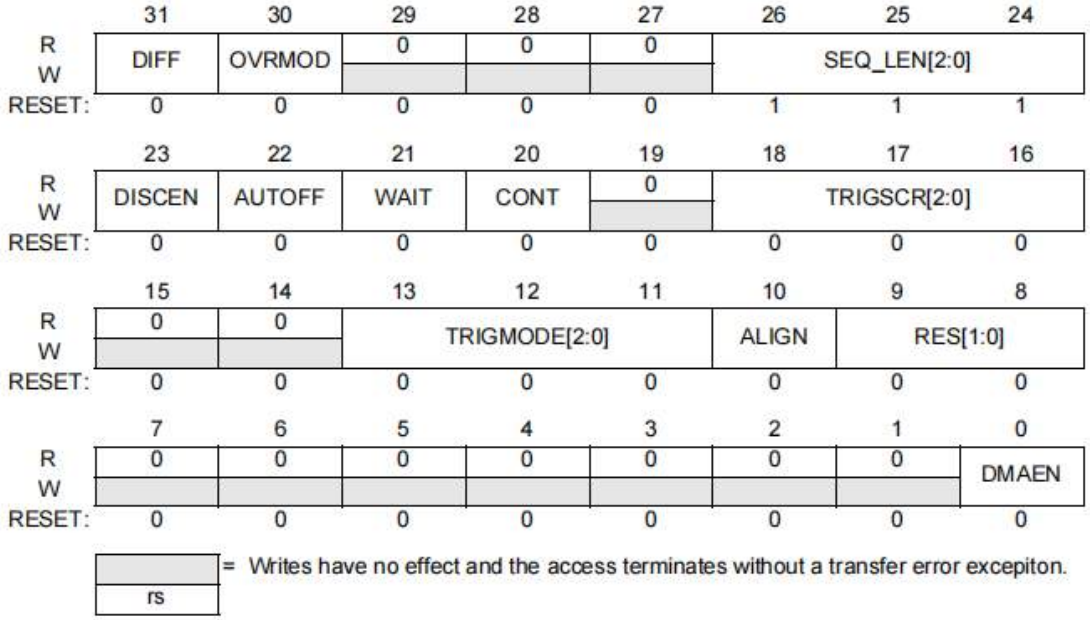


Figure 27-15: ADC Configuration Register 1 (ADC_CFGR1)

Read: Anytime

Write: Before ADC start

DIFF — Select differential-input

This bit determines that if the input is single-ended or differential-input.

1 = The analog input is differentially sampled.

0 = The analog input is single sampled

OVRMOD — Overrun management mode

This bit is set and cleared by software and configure the way data overruns are managed.

1 = ADC_DR register is overwritten with the last conversion result when an overrun is detected.

0 = ADC_DR register is preserved with the old data when an overrun is detected.

SEQ_LEN[2:0] — Sequence length

These bits define the length of sequence. The sequence length=SEQ_LEN+1. For example, SEQ_LEN=7 means sequence length is 8; SEQ_LEN=0 means sequence length is 1.

DISCEN — Discontinuous mode

This bit is set and cleared by software to enable/disable discontinuous mode.

1 = Discontinuous mode enabled

0 = Discontinuous mode disabled

AUTOFF — Auto-off mode

This bit is set and cleared by software to enable/disable auto-off mode.

1 = Auto-off mode enabled
0 = Auto-off mode disabled

WAIT — Wait conversion mode

This bit is set and cleared by software to enable/disable wait conversion mode.

1 = Wait conversion mode on

0 = Wait conversion mode off

CONT — Single / continuous conversion mode

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.

1 = Continuous conversion mode

0 = Single conversion mode

TRIGSCR[2:0] — External trigger source

These bits are used to select which of 8 possible events can trigger conversions. See **Table 27-3**.

TRIGMOD[2:0] — Trigger mode select

These bits are used to select software trigger mode or external trigger polarity, see **Table 27-2**.

ALIGN — Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to **Figure 27-10**.

0 = Right alignment

1 = Left alignment

RES[1:0] — Data resolution

These bits are written by software to select the resolution of the conversion. Refer to **Figure 27-10**.

DMAEN — Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA controller to manage automatically the converted data.

