

**LT168**

# Uart TFT Display Controller

---

## Data Sheet

V1.5

## Version History

Version	Date	Description
V1.0	2023/10/16	<ul style="list-style-type: none"> <li>● LT168 Preliminary Release</li> </ul>
V1.1A	2024/3/7	<ul style="list-style-type: none"> <li>● Fixed the syntax errors described in the previous version.</li> <li>● Update Section 2.3 Signal Properties Summary</li> <li>● Update Chapter 33 Application Circuit</li> </ul>
V1.2	2024/3/8	<ul style="list-style-type: none"> <li>● Update Table 2-3 LT168x Comparison</li> <li>● Update Chapter 33 Application Circuit</li> </ul>
V1.5	2024/6/11	<ul style="list-style-type: none"> <li>● Add Table 3-1 : The differences of LT168B for RGB Panel with or without PSRAM</li> <li>● Add Section 32.4 VDD Power Up Timing</li> <li>● Update Chapter 33 Application Circuit</li> </ul>

## Contents

Version History .....	2
Contents .....	3
Figure List .....	18
Table List .....	26
1. LT168 Introduction .....	29
1.1. Introduction .....	29
1.2. Internal Block Diagram .....	30
1.3. Features .....	31
1.3.1. 32-bits RISC Processor .....	31
1.3.2. On-chip 768K Bytes Of Static Random-Access Memory (SRAM) .....	31
1.3.3. On-chip QSPI Flash .....	31
1.3.4. 8K Bytes Of Static Read-Only Memory (ROM) .....	31
1.3.5. 32K Bytes Of Cache Memory .....	31
1.3.6. External Bus Interface .....	31
1.3.7. RGB Display Panel Interface .....	32
1.3.8. DMA Controller .....	32
1.3.9. Reset Function .....	32
1.3.10. Four Periodic Interval Timer .....	32
1.3.11. One Watchdog Timer .....	32
1.3.12. One RTC Timer .....	33
1.3.13. Three External Interrupts Port (EPORT) .....	33
1.3.14. Three Quad Serial Peripheral Interface Master Module (QSPI) .....	33
1.3.15. Three Serial Communications Interface Module (SCI) .....	33
1.3.16. One Canbus Controller: .....	34
1.3.17. One Usb2.0 Full Speed Compatible Device (USB) .....	34
1.3.18. Two Pulse Width Modulator (PWM) .....	35
1.3.19. ADC With 8-Channel .....	35
1.3.20. Two Analog Comparators .....	35
1.3.21. Power Management Unit (PMU) .....	36
1.3.22. Programmable Voltage Detector .....	36
1.3.23. Internal Oscillator .....	36
1.3.24. External Crystal Oscillator .....	36
1.4. System Block .....	37
1.5. Application Diagram .....	37
2. Signal Description .....	38
2.1. Introduction .....	38
2.2. Pin Assignment .....	38
2.3. Signal Properties Summary .....	40
2.4. Signal Descriptions .....	44
2.5. LT168A vs. LT168B .....	49

3. Hardware Interface .....	50
3.1. Host Communication Interface .....	50
3.2. TFT LCD Panel Interface .....	50
3.3. QSPI Flash Interface .....	52
3.4. Touch Panel Interface .....	53
3.5. Clock Interface .....	53
3.6. Backlight Control Circuit .....	54
3.7. Can Bus Interface .....	55
3.8. Audio Interface .....	55
3.9. Real Time Clock Circuit .....	56
3.10. Reset Interface .....	56
3.11. USB Interface .....	57
3.12. SPI/Uart Interface .....	57
4. System Memory Map .....	58
4.1. Introduction .....	58
4.2. Address Map .....	58
5. Embedded Interrupt Controller(EIC) .....	60
5.1. Introduction .....	60
5.2. Features .....	60
5.3. Memory Map and Registers .....	60
5.3.1. Memory Map .....	60
5.3.2. Register Descriptions .....	61
5.3.2.1. Interrupt Control Status Register (ICSR) .....	61
5.3.2.2. Interrupt Enable Register (IER) .....	63
5.3.2.3. Interrupt Pending Set Register (IPSR) .....	64
5.3.2.4. Interrupt Pending Clear Register (IPCR) .....	65
5.3.2.5. Priority Level Select Registers (PLSR) .....	66
5.3.2.6. System Priority Level Select Registers (SYSPLSR) .....	67
5.4. Function Description .....	68
5.4.1. Interrupt Handling Without Conflicition .....	68
5.4.2. Interrupt With Conflicition .....	69
5.4.3. Pend Trap Function .....	69
5.5. Interrupts .....	70
6. 32-bits RISC Introduction .....	75
6.1. Features .....	75
6.2. Microarchitecture Summary .....	75
6.3. Programming Model .....	76
6.4. Data Format Summary .....	77
6.5. Operand Addressing Capabilities .....	78
6.6. Instruction Set Overview .....	78

7. Embedded Programmable Timer (EPT) .....	81
7.1. Introduction .....	81
7.2. Memory Map and Registers .....	81
7.2.1. Memory Map.....	81
7.2.2. Register Descriptions .....	82
7.2.2.1. EPT Control Status Register (EPTCSR).....	82
7.2.2.2. EPT Reload Register (EPTRLD).....	83
7.2.2.3. EPT Count Register (EPTCNT) .....	84
7.3. Function Description .....	85
7.3.1. Count Timing .....	85
8. Chip Configuration Module (CCM) .....	86
8.1. Introduction .....	86
8.2. Features.....	86
8.3. Memory Map and Registers .....	86
8.3.1. Memory Map.....	86
8.3.2. Register Descriptions .....	87
8.3.2.1. Wakeup Configuration Register (WKUPC) .....	87
8.3.2.2. Chip Pin Pull Down Configuration Register (CPPDC) .....	89
8.3.2.3. QSPI XIP Mode Configuration Register (QSPIXIPMCFR) .....	90
8.3.2.4. QSPI 32-Bit Key Register (QSPILKEYR).....	91
8.3.2.5. QSPI GPIO Configuration Register (QSPIGPIOCR) .....	91
8.3.2.6. MCU Access RAM Priority Configuration Register (MCURAMPRIOCR) .....	92
8.3.2.7. EPORT2 Function Configuration Register (EPORT2FCR).....	94
9. Clock and Power Control Module (CLKPWRM) .....	95
9.1. Overview .....	95
9.2. Features.....	95
9.3. Clock Structure .....	95
9.4. Clock Source Select.....	96
9.4.1. Low-Power Options .....	96
9.4.1.1. Wait And Doze Modes.....	96
9.4.1.2. Stop Mode.....	96
9.5. Memory Map and Registers .....	97
9.5.1. Memory Map.....	97
9.5.2. Register Description .....	98
9.5.2.1. Synthesizer Control Register (SYNCR) .....	98
9.5.2.2. Low Speed Oscillator Control and Status Register (LOSCCSR).....	102
9.5.2.3. PLL Configuration And Status Register (PLLCSR) .....	103
9.5.2.4. Module Stop Control Register (MSCR).....	105
9.5.2.5. EPT External Clock Source Enable Control Register (ECSECR).....	107
9.5.2.6. OSC Bist Test Configuration Register1 (OBTCR1).....	108
9.5.2.7. OSC Bist Test Configuration Register2 (OBTCR2).....	108
9.5.2.8. OSC Bist Test Control Register (OBTCTLR) .....	109
9.5.2.9. OSC BIST Test Counter Register (OBTCNTR).....	110

9.5.2.10. OSC BIST Test Result Register (OBTRR) .....	111
9.6. Functional Description.....	112
9.6.1. Turning on PLL .....	112
9.6.2. The Frequency of PLL Measurement.....	112
9.6.3. The Frequency of 128KHz Measurement .....	112
10. Reset Control Module (RCM) .....	113
10.1. Overview .....	113
10.2. Features.....	113
10.3. Block Diagram.....	113
10.4. Memory Map and Registers .....	114
10.4.1. Memory Map.....	114
10.4.2. Register Description .....	114
10.4.2.1. Reset Test Register (RTR) .....	114
10.4.2.2. Reset Status Register (RSR) .....	115
10.4.2.3. Reset Control Register (RCR).....	116
10.5. Functional Description.....	116
10.5.1. Reset Sources .....	116
10.5.1.1. Power-On Reset (POR) .....	116
10.5.1.2. Watchdog Timer Reset.....	116
10.5.1.3. Software Reset.....	117
10.5.1.4. Programmable Voltage Detect Reset.....	117
10.5.2. Reset Control Flow.....	117
11. Static Random Access Memory (SRAM).....	118
11.1. Introduction .....	118
11.2. Modes Of Operation.....	118
11.3. Low-Power Modes .....	118
11.4. Reset Operation.....	118
11.5. Interrupts.....	118
12. Cache Module (CACHM) .....	119
12.1. Introduction .....	119
12.2. Block Diagram.....	120
12.3. Memory Map and Registers .....	121
12.3.1. Memory Map.....	121
12.3.2. Register Description .....	121
12.3.2.1. Cache Control Register (LMEM_CCR).....	121
12.3.2.2. Cache Line Control Register (LMEM_CLCR) .....	122
12.3.2.3. Cache Search Address Register (LMEM_CSAR).....	124
12.3.2.4. Cache Read/Write Value Register (LMEM_CCVR) .....	125
12.3.2.5. Cache Access Register (LMEM_ACRG).....	126
12.3.2.6. Cache Page Invalidation Base Address Register (LMEM_PAGE_INV_BADDR). 127	
12.3.2.7. Cache Page Invalidation Base Size Register (LMEM_PAGE_INV_SIZE).....	128
12.3.2.8. Cache Clock Enable Register (LMEM_CACHE_CLK_EN) .....	129