1 = DMA enabled

0 = DMA disabled

27.10.2.5. ADC Configuration Register 2 (ADC_CFGR2)

Offset Address: 0x0010

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	QPR[3:0]			
W								
RESET:	0	0	0	0	0	0	1	0

	7	6	5	4	3	2	1	0
R	STCNT[7:0]							
W								
RESET:	0	0	1	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

rs

Figure 27-16: ADC Configuration Register 2 (ADC_CFGR2)

Read: Anytime

Write: Before ADC enable

QPR[3:0] — Prescaler Clock Divider Bits

These bits select the system clock divisor to generate the QADC clock as follows:

$$F_{QCLK} = F_{sys\_QCLK} / (QPR[3:0] + 1)$$

where:

0 < QPR[3:0] <= 15 and the value 1 is not allowed.

STCNT[7:0] — ADC startup counter bits

The ADC needs a stabilization time of tSTAB(~2us) before it starts converting accurately. This time is calculated by counting QCLK cycles until the internal counter reaches the STCNT[7:0]. So, user should set these bits before ADC enable. For example, if the QCLK = 16MHz, then should set the STCNT[7:0] = 2000/(1000/16)=32.

27.10.2.6. ADC Sampling Time Register (ADC_SMPR)

Offset Address: 0x0014

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	SMP[7:0]							
W								
RESET:	0	0	0	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error excepiton.

Figure 27-17: ADC Sampling Time Register (ADC_SMPR)

Read: Anytime

Write: Before ADC start

SMP[7:0] — Sampling time selection

These bits are written by software to select the sampling time that applies to all channels.

The sample time is calculated as (SMP[7:0]+2) QCLKs

Example : SMP[7:0] = 0x02 means the sample time is 4 QCLKs

27.10.2.7. ADC Watch Dog Register (ADC_WDG)

Offset Address: 0x0018

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	AWDEN	AWDSGL	0	0	AWDCH[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

Figure 27-18: ADC Watch Dog Register (ADC_WDG)

Read: Anytime

Write: Before ADC start

AWDEN — Analog watchdog enable

This bit is set and cleared by software.

1 = Analog watchdog enabled

0 = Analog watchdog disabled

AWDSGL — Enable the watchdog on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWDCH[4:0] bits or on all the channels

1 = Analog watchdog enabled on a single channel

0 = Analog watchdog enabled on all channels

AWDCH[3:0] — Analog watchdog channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

- 0000: ADC analog input Channel 0 monitored by AWD
- 0001: ADC analog input Channel 1 monitored by AWD
- .....
- 0111: ADC analog input Channel 7 monitored by AWD
- 1111: Temperature sensor monitored by AWD
- other values: Reserved, must not be used

27.10.2.8. ADC Watchdog Threshold Register (ADC_TR)

Offset Address: 0x001C

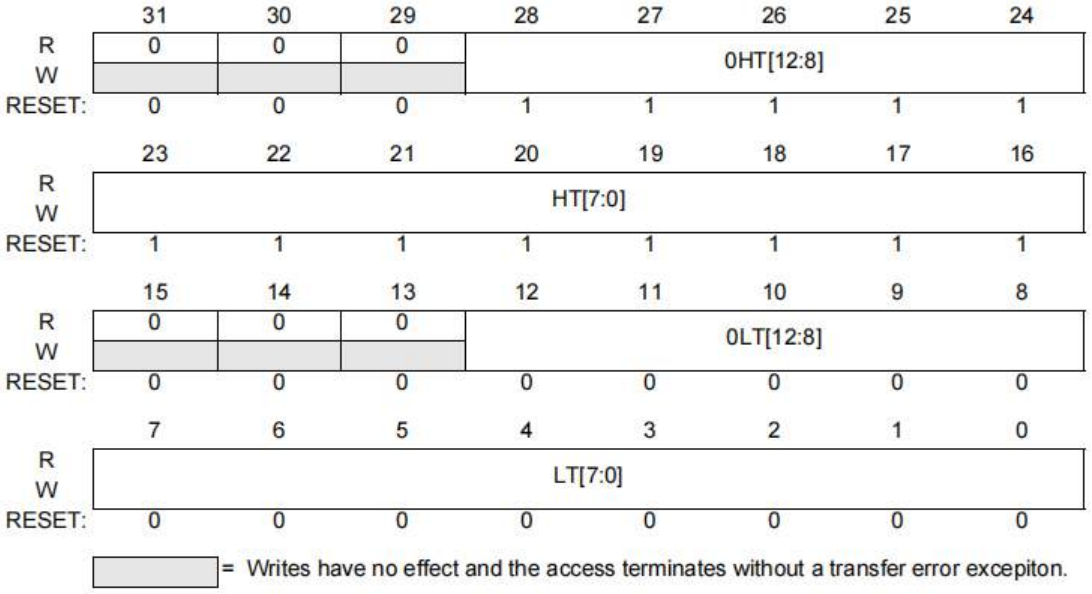


Figure 27-19: ADC Watchdog Threshold Register (ADC_TR)