12.4. Cache Function .....	130
12.5. Cache Control .....	131
12.5.1. Cache Set Commands .....	131
12.5.2. Cache Line Commands .....	132
12.5.2.1. Executing a Series Of Line Commands Using Cache Addresses .....	133
12.5.2.2. Executing a Series of Line Commands Using Physical Addresses .....	133
12.5.2.3. Line Command Results .....	133
13. Crossbar Switch (XBAR) .....	135
13.1. Introduction .....	135
13.1.1. Overview .....	135
13.1.2. Features .....	135
13.2. Memory Map and Registers .....	136
13.2.1. Memory Map .....	136
13.2.2. Register Descriptions .....	137
13.2.2.1. Master Priority Register (XBAR_MPRn) .....	137
13.2.2.2. Slave General Purpose Control Register (XBAR_SGPCRn) .....	138
13.3. Function .....	140
13.3.1. General Operation .....	140
13.3.2. Register Coherency .....	141
13.3.3. Arbitration .....	141
13.3.3.1. Fixed Priority Operation .....	141
13.3.3.2. Round-robin Priority Operation .....	141
13.3.4. Priority Assignment .....	142
13.4. Initialization/Application Information .....	142
14. Direct Memory Access Controller (DMAC) .....	143
14.1. Information Specific of DMA Controller .....	143
14.1.1. DMAC Features .....	143
14.1.2. Channel Assignments .....	144
14.2. Introduction .....	145
14.2.1. Features .....	146
15. Efuse Control Module (EFM) .....	147
15.1. Introduction .....	147
15.2. Features .....	147
15.3. Memory Map and Registers .....	147
15.3.1. Memory Map .....	147
15.3.2. Register Descriptions .....	148
15.3.2.1. Efuse Configuration Register(EFMCR) .....	148
15.3.2.2. Efuse Read Data Register(EFMDR) .....	149
16. Option Byte (OPB) .....	150
16.1. Memory Map and Registers .....	150
16.1.1. Memory Map .....	150
16.1.2. Register Descriptions .....	150

16.1.2.1. Programmable Voltage Detector Configuration Register (PVDC) .....	150
16.1.2.2. Customer Configuration Register (CCR) .....	152
16.1.2.3. External High Speed Oscillator Stable Time Configuration Register (EOSCST)..	155
16.1.2.4. PLL Lock Time Configuration Register (PLLOCKCR).....	155
16.1.2.5. RESET Pin Filter Enable and Value Register (RFEVR).....	156
16.1.2.6. Programmable Voltage Detector Filter Enable and Value Register (PVDFEVR) .	157
16.1.2.7. Factory Configuration Register (FCR) .....	158
16.1.2.8. Channel Disable Configuration Register (ADCCDISR) .....	159
16.1.2.9. VREF1P2 Trimming Configuration Register (VREFTCR).....	160
16.1.2.10. LDO1P2 Trimming Configuration Register (LDOTCR) .....	161
16.1.2.11. RTC Trimming Configuration Register (RTCTCR).....	162
<b>17. RGB Controller Module (RGBC).....</b>	<b>163</b>
17.1. Introduction .....	163
17.2. Block Diagram.....	163
17.3. RGB Interface Definition .....	164
17.4. RGB Interface Timing.....	165
17.5. Memory Map and Registers .....	166
17.5.1. Memory Map.....	166
17.5.2. Register Descriptions .....	167
17.5.2.1. Horizontal Pulse Width And Back Porch Register (THPB) .....	167
17.5.2.2. Horizontal Period And Front Porch Register (THDF).....	168
17.5.2.3. Vertical Pulse Width And Back Porch Register (TVPB).....	169
17.5.2.4. Vertical Period And Front Porch Register (TVDF).....	170
17.5.2.5. Memory Start Address Register (MSADDR).....	171
17.5.2.6. Memory End Address Register (MEADDR) .....	172
17.5.2.7. Number of Frames Transferred (NFTR).....	173
17.5.2.8. Number of Frames Transferred Threshold Configuration Register (NFTTCR).....	174
17.5.2.9. RGB Control And Status Register (RCSR) .....	175
17.6. Function Description .....	177
<b>18. Blender Controller (BLDC).....</b>	<b>178</b>
18.1. Introduction .....	178
18.2. Blending Algorithm .....	179
18.3. Memory Map and Registers .....	179
18.3.1. Memory Map.....	179
18.3.2. Register Descriptions .....	180
18.3.2.1. Blender Control and Status Register (BCSR).....	180
18.3.2.2. Source Address Register (SADDR) .....	181
18.3.2.3. Transfer Total Address Length Register (TTALR) .....	182
18.3.2.4. Destination Address Register (DADDR) .....	182
18.3.2.5. Transfer Destination Minor Length and Offset Register (TDMLOR).....	183
<b>19. External Bus Interface (EBI) .....</b>	<b>184</b>
19.1. Introduction .....	184
19.2. Features.....	184



19.3. Signal Description .....	184
19.4. Memory Map and Registers .....	184
19.4.1. Memory Map.....	184
19.4.2. Register Descriptions .....	185
19.4.2.1. EBI Control Registers (EBICR) .....	185
19.4.2.2. IO Pull Control Register (IOPCR) .....	187
19.4.2.3. IO Control Register (IOCR) .....	188
19.4.2.4. GPIO Data Output Register (GPIODO) .....	189
19.4.2.5. GPIO Direction Register (GPIODIR).....	190
19.4.2.6. GPIO Data Input Register (GPIODI) .....	191
19.5. Functional Description.....	192
19.5.1. EBI_WR#, EBI_RD# Signal Timing.....	192
19.5.2. EBI Functional Description .....	193
19.5.2.1. EBI_CS# Chip Select.....	193
19.5.2.2. Operand Transfer .....	193
20. Programmable Interrupt Timer (PIT).....	195
20.1. Introduction .....	195
20.2. Block Diagram.....	195
20.3. Modes of Operation.....	195
20.3.1. Wait Mode.....	195
20.3.2. Doze Mode .....	195
20.3.3. Stop Mode .....	195
20.3.4. Debug Mode .....	195
20.4. Signals .....	196
20.5. Memory Map and Registers .....	196
20.5.1. Memory Map.....	196
20.5.2. Register Descriptions .....	196
20.5.2.1. PIT Modulus Register (PMR) .....	197
20.5.2.2. PIT Control and Status Register (PCSR).....	197
20.5.2.3. PIT Count Register (PCNTR).....	199
20.6. Functional Descripton.....	200
20.6.1. Set-and-Forget Timer Operation .....	200
20.6.2. Free-Running Timer Operation.....	200
20.6.3. Timeout Specifications .....	200
20.7. Interrupt Operation .....	201
21. Watchdog Timer (WDT).....	202
21.1. Introduction .....	202
21.2. Modes of Operation.....	202
21.2.1. Wait Mode.....	202
21.2.2. Doze Mode .....	202
21.2.3. Stop Mode .....	202
21.2.4. Debug Mode .....	202
21.3. Block Diagram.....	203

21.4. Signals .....	203
21.5. Memory Map and Registers .....	203
21.5.1. Memory Map.....	203
21.5.2. Register Description .....	204
21.5.2.1. Watchdog Modules Register (WMR).....	204
21.5.2.2. Watchdog Control Register (WCR).....	205
21.5.2.3. Watchdog Service Register (WSR).....	207
21.5.2.4. Watchdog Count Register (WCNTR).....	207
22. Real Time Controller (RTC) .....	208
22.1. Introduction .....	208
22.2. Features .....	208
22.3. Test Mode .....	208
22.4. Block Diagram.....	208
22.5. Application Circuit .....	209
23. Edge Port Module (EPORT) .....	210
23.1. Introduction .....	210
23.2. Low-Power Mode Operation .....	210
23.2.1. Wait and Doze Modes .....	210
23.2.2. Stop Mode .....	210
23.3. Interrupt/General-Purpose I/O Pin Descriptions .....	211
23.4. Memory Map And Registers .....	211
23.4.1. Memory Map.....	211
23.4.2. Register Description .....	212
23.4.2.1. Edge Port Interrupt Enable Register (EPIER).....	212
23.4.2.2. Eport Data Direction Register (EPDDR) .....	212
23.4.2.3. Eport Pin Assignment Register (EPPAR).....	213
23.4.2.4. Eport Pin Pull-up Enable Register (EPPUE).....	214
23.4.2.5. Edge Port Flag Register (EPFR).....	214
23.4.2.6. Edge Port Pin Data Register (EPPDR).....	215
23.4.2.7. Edge Port Data Register (EPDR).....	215
23.4.2.8. Eport Port Bit Set Register (EPBSR) .....	215
23.4.2.9. Eport Digital Filter Control Register (EPFC) .....	216
23.4.2.10. Eport Open Drain Enable Register (EPODE) .....	216
23.4.2.11. Eport Level Polarity Register (EPLPR) .....	217
23.4.2.12. Eport Port Bit Clear Register (EPBCR).....	217
24. CANBus Controller(CANBC) .....	218
24.1. Introduction .....	218
24.1.1. Overview.....	218
24.1.2. CANBus Module Features.....	219
24.1.3. Modes Of Operation .....	219
24.2. External Signal Description .....	220
24.2.1. Overview.....	220
24.2.2. Signal Descriptions.....	220

24.2.2.1. CANRX.....	220
24.2.2.2. CANTX .....	220
<b>25. Serial Communication Interface (SCI) .....</b>	<b>221</b>
25.1. Introduction .....	221
25.2. Features.....	221
25.3. Modes of Operation.....	222
25.4. Block Diagram.....	222
25.5. Modes of Operation.....	223
25.5.1. Stop Mode .....	223
25.5.2. Wait Mode.....	223
25.6. Signal Description .....	223
25.7. Memory Map and Registers .....	224
25.7.1. Memory Map.....	224
25.7.2. Register Description .....	225
25.7.2.1. SCI Version ID Register (SCI_VERID).....	225
25.7.2.2. SCI Parameter Register (SCI_PARAM).....	225
25.7.2.3. SCI Reset Register (SCI_RESET).....	226
25.7.2.4. SCI Pin Register (SCI_PIN) .....	227
25.7.2.5. SCI Baud Rate Register (SCI_BAUD) .....	227
25.7.2.6. SCI Status Register (SCI_STAT) .....	229
25.7.2.7. SCI Control Register (SCI_CTRL) .....	233
25.7.2.8. SCI Data Register (SCI_DATA).....	237
25.7.2.9. SCI Match Address Register (SCI_MATCH).....	238
25.7.2.10. SCI Modem IrDA Register (SCI_MODIR) .....	239
25.7.2.11. SCI FIFO Register (SCI_FIFO).....	241
25.7.2.12. SCI Watermark Register (SCI_WATER) .....	243
25.7.2.13. SCI Oversampling Ratio Register (SCI_OSR).....	244
25.8. Functional Description.....	245
25.9. Baud Rate Generation .....	245
25.10. Transmitter Functional Description .....	246
25.10.1. Send Break And Queued Idle.....	246
25.11. Receiver Functional Description .....	248
25.11.1. Data Sampling Technique .....	248
25.11.2. Receiver Wakeup Operation .....	249
25.11.2.1. Idle-line Wakeup.....	250
25.11.2.2. Address-mark Wakeup.....	250
25.11.2.3. Data Match Wakeup.....	250
25.11.2.4. Address Match Operation .....	250
25.11.2.5. Idle Match Operation.....	251
25.11.2.6. Match On Match Off Operation .....	251
25.11.3. Infrared Decoder.....	251
25.11.3.1. Start Bit Detection .....	251
25.11.3.2. Noise Filtering .....	251
25.11.3.3. Lowbits Detection.....	251

25.11.3.4. Highbits Detection .....	252
25.12.    Additional SCI Functions.....	252
25.12.1. 8-bits, 9-bits and 10-bits Data Modes.....	252
25.12.2. Idle Length.....	252
25.12.3. Single-wire Operation.....	253
25.12.4. Loop Mode.....	253
25.13.    Infrared Interface .....	253
25.13.1. Infrared Transmit Encoder.....	253
25.13.2. Infrared Receive Decoder .....	253
25.14.    Interrupts And Status Flags.....	254
26. Synchronous Serial Interface (SSI) .....	255
26.1. Introduction .....	255
26.2. Features.....	255
26.3. Modes of Operation.....	255
26.4. Block Diagram.....	256
26.5. Appliaction Diagram.....	256
26.6. Memory Map and Registers .....	257
26.6.1. Memory Map.....	257
26.6.2. Register Descriptions .....	258
26.6.2.1. Control Register 0 (CTRLR0).....	258
26.6.2.2. Control Register 1 (CTRLR1).....	261
26.6.2.3. SSI Enable Register (SSIENR).....	262
26.6.2.4. Microwire Control Register (MWCR).....	263
26.6.2.5. Slave Enable Register (SER).....	264
26.6.2.6. Baud Rate Select (BAUDR) .....	265
26.6.2.7. Transmit FIFO Threshold Level (TXFTLR) .....	266
26.6.2.8. Receive FIFO Threshold Level (RXFTLR).....	267
26.6.2.9. Transmit FIFO Level Register (TXFLR) .....	268
26.6.2.10. Receive FIFO Level Register (RXFLR).....	269
26.6.2.11. Status Register (SR) .....	270
26.6.2.12. Interrupt Mask Register (IMR).....	272
26.6.2.13. Interrupt Status Register (ISR).....	273
26.6.2.14. Raw Interrupt Status Register (RISR).....	274
26.6.2.15. Transmit FIFO Overflow Interrupt Clear Registers (TXOICR) .....	275
26.6.2.16. Receive FIFO Overflow Interrupt Clear Register (RXOICR).....	276
26.6.2.17. Receive FIFO Underflow Interrupt Clear Register (RXUICR).....	277
26.6.2.18. Interrupt Clear Register (ICR) .....	278
26.6.2.19. DMA Control Register (DMACR).....	279
26.6.2.20. DMA Transmit Data Level (DMATDLR).....	280
26.6.2.21. DMA Receive Data Level (DMARDLR).....	281
26.6.2.22. Identification Register (IDR).....	282
26.6.2.23. Version ID Register (VIDR) .....	283
26.6.2.24. SSI Data Register (DRx).....	284
26.6.2.25. RX Sample Delay Register (RXSDR) .....	285

26.6.2.26. SPI Control Register 0 (SPICTRLR0) .....	286
26.6.2.27. XIP Mode Bits (XIPMBR) .....	287
26.6.2.28. XIP Incr Inst Register (XIPHIR) .....	288
26.6.2.29. XIP Wrap Inst Register (XIPWIR) .....	289
26.6.2.30. XIP Control Register (XIPCR) .....	290
26.6.2.31. XIP Slave Enable Register (XIPSER) .....	292
26.6.2.32. XIP Receive FIFO Overflow Interrupt Clear Register (XRXIOCR) .....	293
26.6.2.33. XIP Continus Transfer Time Out Register (XIPCTTOR) .....	294
<b>26.7. Functional Description.....</b>	<b>295</b>
26.7.1. Master Mode.....	295
26.7.2. Clock Ratios .....	295
26.7.3. Receive and Transmit FIFO Buffers .....	296
26.7.3.1. Transmit FIFO .....	296
26.7.3.2. Receive FIFO .....	296
26.7.4. DMA Operation .....	296
26.7.5. SSI Interrupts.....	296
26.7.6. Enhanced SPI Modes.....	297
26.7.7. Execute In Place (XIP) Mode .....	298
26.7.8. Continuous Transfer Mode in XIP .....	299
26.7.9. Data Pre-fetch in XIP Operations .....	299
<b>27. Inter-Integrated Circuit (I2C).....</b>	<b>300</b>
27.1. Introduction .....	300
27.2. Features .....	300
27.3. System And Block Diagram.....	301
27.4. Memory Map and Registers .....	301
27.4.1. Memory Map.....	301
27.4.2. Register Descriptions .....	302
27.4.2.1. I2C Status Register (I2CS).....	302
27.4.2.2. I2C Clock Prescalar Register (I2CP) .....	303
27.4.2.3. I2C Control Register (I2CC).....	304
27.4.2.4. I2C Slave Address Register (I2CSA) .....	305
27.4.2.5. I2C Port Control Register (I2CPCR) .....	306
27.4.2.6. I2C Slave High-Speed Mode Indicator Register (I2CSHIR) .....	307
27.4.2.7. I2C Slave SDA Hold Time Register (I2CSHT) .....	307
27.4.2.8. I2C Data Register (I2CD).....	308
27.4.2.9. I2C Port Direction Register (I2CDDR) .....	308
27.4.2.10. I2C Port Data Register (I2CPDR) .....	309
27.5. Functional Description.....	309
27.5.1. Master Mode.....	309
27.5.2. Slave Mode.....	309
27.5.3. Protocol .....	310
27.5.4. Arbitration Procedure .....	311
27.5.5. Clock Synchronization.....	311
27.5.6. Handshaking.....	311
27.5.7. Clock Stretching .....	312