Read: Anytime

Write: Before ADC start

HT[12:0] — Analog watchdog higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

LT[12:0] — Analog watchdog lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

27.10.2.9. ADC Channel Selection Register 1(ADC_CHSELR1, ADC_CHSELR2)

Offset Address: 0x002C

	31	30	29	28	27	26	25	24
R	0	0	0	0	CCW3[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	CCW2[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	CCW1[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	CCW0[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

Figure 27-20: ADC Channel Selection Register 1(ADC_CHSELR1)

Offset Address: 0x0030

	31	30	29	28	27	26	25	24
R	0	0	0	0	CCW7[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	CCW6[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	CCW5[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	CCW4[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

Figure 27-21: ADC Channel Selection Register 2(ADC_CHSELR2)

Read: Anytime

Write: Before ADC start

CCWi[3:0] — The number i conversion select channel, refer to **Table 27-1**.

27.10.2.10. ADC FIFO Access Register (ADC_FIFO)

Offset Address: 0x004C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	DATA[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	DATA[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

Figure 27-22: ADC FIFO Access Register (ADC_FIFO)

Read: Anytime

Write: Never

DATA[15:0]: Converted data, Refer to **Figure 27-10**.

28. Electrical Characteristic

This chapter provides the electrical characteristic parameters and limits for LT165.

28.1. Absolute Maximum Ratings

Table 28-1: Absolute Maximum Rating

Symbol	Item	Range	Unit
V_{DD33}	Power Supply	-0.5 ~ 4.6	V
V_{IN}	Input Voltage Range	-0.5 ～ VDD33+0.5	V
V_{OUT}	Output Voltage Range	-0.5 ～ VDD33+0.5	V
P_D	Power Dissipation	≦300	mW
T_{OPR}	Operation Temperature	-40 ~ 105	℃
T_{JT}	Operation Junction Temperature	-40 ~ 125	℃
T_{ST}	Storage Temperature	-55 ~ 150	℃
T_{SOL}	Soldering Temperature	260	℃

NOTE: If the loading to the chip exceeds the absolute maximum rating listed in **Table 5-1**, it may result in permanent damage to the chip. Although the chip contains circuitry to resist damage from high quiescent voltages, do not apply more voltages on the chip than the values rated in the table. These values are only ratings and do not mean that the chip functions properly under these conditions.

28.2. DC Electrical Specification

Table 28-2: IO Static Characteristic (3.3V)

Item	Symbol	Min	Typical	Max	Unit
IO Supply Power	VDD33	2.97	3.3	3.63	V
Input High Voltage	V_{IH}	2.0	-	VDD33+0.3	V
Input Low Voltage	V_{IL}	-0.3	-	0.8	V
Output High Voltage	V_{OH}	2.4	-	VDD33	V
Output Low Voltage	V_{OL}	0	-	0.4	V
Input Leakage Current	I_{IN}	-	-	1	uA
Pull-Up Resistor	RPU	33	41	62	KΩ
Pull-Down Resistor	RPD	33	42	68	KΩ

Table 28-3: Power Characteristic

Item	Symbol	Min	Typical	Max	Unit
Chip Power	VDD33	2.97	3.3	3.63	V
ADC Power I/P	AVDD	2.97	3.3	3.63	V
Chip Core Power (LDO O/P)	VDD12	1.1	1.2	1.3	V
RTC Power	VBAT	2.7	3.3	3.6	V

28.3. Electrostatic Discharge (ESD) Protection

Table 28-4: Electrostatic Discharge Protection Characteristic

ESD Test	Symbol	Max	Unit	Reference Standard
Human Body Model	HBM	4000	V	ANSI/ESDA/JEDEC JS-001-2017
Machine Model	MM	200	V	JEDEC JESD22-A115C-2010
Charged Device Model	CDM	800	V	ANSI/ESDA/JEDEC JS-002-2022
Latch Up	LU	200	mA	JEDEC JESD78F.01-2022, @105℃

Note: When performing manual soldering, it is recommended that personnel and equipment should be treated with anti-static. For example, appropriate temperature and humidity environment, grounding of welding equipment, anti-static workbench, and welding personnel wearing anti-static wrist straps, etc.

28.4. VDD Power Up Timin

When using the LT165x, it is important to pay attention to the power up requirements of VDD. The VDD33 must maintain a waiting time of at least 400ms at the low voltage (V_L) at the time of power-up (T_{WAIT}). At the same time, the rise time (T_R) of VDD33 from V_L to normal operating voltage should not be too long. The normal operating voltage range must be reached within 500ms. Otherwise, the MCU inside the LT165 will not boot properly.

Table 28-5: VDD Power Up Characteristic

Item	Symbol	Description	Min.	Nom.	Max.	Unit
Rise Time	T_R	The rise time of input voltage from V_L to the normal operating voltage	-	-	500	ms
Wait Time	T_{WAIT}	The retention time of the V_L before Power On	400	-	-	ms
VDD Input Voltage	V_L	at $T=T_1$ on pin VDD33 (The input voltage before Power Up)	-	-	200	mV
VDD Input Voltage	V_H	Normal Operation Voltage	2.97	3.3	3.63	V

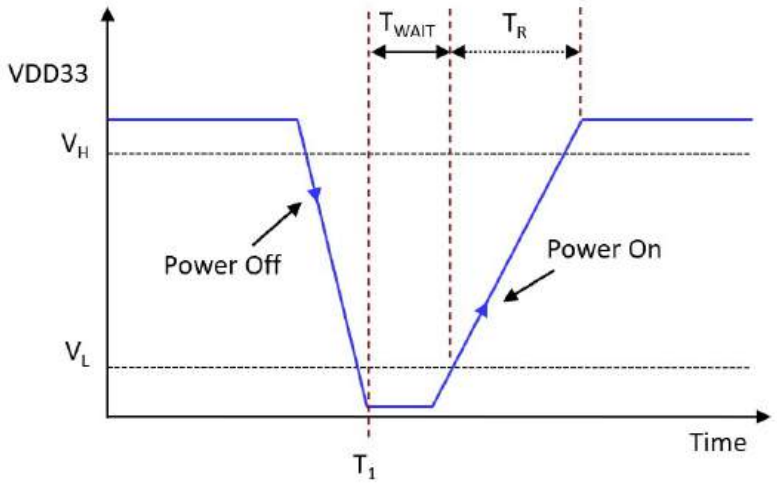


Figure 28-1: VDD Power up Timing Requirement

29. Application Circuit