27.5.8. High-Speed Mode Operation.....	312
27.5.9. Software Transaction Flow Diagrams.....	314
<b>28. Pulse Width Modulator (PWM) .....</b>	<b>317</b>
28.1. Introduction .....	317
28.2. Features.....	317
28.3. Block Diagram.....	318
28.4. Signal Description .....	318
28.5. Memory Map and Registers .....	319
28.5.1. Memory Map.....	319
28.5.2. Register Descriptions .....	320
28.5.2.1. PWM Pre-scale Register (PPR).....	320
28.5.2.2. PWM Clock Select Register (PCSR) .....	321
28.5.2.3. PWM Control Register (PCR) .....	322
28.5.2.4. PWM Counter Register (PCNRn) .....	324
28.5.2.5. PWM Comparator Register (PCMRn).....	326
28.5.2.6. PWM Timer Register (PTRn) .....	329
28.5.2.7. PWM Interrupt Enable Register (PIER) .....	331
28.5.2.8. PWM Interrupt Flag Register (PIFR).....	332
28.5.2.9. PWM Capture Control Register (PCCR0/1).....	333
28.5.2.10. PWM Capture Rising Latch Register (PCRLRn).....	335
28.5.2.11. PWM Capture Falling Latch Register (PCFLRn) .....	337
28.5.2.12. PWM Port Control Register (PPCR) .....	339
28.6. Functional Descriptions .....	340
28.6.1. PWM Double Buffering And Automatic Reload .....	340
28.6.2. Modulate Duty Ratio .....	340
28.6.3. Dead-Zone Generator .....	341
28.6.4. PWM Timer Start Procedure .....	341
28.6.5. PWM Timer Stop Procedure.....	341
28.6.6. Capture Start Procedure .....	342
28.6.7. Capture Basic Timer Operation .....	342
<b>29. Analog Comparator (COMP) .....</b>	<b>343</b>
29.1. Introduction .....	343
29.2. Block Diagram.....	343
29.3. Modes of Operation.....	344
29.3.1. Wait Mode.....	344
29.3.2. Doze Mode .....	344
29.3.3. Stop Mode .....	344
29.4. Memory Map and Registers .....	345
29.4.1. Memory Map.....	345
29.4.2. Register Descriptions .....	345
29.4.2.1. Comparator Control Register (CPTCN) .....	345
29.4.2.2. Comparator Mode Selection Register (CPTMD).....	346
29.4.2.3. Comparator MUX Selection Register (CPTMX).....	347
29.4.2.4. Comparator Output Filter Selection Register (CPTFLS).....	348

29.5. Function Description .....	348
<b>30. USB2.0 Full-Speed Device Controller (USBC).....</b>	<b>350</b>
30.1. Introduction .....	350
30.2. Features .....	350
30.3. Block Diagram.....	351
30.4. Modes Of Operation.....	351
30.4.1. Wait Mode.....	351
30.4.2. Doze Mode .....	351
30.4.3. Stop Mode .....	351
30.5. Memory Map and Registers .....	352
30.5.1. Memory Map.....	352
30.5.2. Register Descriptions .....	353
30.5.2.1. USBPHY Control Register 1 (USBPHY_CTRL1).....	353
30.5.2.2. Interrupt Status Register (INT_STAT) .....	354
30.5.2.3. Interrupt Enable Register (INT_ENB) .....	355
30.5.2.4. Error Interrupt Status Register (ERR_STAT) .....	356
30.5.2.5. Error Interrupt Enable Register (ERR_ENB).....	358
30.5.2.6. Status Register (STAT).....	359
30.5.2.7. Control Register (CTL).....	360
30.5.2.8. Address Register (ADDR).....	362
30.5.2.9. EBT Page Register 1 (EBT_PAGE_01) .....	363
30.5.2.10. Frame Number Register (FRMNUML) .....	364
30.5.2.11. Frame Number Register (FRMNUMH) .....	365
30.5.2.12. Token Register (TOKEN) .....	366
30.5.2.13. SOF Threshold Register (SOF_THLD) .....	367
30.5.2.14. EBT Page Register 2 (EBT_PAGE_02) .....	368
30.5.2.15. EBT Page Register 3 (EBT_PAGE_03) .....	369
30.5.2.16. Endpoint Control Registers (ENDPTn) .....	369
30.5.2.17. USBPHY Control Register 2 (USBPHY_CTRL2).....	371
30.5.2.18. USB PHY Observe Register (USB_PHY_OBSERVE).....	372
30.5.2.19. USBPHY GPIO Register (USB_PHY_GPIO).....	373
30.5.2.20. USB Resume Enable Register (USB_RESMEN) .....	374
30.5.2.21. USB PHY Control Register3 (USBPHY_CTRL3).....	375
30.5.2.22. USB PHY Control Register4 (USBPHY_CTRL4).....	376
30.6. Function Description .....	377
30.6.1. Data Structure .....	377
30.6.2. Endpoint Buffer Table .....	377
30.6.3. Rx vs. Tx As A USB Target Device .....	378
30.6.4. Addressing Endpoint Buffer Table Entries .....	378
30.6.5. Endpoint Buffer Table Formats .....	379
30.6.6. USB Transaction.....	382
<b>31. Analog-to-Digital Converter (ADC) .....</b>	<b>384</b>
31.1. Introduction .....	384
31.2. ADC Main Features.....	384

31.3. ADC Functional Description .....	385
31.3.1. ADC On-Off Control (ADEN, ADDIS, ADRDY).....	386
31.3.2. ADC Clock.....	387
31.3.3. Configuring The ADC.....	387
31.3.4. Channel Selection (CCWi) .....	387
31.3.5. Programmable Sampling Time (SMP).....	388
31.3.6. Single Conversion Mode (CONT = 0) .....	388
31.3.7. Continuous Conversion Mode (CONT = 1) .....	389
31.3.8. Starting Conversions (ADSTART).....	389
31.3.9. Timings .....	390
31.3.10. Stopping An Ongoing Conversion (ADSTP).....	391
31.4. Conversion On External Trigger And Trigger Polarity.....	391
31.4.1. Discontinuous Mode (DISCEN).....	393
31.4.2. Programmable Resolution (RES) - Fast Conversion Mode .....	393
31.4.3. End of Conversion, End of Sampling Phase (EOC, EOSMP Flags).....	393
31.4.4. End Of Conversion Sequence (Eoseq Flag).....	393
31.4.5. Example Timing Diagrams .....	394
31.5. Data Management .....	396
31.5.1. Data FIFO & Data Alignment (ADC_FIFO, ALIGN).....	396
31.5.2. ADC Overrun (OVR, OVRMOD) .....	396
31.5.3. Managing A Sequence Of Data Converted Without Using The DMA .....	397
31.5.4. Managing Converted Data Without Using The DMA Without Overrun .....	397
31.5.5. Managing Converted Data Using The DMA.....	397
31.6. Low Power Features .....	398
31.6.1. Wait Mode Conversion .....	398
31.6.2. Auto-off Mode (AUTOFF) .....	398
31.7. Analog Window Watchdog (AWD).....	399
31.8. Temperature Sensor.....	399
31.9. ADC Interrupts .....	400
31.10. Memory Map and Registers.....	401
31.10.1. Memory Map.....	401
31.10.2. Register Descriptions .....	402
31.10.2.1. ADC Interrupt And Status Register (ADC_ISR).....	402
31.10.2.2. ADC Interrupt Enable Register (ADC_IER) .....	404
31.10.2.3. ADC Control Register (ADC_CR) .....	405
31.10.2.4. ADC Configuration Register 1 (ADC_CFGR1) .....	407
31.10.2.5. ADC Configuration Register 2 (ADC_CFGR2) .....	409
31.10.2.6. ADC Sampling Time Register (ADC_SMPR).....	410
31.10.2.7. ADC Watch Dog Register (ADC_WDG).....	411
31.10.2.8. ADC Watchdog Threshold Register (ADC_TR) .....	412
31.10.2.9. ADC Channel Selection Register (ADC_CHSELR).....	413
31.10.2.10. ADC FIFO Access Register (ADC_FIFO).....	414
32. Electrical Characteristic .....	415
32.1. Absolute Maximum Ratings .....	415



32.2. Electrostatic Discharge (ESD) Protection.....	415
32.3. DC Electrical Specification .....	416
32.4. VDD Power Up Timing .....	417
33. Application Circuit.....	418
34. Package Information.....	423
34.1. LT168A (QFN-48pin).....	423
34.2. LT168B (QFN-68pin).....	425
35. Copyright .....	426