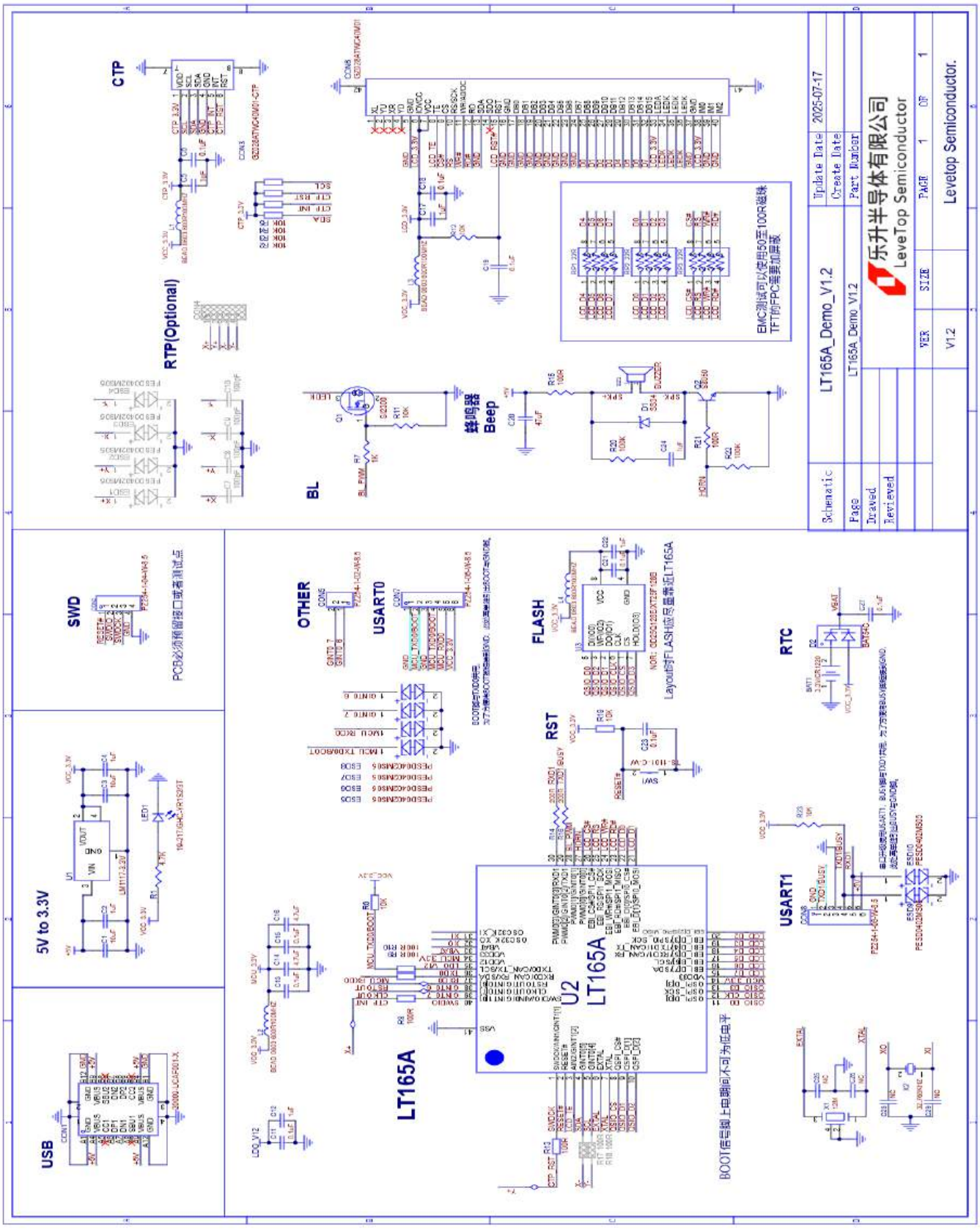


Figure 29-1: AP Circuit of LT165A for 8bit 8080 Interface TFT LCD Panel

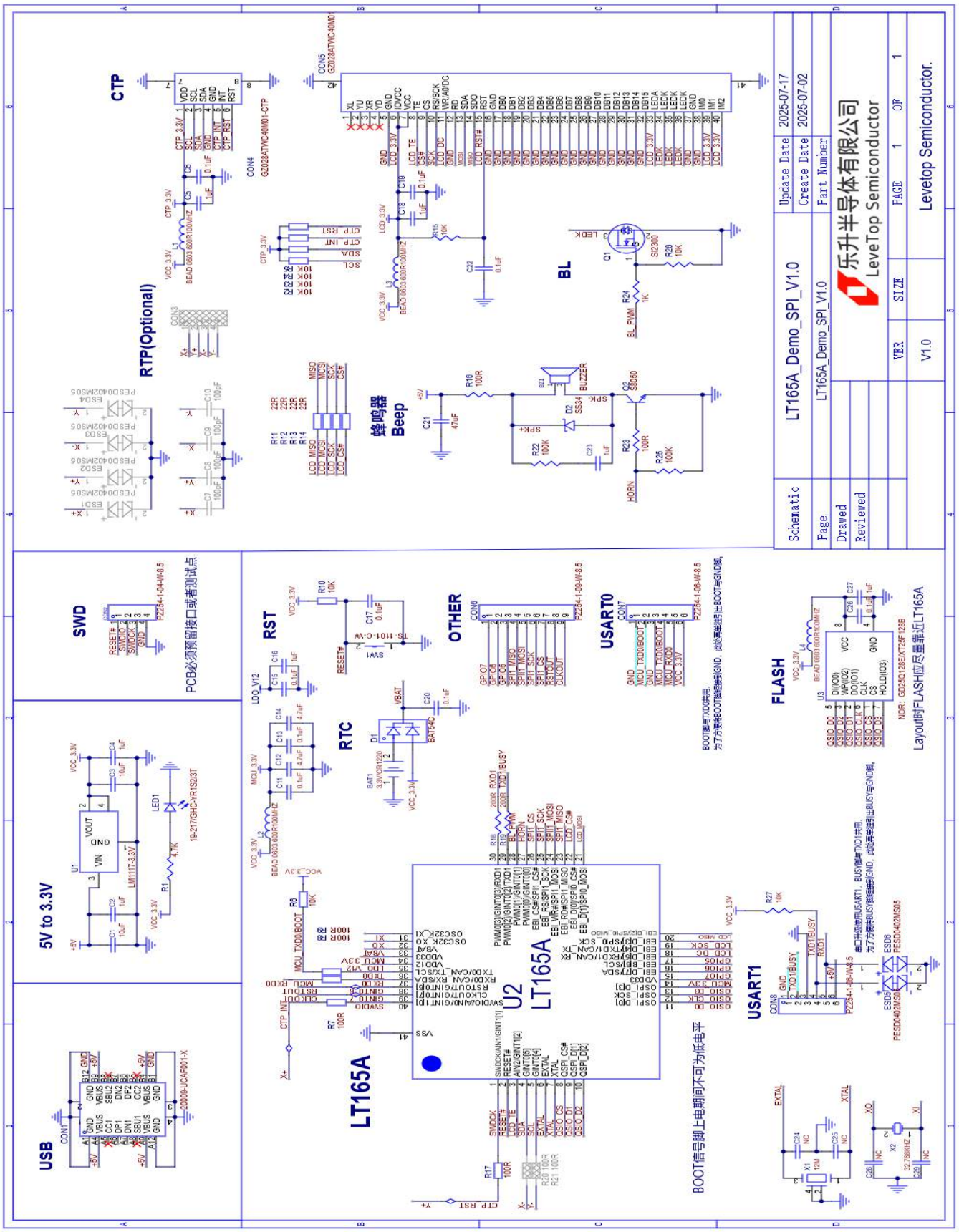


Figure 29-2: AP Circuit of LT165A for SPI Interface TFT LCD Panel

30. Package

30.1. LT165A (QFN-40pin)

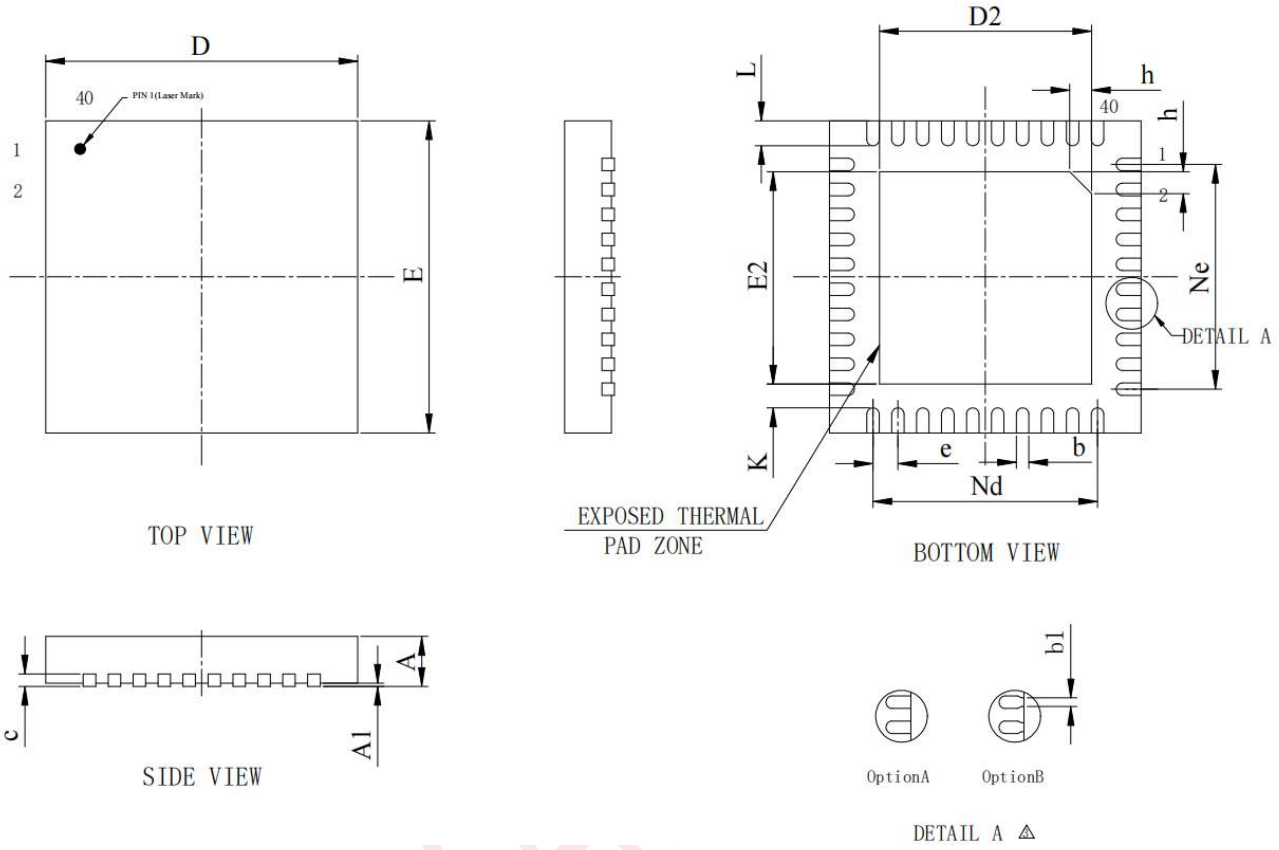


Figure 30-1: LT165A Package Overview

Note: When layout PCB, LT165A Thermal Pad Zone must be connected to ground. Please refer to Section 7.3 for PCB routing of ground thermal pad.

Table 30-1: LT165A Package Parameter

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
A	0.70	0.75	0.80	Nd	3.6BSC		
A1	--	0.02	0.05	E	4.90	5.00	5.10
b	0.15	0.20	0.25	E2	3.30	3.40	3.50
b1	0.14REF			Ne	3.60BSC		
c	0.18	0.25	0.30	L	0.35	0.40	0.45
D	4.90	5.00	5.10	K	0.20	--	--
D2	3.30	3.40	3.50	h	0.30	0.35	0.40
e	0.40BSC						

30.2. LT165B (TSSOP-30pin)

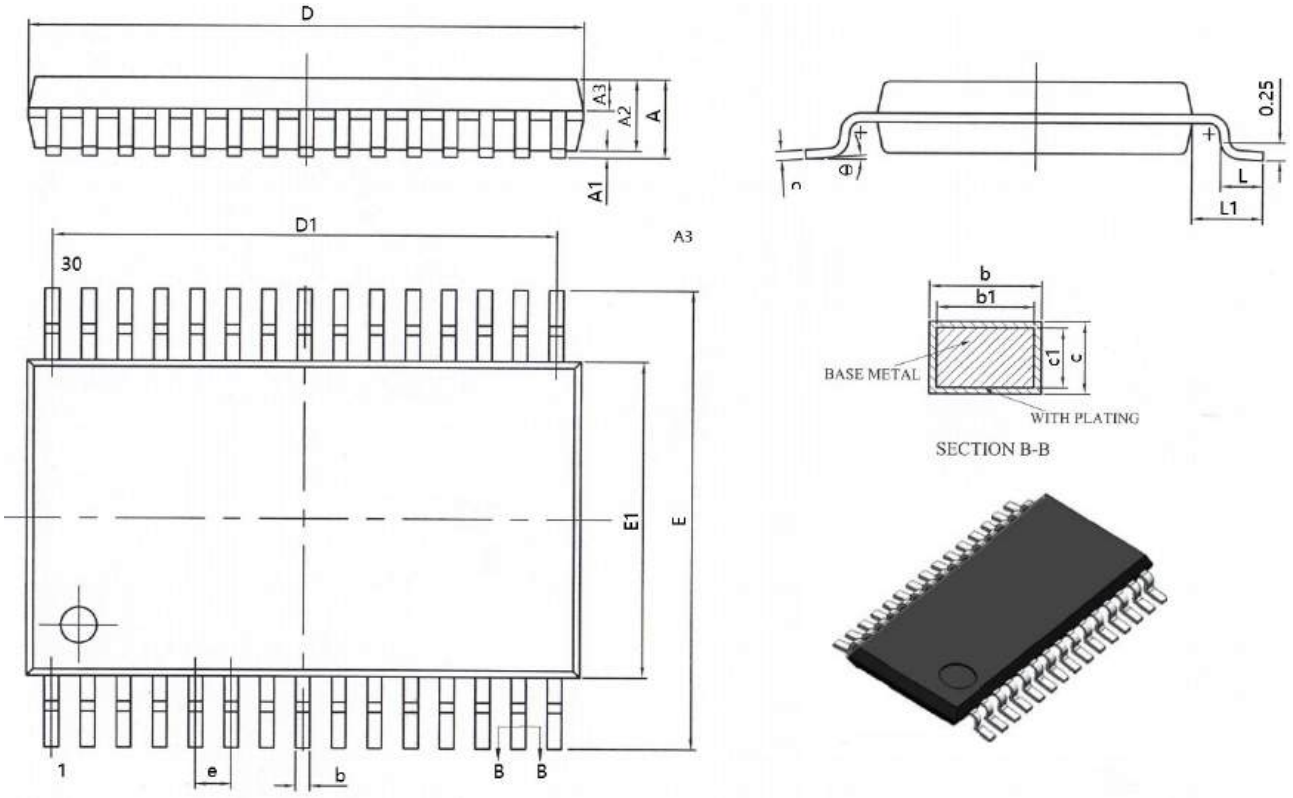


Figure 30-2: LT165B Package Overview

Table 30-2: LT165B Package Parameter

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
A	--	--	1.20	D	7.70	7.80	7.90
A1	0.05	--	0.15	D1	6.90	7.00	7.10
A2	0.80	1.00	1.05	E	6.20	6.40	6.60
A3	0.39	0.44	0.49	E1	4.30	4.40	4.50
b	0.18	--	0.27	e	0.50BSC		
b1	0.17	0.20	0.23	L	0.45	0.60	0.75
c	0.13	--	0.17	L1	1.00BSC		
c1	0.12	0.13	0.14	θ	0°	--	8°

30.3. PCB Design for Ground Pad