Levetop Semiconductor

**Figure List**

Figure 1-1: LT168A and LT168B Outline..... 30

Figure 1-2: Internal Block Diagram ..... 30

Figure 1-3: System Block of LT168..... 37

Figure 1-4: Application Diagram of TFT LCD Module..... 37

Figure 2-1: LT168A (QFN48) Pin Assignment..... 38

Figure 2-2: LT168B (QFN68) Pin Assignment..... 39

Figure 3-1: UART Connection between LT168x and Host MCU..... 50

Figure 3-2: LT168A Connect to 8-bits MCU Panel ..... 50

Figure 3-3: LT168B Connect to 16-bits MCU Panel..... 50

Figure 3-4: LT168B Connect to RGB Type TFT Panel..... 51

Figure 3-5: LT168B Connect to QSPI Panel..... 52

Figure 3-6: LT168x Connect to QSPI Flash..... 52

Figure 3-7: LT168x Connect to Resistive Touch Panel ..... 53

Figure 3-8: LT168x Connect to Capacitive Touch Panel..... 53

Figure 3-9: External Clock Circuit ..... 53

Figure 3-10: TFT LCD Backlight Circuit Example 1 ..... 54

Figure 3-11: TFT LCD Backlight Circuit Example 2 ..... 54

Figure 3-12: Canbus Circuit Example ..... 55

Figure 3-13: Audio Output Application Circuit..... 55

Figure 3-14: RTC Application Circuit..... 56

Figure 3-15: External Reset Circuit..... 56

Figure 3-16: LT168x Connect to USB Connector..... 57

Figure 3-17: Extension for SPI and Uart Device..... 57

Figure 4-1: Address Map..... 58

Figure 5-1: Interrupt Control Status Register (ICSR)..... 61

Figure 5-2: Interrupt Enable Register (IER)..... 63

Figure 5-3: Interrupt Pending Set Register (IPSR)..... 64

Figure 5-4: Interrupt Pending Clear Register (IPCR)..... 65

Figure 5-5: Priority Level Select Registers (PLSR0-PLSR31)..... 66

Figure 5-6: System Priority Level Select Registers (SYSPLSR) ..... 67

Figure 5-7: One Pulse Interrupt without Conflicion ..... 68

Figure 5-8: Level-sensitive Interrupt without Conflicion ..... 69

Figure 5-9: Two Interrupts Occur at the Same Time..... 69

Figure 5-10: A lower Priority Interrupt Asserted with Conflicion ..... 69

Figure 5-11: A higher Priority Interrupt Asserted with Conflicion..... 69

Figure 6-1: Programming Model..... 76

Figure 6-2: Data Organization in Memory ..... 77

Figure 6-3: Data Organization in Registers ..... 77

Figure 7-1: EPT Control Status Register (EPTCSR) ..... 82

Figure 7-2: EPT Reload Register (EPTRLD)..... 83

Figure 7-3: EPT Counter Register (EPTCNT)..... 84

Figure 7-4: EPT Count Timing..... 85

Figure 8-1: Wakeup Configuration Register (WKUPC)..... 87

Figure 8-2: Chip Pin Pull Down Configuration Register (CPPDC) ..... 89

Figure 8-3: QSPI XIP Mode Configuration Register (QSPIXIPMCFR)..... 90

Figure 8-4: QSPI 32-Bit Key Register (QSPILKEYR)..... 91

Figure 8-5: QSPI GPIO Configuration Register (QSPIGPIOCR)..... 91

Figure 8-6: MCU Access RAM Priority Configuration Register (MCURAMPRIOCR) ..... 92

Figure 8-7: EPORT2 Function Configuration Register (EPORT2FCR)..... 94

Figure 9-1: Clock Structure..... 95

Figure 9-2: Synthesizer Control Register (SYNCR)..... 98

Figure 9-3: Low Speed Oscillator Control and Status Register (LOSCCSR) ..... 102

Figure 9-4: PLL Configuration and Status Register (PLLCSR) ..... 103

Figure 9-5: Module Stop Control Register (MSCR) ..... 105

Figure 9-6: EPT External Clock Source Enable Control Register (ECSECR)..... 107

Figure 9-7: OSC Bist Test Configuration Register1 (OBTCR1)..... 108

Figure 9-8: OSC Bist Test Configuration Register2 (OBTCR2)..... 108

Figure 9-9: OSC Bist Test Control Register (OBTCTLR)..... 109

Figure 9-10: OSC BIST Test Counter Register (OBTCTR)..... 110

Figure 9-11: OSC BIST Test Result Register (OBTRR)..... 111

Figure 10-1: Reset Controller Block Diagram..... 113

Figure 10-2: Reset Test Register (RTR) ..... 114

Figure 10-3: Reset Status Register (RSR)..... 115

Figure 10-4: Reset Control Register (RCR)..... 116

Figure 10-5: Reset Control Flow..... 117

Figure 12-1: Cache Module Block Diagram..... 120

Figure 12-2: Cache Control Register (LMEM\_CCR) ..... 121

Figure 12-3: Cache Line Control Register (LMEM\_CLCR)..... 122

Figure 12-4: Cache Search Address Register (LMEM\_CSAR)..... 124

Figure 12-5: Cache Read/Write Value Register (LMEM\_CCVR)..... 125

Figure 12-6: Cache Access Register (LMEM\_ACRG)..... 126

Figure 12-7: Cache Page Invalidate Base Address Register (LMEM\_PAGE\_INV\_BADDR)..... 127

Figure 12-8: Cache Page Invalidate Size Register (LMEM\_PAGE\_INV\_SIZE)..... 128

Figure 12-9: Cache Clock Enable Register (LMEM\_CACHE\_CLK\_EN) ..... 129

Figure 12-10: Cache Tag and Data Access Structure ..... 130

Figure 13-1: XBAR Device-Specific Block Diagram ..... 135

Figure 13-2: Master Priority Register (XBAR\_MPRn) ..... 137

Figure 13-3: Slave General Purpose Control Register (XBAR\_SGPCRn)..... 139

Figure 14-1: DMA Block Diagram..... 145

Figure 15-1: Efuse Module Configuration Register(EFMCR) .....	148
Figure 15-2: Efuse Read Timing.....	149
Figure 15-3: Efuse Read Data Register(EFMDR).....	149
Figure 16-1: Programmable Voltage Detector Configuration Register (PVDC).....	150
Figure 16-2: Customer Configuration Register (CCR) .....	152
Figure 16-3: External High Speed Oscillator Stable Time Configuration Register (EOSCST) .....	155
Figure 16-4: PLL Lock Time Configuration Register (PLLOCKCR).....	155
Figure 16-5: RESET Pin Filter Enable and Value Register (RFEVR).....	156
Figure 16-6: Programmable Voltage Detector Filter Enable and Value Register (PVDFEVR).....	157
Figure 16-7: Factory Configuration Register (FCR).....	158
Figure 16-8: ADC Channel Disable Configuration Register (ADCCDISR) .....	159
Figure 16-9: VREF Trimming Configuration Register (VREFTCR) .....	160
Figure 16-10: LDO1P2 Trimming Configuration Register (LDOTCR).....	161
Figure 16-11: RTC Trimming Configuration Register (RTCTCR).....	162
Figure 17-1: RGB Controller Block Diagram .....	163
Figure 17-2: Display RAM Access Area by RGB Interface.....	164
Figure 17-3: Timing Chart of Signals in RGB Interface.....	165
Figure 17-4: Horizontal Pulse Width and Back Porch Register (THPB) .....	167
Figure 17-5: Horizontal Period and Front Porch Register (THDF) .....	168
Figure 17-6: Vertical Pulse Width and Back Porch Register (TVPB).....	169
Figure 17-7: Vertical Period and Front Porch Register (TVDF).....	170
Figure 17-8: Memory Start Address Register (MSADDR).....	171
Figure 17-9: Memory End Address Register (MEADDR) .....	172
Figure 17-10: Number of Frames Transferred Register (NFTR).....	173
Figure 17-11: Number of Frames Transferred Threshold Configuration Register(NFTTCR) .....	174
Figure 17-12: RGB Control and Status Register(RCSR).....	175
Figure 18-1: Blender Block Diagram .....	178
Figure 18-2: Blender Control and Status Register(BCSR).....	180
Figure 18-3: Source Address Register(SADDR) .....	181
Figure 18-4: Transfer Total Address Length Register (TTALR) .....	182
Figure 18-5: Destination Address Register(DADDR).....	182
Figure 18-6: Transfer Destination Minor Length and Offset Register (TDMLOR).....	183
Figure 19-1: EBI Control Register (EBICR) .....	185
Figure 19-2: IO Pull Control Register(IOPCR) .....	187
Figure 19-3: IO Control Register(IOCR) .....	188
Figure 19-4: GPIO Data Output Register(GPIODO).....	189
Figure 19-5: GPIO Direction Register(GPIODIR).....	190
Figure 19-6: GPIO Data Input Register(GPIODI) .....	191
Figure 19-7: EBI_WR#, EBI_RD# Assert/Negate Timing .....	192
Figure 20-1: PIT Block Diagram .....	195