The LT165A is available in a QFN package with a ground (GND) thermal pad on the back of the chip. In order to achieve better heat dissipation and reduce the risk of soldering, it is recommended to divide the copper surface of the PCB on the bottom pad of LT165A into four small solder surfaces (square or round) when laying out the PCB. And the spacing between each soldering surface is set at ~0.8mm, so as to avoid incomplete soldering caused by the PCB using a complete soldering surface that is the same or even larger than the size of the LT165A pad, or the chip deformation and poor contact caused by the pulling of the PCB and the chip pad after soldering cooling.

The correct PCB pad layout is shown in the following examples, the light yellow area in the middle is the ground pad at the bottom of the LT165A, and the gray area is the small PCB ground pad (solder surface). Each pad has 1~2 vias grounded.

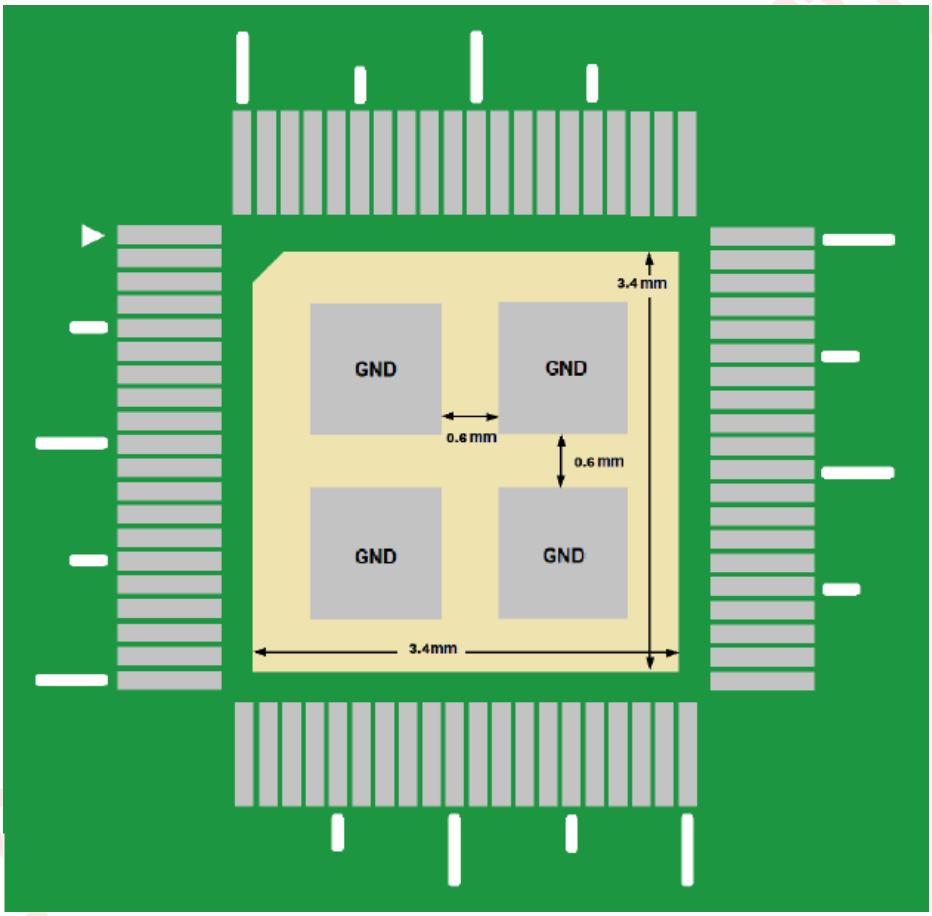


Figure 30-3: PCB Design Recommendation for the LT165A Bottom Ground Pad