Figure 20-2: PIT Modulus Register (PMR).....197

Figure 20-3: PIT Control and Status Register (PCSR).....197

Figure 20-4: PIT Count Register (PCNTR) .....199

Figure 20-5: Counter Reloading from the Modulus Latch .....200

Figure 20-6: Counter in Free-Running Mode .....200

Figure 21-1: Watchdog Timer Block Diagram.....203

Figure 21-2: Watchdog Modulus Register (WMR) .....204

Figure 21-3: Watchdog Control Register (WCR) .....205

Figure 21-4: Watchdog Service Register (WSR).....207

Figure 21-5: Watchdog Count Register (WCNTR).....207

Figure 22-1: RTC Block Diagram .....208

Figure 22-2: RTC Application Circuit.....209

Figure 23-1: EPORT Block Diagram .....210

Figure 23-2: EPORT Port Interrupt Enable Register (EPIER).....212

Figure 23-3: EPORT Data Direction Register (EPDDR).....212

Figure 23-4: EPORT Pin Assignment Register (EPPAR) .....213

Figure 23-5: EPORT Pin Pull-up enable Register (EPPUE).....214

Figure 23-6: EPORT Port Flag Register (EPFR).....214

Figure 23-7: EPORT Port Pin Data Register (EPPDR).....215

Figure 23-8: EPORT Port Data Register (EPDR).....215

Figure 23-9: EPORT Port Bit Set Register (EPBSR).....215

Figure 23-10: EPORT Digital Filter Control Register (EPFC).....216

Figure 23-11: EPORT Open Drain enable Register (EPODE) .....216

Figure 23-12: EPORT Level Polarity Register (EPLPR).....217

Figure 23-13: EPORT Port Bit Clear Register (EPBCR).....217

Figure 24-1: CANBus Block Diagram.....218

Figure 24-2: Canbus Circuit Example .....220

Figure 25-1: SCI Transmitter Block Diagram.....222

Figure 25-2: SCI Receiver Block Diagram .....223

Figure 25-3: SCI Version ID Register (SCI\_VERID).....225

Figure 25-4: SCI Parameter Register (SCI\_PARAM).....225

Figure 25-5: SCI Reset Register (SCI\_RESET) .....226

Figure 25-6: SCI Pin Register (SCI\_PIN).....227

Figure 25-7: SCI Baud Rate Register (SCI\_BAUD).....227

Figure 25-8: SCI Status Register (SCI\_STAT).....229

Figure 25-9: SCI Control Register (SCI\_CTRL).....233

Figure 25-10: SCI Data Register (SCI\_DATA) .....237

Figure 25-11: SCI Match Address Register (SCI\_MATCH).....238

Figure 25-12: SCI Modem IrDA Register (SCI\_MODIR).....239

Figure 25-13: SCI FIFO Register (SCI\_FIFO).....241

Figure 25-14: SCI Watermark Register (SCI_WATER).....	243
Figure 25-15: SCI Oversampling Ratio Register (SCI_OSR).....	244
Figure 25-16: SCI Baud Rate Generation.....	245
Figure 26-1: SSI Block Diagram.....	256
Figure 26-2: SSI Application Diagram.....	256
Figure 26-3: Control Register 0 (CTRLR0).....	258
Figure 26-4: Control Register 1 (CTRLR1).....	261
Figure 26-5: SSI Enable Register (SSIENR).....	262
Figure 26-6: Microwire Control Register (MWCR).....	263
Figure 26-7: Slave Enable Register (SER).....	264
Figure 26-8: Baud Rate Select (BAUDR).....	265
Figure 26-9: Transmit FIFO Threshold Level (TXFTLR).....	266
Figure 26-10: Receive FIFO Threshold Level (RXFTLR).....	267
Figure 26-11: Transmit FIFO Level Register (TXFLR).....	268
Figure 26-12: Receive FIFO Level Register (RXFLR).....	269
Figure 26-13: Status Register (SR).....	270
Figure 26-14: Interrupt Mask Register (IMR).....	272
Figure 26-15: Interrupt Status Register (ISR).....	273
Figure 26-16: Raw Interrupt Status Register (RISR).....	274
Figure 26-17: Transmit FIFO Overflow Interrupt Clear Registers (TXOICR).....	275
Figure 26-18: Receive FIFO Overflow Interrupt Clear Register (RXOICR).....	276
Figure 26-19: Receive FIFO Underflow Interrupt Clear Register (RXUICR).....	277
Figure 26-20: Interrupt Clear Register (ICR).....	278
Figure 26-21: DMA Control Register (DMACR).....	279
Figure 26-22: DMA Transmit Data Level (DMATDLR).....	280
Figure 26-23: DMA Receive Data Level (DMARDLR).....	281
Figure 26-24: Identification Register (IDR).....	282
Figure 26-25: Version ID Register (VIDR).....	283
Figure 26-26: SSI Data Register (DRx).....	284
Figure 26-27: RX Sample Delay Register (RXSDR).....	285
Figure 26-28: SPI Control Register 0 (SPICTRLR0).....	286
Figure 26-29: XIP Mode Bits (XIPMBR).....	287
Figure 26-30: XIP Incr Inst Register (XIPHIR).....	288
Figure 26-31: XIP Wrap Inst Register (XIPWIR).....	289
Figure 26-32: XIP Control Register (XIPCR).....	290
Figure 26-33: XIP Slave Enable Register (XIPSER).....	292
Figure 26-34: XIP Receive FIFO Overflow Interrupt Clear Register (XRXIOCR).....	293
Figure 26-35: XIP Continus Transfer Time Out Register (XIPCTTOR).....	294
Figure 26-36: SSI Configured as Master Device.....	295
Figure 26-37: Typical Write Operation Dual/Quad/Octal SPI Mode.....	297

Figure 26-38: Typical Read Operation Dual/Quad/Octal SPI Mode.....	297
Figure 26-39: XIP Transfer with Instruction Phase.....	298
Figure 27-1: I2C Block Diagram .....	301
Figure 27-2: The I2CS Register Shows the Status of I2C Module.....	302
Figure 27-3: I2C Clock Prescaler Register (I2CP) .....	303
Figure 27-4: I2C Control Register (I2CC).....	304
Figure 27-5: I2C Slave Address Register (I2CSA).....	305
Figure 27-6: I2C Port Control Register (I2CPCR) .....	306
Figure 27-7: I2C Slave High-Speed Mode Indicator Register (I2CSHIR).....	307
Figure 27-8: I2C Slave SDA Hold Time Register (I2CSHT).....	307
Figure 27-9: I2C Data Register (I2CD) .....	308
Figure 27-10: I2C Port Direction Register (I2CDDR).....	308
Figure 27-11: I2C Port Data Register (I2CPDR).....	309
Figure 27-12: I2C Communication Protocol.....	310
Figure 27-13: Repeat Start of I2C Protocol.....	311
Figure 27-14: SCL Synchronization .....	312
Figure 27-15: Data Transfer Format in HS Mode .....	313
Figure 27-16: A Complete HS Mode Transfer .....	313
Figure 27-17: Slave Mode Initialization.....	314
Figure 27-18: Master Mode Initialization .....	314
Figure 27-19: Interrupt Transaction .....	315
Figure 28-1: PWM Block Diagram .....	318
Figure 28-2: PWM Pre-scale Register (PPR).....	320
Figure 28-3: PWM Clock Select Register(PCSR) .....	321
Figure 28-4: PWM Control Register (PCR).....	322
Figure 28-5: PWM Counter Register (PCNR) .....	325
Figure 28-6: PWM Comparator Register (PCMR).....	327
Figure 28-7: PWM Output.....	328
Figure 28-8: PWM Duty Cycle .....	328
Figure 28-9: PWM Timer Register (PTR) .....	330
Figure 28-10: PWM Interrupt Enable Register (PIER) .....	331
Figure 28-11: PWM Interrupt Flag Register (PIFR).....	332
Figure 28-12: PWM Capture Control Register (PCCR0/1).....	333
Figure 28-13: PWM Capture Rising Latch Register (PCRLR0/1/2/3).....	336
Figure 28-14: PWM Capture Falling Latch Register (PCFLR0/1/2/3).....	338
Figure 28-15: PWM Port Control Register (PPCR).....	339
Figure 28-16: PWM Double Buffering Illustration .....	340
Figure 28-17: PWM Controller Output Duty Ratio.....	340
Figure 28-18: Dead Zone Generation Operation.....	341
Figure 28-19: Capture Basic Timer Operation .....	342

Figure 29-1: Comparator Block Diagram.....343

Figure 29-2: Comparator Control Register (CPTCN) .....345

Figure 29-3: Comparator Mode Selection Register(CPTMD).....346

Figure 29-4: Comparator MUX Selection Register(CPTMX) .....347

Figure 29-5: Comparator Output Filter Selection Register(CPTFLS) .....348

Figure 29-6: Comparator Hysteresis Plot.....349

Figure 30-1: USB Full-speed Device(USB module) Block Diagram .....351

Figure 30-2: USBPHY Control Register 1 (USBPHY\_CTRL1).....353

Figure 30-3: Interrupt Status Register (INT\_STAT).....354

Figure 30-4: Interrupt Enable Register (INT\_ENB).....355

Figure 30-5: Error Interrupt Status Register (ERR\_STAT) .....356

Figure 30-6: Error Interrupt Enable Register (ERR\_ENB).....358

Figure 30-7: Status Register (STAT) .....359

Figure 30-8: Control Register (CTL).....360

Figure 30-9: Address Register (ADDR) .....362

Figure 30-10: EBT Page Register 1 (EBT\_PAGE\_01).....363

Figure 30-11: Frame Number Register (FRMNUML) .....364

Figure 30-12: Frame Number Register (FRMNUMH) .....365

Figure 30-13: Token Register (TOKEN).....366

Figure 30-14: SOF Threshold Register (SOF\_THLD).....367

Figure 30-15: EBT Page Register 2 (EBT\_PAGE\_02).....368

Figure 30-16: EBT Page Register 3 (EBT\_PAGE\_03).....369

Figure 30-17: Endpoint Control Registers(ENDPTn, n = 0-7) .....370

Figure 30-18: USBPHY Control Register 2 (USBPHY\_CTRL2) .....371

Figure 30-19: USB PHY Observe Register (USB\_PHY\_OBSERVE).....372

Figure 30-20: USB PHY GPIO Register (USB\_PHY\_GPIO).....373

Figure 30-21: USB Resume Enable Register (USB\_RESMEN).....374

Figure 30-22: USB PHY Control Register 3 (USBPHY\_CTRL3).....375

Figure 30-23: USB PHY Control Register 4 (USBPHY\_CTRL4).....376

Figure 30-24: Endpoint Buffer Table .....377

Figure 30-25: USB Token Transaction .....382

Figure 31-1: ADC Block Diagram .....385

Figure 31-2: Enabling/Disabling the ADC.....386

Figure 31-3: ADC Clock Scheme .....387

Figure 31-4: Analog to Digital Conversion Time .....390

Figure 31-5: Stopping an Ongoing Conversion.....391

Figure 31-6: Single conversions of a sequence, software trigger.....394

Figure 31-7: Continuous Conversion of a Sequence, Software Trigger.....394

Figure 31-8: Single Conversions of a Sequence, Hardware Trigger.....394

Figure 31-9: Continuous Conversions of a Sequence, Hardware Trigger.....395



Figure 31-10: Analog Watchdog Guarded Area .....	399
Figure 31-11: ADC Interrupt and Status Register (ADC_ISR).....	402
Figure 31-12: ADC Interrupt Enable Register (ADC_IER).....	404
Figure 31-13: ADC Control Register (ADC_CR).....	405
Figure 31-14: ADC Configuration Register 1 (ADC_CFGR1).....	407
Figure 31-15: ADC Configuration Register 2 (ADC_CFGR2).....	409
Figure 31-16: ADC Sampling Time Register (ADC_SMPR).....	410
Figure 31-17: ADC Watchdog Register (ADC_WDG) .....	411
Figure 31-18: ADC Watchdog Threshold Register (ADC_TR) .....	412
Figure 31-19: ADC Channel Selection Register 1(ADC_CHSELR1).....	413
Figure 31-20: ADC Channel Selection Register 2(ADC_CHSELR2).....	413
Figure 31-21: ADC FIFO Access Register (ADC_FIFO).....	414
Figure 33-1: VDD Power up Timing Requirement .....	417
Figure 33-1: Application Circuit – LT168A for 8bit MCU Panel.....	418
Figure 33-2: Application Circuit – LT168A for SPI Panel .....	419
Figure 33-3: Application Circuit – LT168B for 16bit MCU Panel .....	420
Figure 33-4: Application Circuit – LT168B for RGB TFT Panel .....	421
Figure 33-5: Application Circuit – LT168B for QSPI Panel.....	422
Figure 34-1: LT168A Package Overview.....	423
Figure 34-2: LT168B Package Overview.....	425

## Table List

Table 1-1: LT168 Part Number .....	29
Table 2-1: Signal Properties .....	40
Table 2-2: Signal Description.....	44
Table 2-3: LT168x Comparison.....	49
Table 3-1: The differences of LT168B for RGB Panel with or without PSRAM.....	51
Table 4-1: Register Address Location Map .....	59
Table 5-1: Interrupt Controller Module Memory Map .....	60
Table 5-2: Priority Value Adjustment.....	66
Table 5-3: Priority Value Adjustment.....	67
Table 5-4: Priority Value Adjustment.....	68
Table 5-5: Interrupt Source Assignment.....	70
Table 6-1: 32-bits RISC Instruction Set.....	78
Table 7-1: Programmable Timer Module Memory Map.....	81
Table 8-1: CCM Memory Map .....	86
Table 8-2: WKUPSEN and Corresponding Wakeup Source.....	87
Table 8-3: Chip Pin Pull Down Configuration.....	89
Table 8-4: EPORT2 Function Control Bit.....	94
Table 9-1: Clock Memory Map .....	97
Table 9-2: PLL Clock Divider .....	99
Table 9-3: ADC Clock Divider.....	100
Table 9-4: CLKOUTSEL Mode.....	101
Table 9-5: Sleep Operation Control Bit in Sleep Mode.....	101
Table 9-6: PLL Input Divider .....	104
Table 9-7: PLL VCO Output Clock Divider .....	104
Table 9-8: PLL Feedback Divider Value.....	105
Table 9-9: MS[29:0] Bits Corresponding Modules.....	106
Table 9-10: EPT Clock Divider .....	107
Table 10-1: Reset Controller Address Map .....	114
Table 10-2: Reset Source Summary.....	116
Table 12-1: Cache Module Memory Map .....	121
Table 12-2: Cache Set Commands.....	131
Table 12-3: Cache Line Commands .....	132
Table 12-4: Line Command Results .....	133
Table 13-1: XBAR Register Configuration Summary.....	136
Table 14-1: DMA Channel Assignment .....	144
Table 15-1: Efuse Register Memory Map.....	147
Table 16-1: Register Memory Map.....	150
Table 16-2: Programmable Voltage Detector .....	151
Table 16-3: RGB Interface Enable Control.....	152

Table 16-4: Programming Debug Interface Control .....	153
Table 16-5: RSTOUT Disable Control .....	153
Table 16-6: CLKOUT Disable Control .....	153
Table 16-7: QSPI2 Interface Enable Control .....	153
Table 16-8: ADC Channel Disable Control .....	159
Table 17-1: RGB Controller Module Memory Map .....	166
Table 17-2: Pixel Clock Divider .....	176
Table 18-1: Source Data Format .....	179
Table 18-2: Destination Data Format.....	179
Table 18-3: Blender Module Memory Map .....	179
Table 19-1: Signals Properties .....	184
Table 19-2: Shows The EBI Register Memory Map .....	184
Table 19-3: EBI_CS# Chip Select Wait States Encoding .....	186
Table 19-4: EBI Control Pin as EPORT1 .....	188
Table 19-5: EBI_WR# and EBI_RD# Assert and Negate Timing.....	192
Table 19-6: Chip Select Address Range Encoding .....	193
Table 19-7: 8-bits Port Data Transfer of EBI_CS# (Big Endian) .....	194
Table 19-8: 16-bits Port Data Transfer of EBI_CS# (Big Endian).....	194
Table 20-1: Programmable Interrupt Timer Module Memory Map.....	196
Table 20-2: Prescaler Select Encoding.....	197
Table 20-3: PIT Interrupt Requests.....	201
Table 21-1: Watchdog Timer Module Memory Map .....	203
Table 21-2: Watchdog Timer Prescaler.....	206
Table 23-1: Module Memory Map.....	211
Table 23-2: EPPAx Field Settings.....	213
Table 24-1: CANBus Signals .....	220
Table 25-1: Signal Properties .....	223
Table 25-2: SCI Module Memory Map.....	224
Table 25-3: Break Character Length.....	247
Table 25-4: Receiver Wakeup Options.....	249
Table 26-1: SSI Memory Map.....	257
Table 27-1: I2C Memory Map.....	301
Table 28-1: PWM Signal Description.....	318
Table 28-2: Module Memory Map.....	319
Table 28-3: Timer 3 Clock Source Selection.....	321
Table 29-1: Comparator Module Memory Map .....	345
Table 29-2: Comparator Mode Selection .....	347
Table 29-3: Comparator Negative Input MUX Selection .....	347
Table 29-4: Comparator Positive Input MUX Selection.....	347
Table 30-1: USB Module Memory Map.....	352

Table 30-2: Endpoint Enable/Direction Control .....	371
Table 30-3: USB PHY Oscillator Mode Selection .....	376
Table 30-4: Data Direction for USB Target Device .....	378
Table 30-5: EBT Address Calculation Fields .....	378
Table 30-6: Endpoint Buffer Table Byte Format .....	379
Table 30-7: Endpoint Buffer Table Byte Fields .....	380
Table 30-8: USB Responses to DMA Overrun Errors.....	383
Table 31-1: Channel Decode.....	388
Table 31-2: Configuring the Trigger Polarity.....	391
Table 31-3: Configuring the Trigger Polarity .....	392
Table 31-4: Data Alignment and Resolution.....	396
Table 31-5: Analog Watchdog Channel Selection .....	399
Table 31-6: ADC Interrupts .....	400
Table 31-7: ADC Module Memory Map.....	401
Table 32-1: Absolute Maximum Rating .....	415
Table 32-2: Electrostatic Discharge Protection Characteristic.....	415
Table 32-3: IO Static Characteristic (3.3V).....	416
Table 32-4: Power Characteristic .....	416
Table 32-5: Power Consumption .....	416
Table 32-6: Wake up Timing Characteristic .....	417
Table 32-6: VDD Power Up Characteristic.....	417
Table 34-1: LT168A Package Parameter.....	424
Table 34-2: LT168B Package Parameter.....	425

# 1. LT168 Introduction

## 1.1. Introduction

**LT168** is a serial Uart TFT display controller containing a built-in 32-bits MCU, Flash, and serial communication protocol. It supports TFT panels of max. 800\*480 resolution, with SPI/QSPI or 8/16-bits parallel MCU interface and TFT panels of max. 480\*480 resolution, with RGB interface.

LT168 supports the latest UI editing tool, UI\_Editor-II, which enables developers to easily implement their display layout and operations through built-in widgets, such as displaying pictures, GIF, progress bar, trend chart, text, QRcode, and much more. Once the UI design is done, developers may export the bin file (UartTFT\_Flash-II.bin) and program it to the external SPI flash connected to LT168. After power on, LT168 will then control the connected TFT panel to display the contents designed by the developers. In addition, the host MCU may communicate with LT168 by sending predefined commands through Uart port. LT168 will then interpret the received commands and execute them. For more information, please refer to UI\_Editor-II user manual.

LT168 has an embedded 32-bits RISC core which can operate at maximum 200MHz frequency, high-speed memories (including 768K Bytes SRAM and 8K Bytes ROM), a 32K Bytes two-way set associative cache, Quad SPI Flash memory interface, an extensive range of enhanced peripherals and I/Os. All models offer standard communication interfaces (Master QSPI x 3, UARTs x 3, USB2.0 FS device x 1, I2C and CANBus controller x 1), EBI (External Bus Interface, support 8080 bus for 8080 MCU panels), a set of RGB Panel Interface, Timers (up to four general-purpose 16-bits timers, advanced-control PWM timer x 8, asynchronous Watch Dog Timer x 1, and RTC timers x 1), and analog modules (1MSPS ADC with 8-channels x 1 and Comparators x 2). LT168 also supports Little VGL for smooth display effects and good cost performance.

Benefiting from its high capacity Flash and SRAM, LT168 can be used as a master MCU as well. It is ideal for electronic products with small size TFT-LCD panel such as smart home appliances, handheld control equipment, industrial control panels, electronic instruments, medical equipment, small testing equipment, small electric motorcycles, personal medical beauty, small testing equipment, charging equipment, water and electricity meters, smart speakers with panel, etc.

The operating temperature range of LT168 is -40°C to 105°C, and the operating voltage is 3.3V. LT168 has two part numbers as listed below:

**Table 1-1: LT168 Part Number**

Part	Package	Embedded Flash	Embedded SRAM	TFT Panel Solution
LT168A	QFN48 (6*6 mm <sup>2</sup> )	512K Bytes	256K Bytes + 512K Bytes	<ul style="list-style-type: none"> <li>● 8-bits 8080 MCU Type TFT Display</li> <li>● QSPI Type TFT Display</li> </ul>
LT168B	QFN68 (8*8 mm <sup>2</sup> )	2M Bytes	256K Bytes + 512K Bytes	<ul style="list-style-type: none"> <li>● 8/16-bits 8080 MCU Type TFT Display (Max. 800*480)</li> <li>● QSPI Type TFT Display</li> <li>● RGB 565 TFT Display (Max. 480*480)</li> </ul>

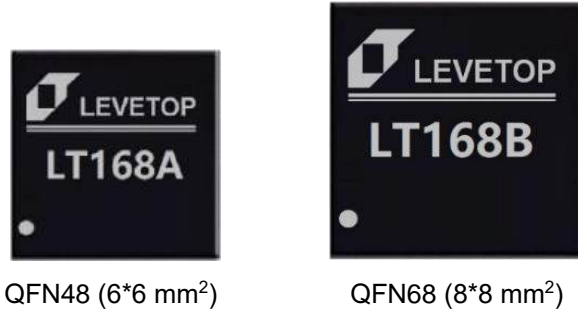


Figure 1-1: LT168A and LT168B Outline

**1.2. Internal Block Diagram**

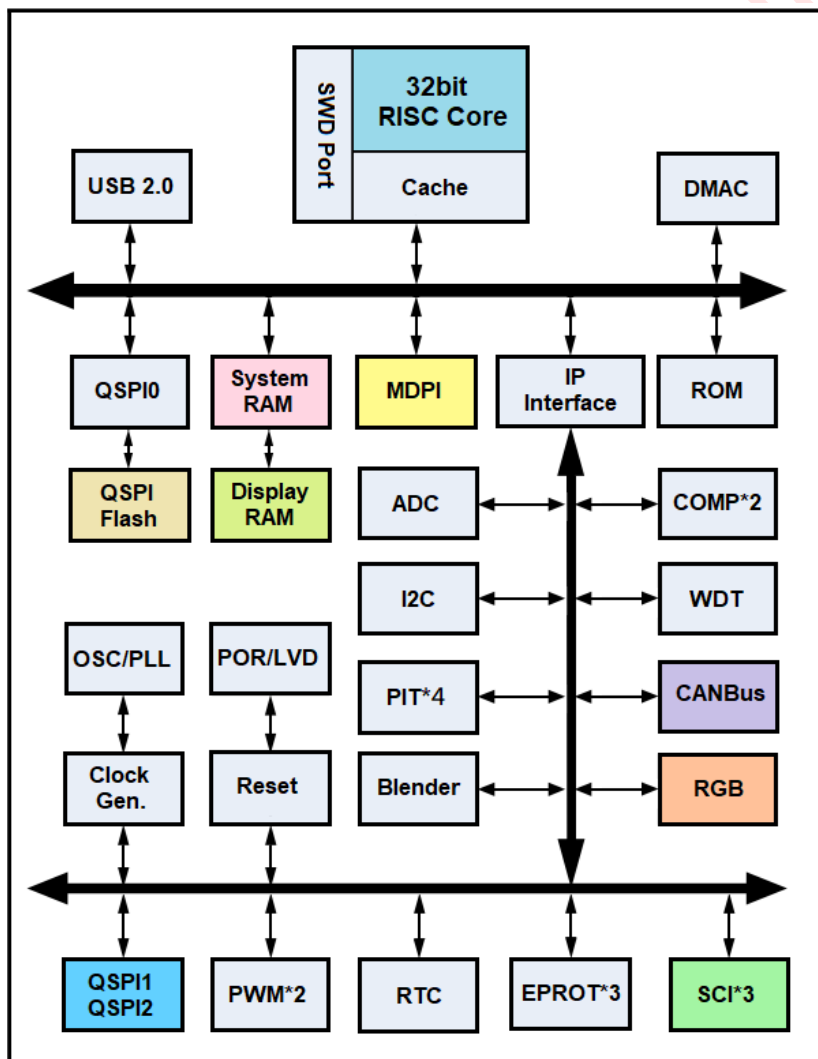


Figure 1-2: Internal Block Diagram

## 1.3. Features

### 1.3.1. 32-bits RISC Processor

- 32-bits load/store reduced instruction set computer (RISC) architecture with fixed 16-bits instruction length
- 16 entry 32-bits 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 32-bits x 32-bits hardware integer multiplier array
- 3~13 cycles hardware integer divider array

### 1.3.2. On-chip 768K Bytes Of Static Random-Access Memory (SRAM)

- Single cycle byte, half-word (16-bits), and word (32-bits) reads and writes
- Two segment for improving performance at certain application
  - System RAM: 256K Bytes and address range from 0x800000 to 0x83FFFF.
  - Display RAM: 512K Bytes and address range from 0x840000 to 0x8BFFFF

### 1.3.3. On-chip QSPI Flash

- LT168A embedded 512K Bytes QSPI Flash, support Levetop's Communication Program
- LT168B embedded 2M Bytes QSPI Flash, support Levetop's Communication Program

### 1.3.4. 8K Bytes Of Static Read-Only Memory (ROM)

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

### 1.3.5. 32K Bytes Of Cache Memory

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

### 1.3.6. External Bus Interface

- Programmable wait states -up to 16 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 16 chip selects asserted cycle can be programmed
- Programmable read/write asserted cycle
- Programmable read/write negated cycle
- Support 8/16-bits port size.
- Support 8080 standard bus
- Support 8080 MCU Type TFT Panel:
  - 800 x 480
  - 480 x 272
  - 480 x 320
  - 320 x 240 or Less

### 1.3.7. RGB Display Panel Interface

- Data frame format: Red: 5, Green: 6, Blue: 5
- Supported resolution ratio:
  - 480 x 480
  - 480 x 272 or Less
- Timing of the interface signals is configurable
  - Polarity and pixel clock polarity is configurable

### 1.3.8. DMA Controller

- 32 independent programmable DMA controller channels
- Data transfers in 8, 16, 32-bits
- Support single transfer, Burst 4, 8, 16 transfer, and burst always under a special case.
- Support single cycle transfer
- Support automatic transfer mode
- Support LLI transfer mode
- Follow a fixed priority rule

### 1.3.9. Reset Function

- 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 the last reset

### 1.3.10. Four Periodic Interval Timer

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

### 1.3.11. One Watchdog Timer

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



### 1.3.12. One RTC Timer

- Support loading time data to and reading 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

### 1.3.13. Three External Interrupts Port (EPORT)

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

### 1.3.14. Three Quad Serial Peripheral Interface Master Module (QSPI)

- 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 32-bits) – Item size of each data transfer is 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 32-bits
- Enhanced SPI support
- Execute in Place (XIP) mode support

### 1.3.15. Three Serial Communications Interface Module (SCI)

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13-bits 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 8-bits, 9-bits or 10-bits character length
- Programmable 1-bit or 2-bits 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 13-bits break character generation / 11-bits 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 the “receive” FIFO is not empty

### 1.3.16. One 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
- 64 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 1056 bytes (64 MBs) of SRAM used for MB storage
- Includes 256 bytes (64 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 (8-bits) 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 16-bits 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

### 1.3.17. One Usb2.0 Full Speed Compatible Device (USB)

- Supports internal reference clock or external 12MHz crystal reference clock
- Compliant with USB2.0 full speed specification with on-chip integrated PHY module
- Supports FS (12Mbps) mode
- Supports up to 8 endpoints including endpoint 0
- All endpoints except endpoint 0 can support interrupt and bulk transfer
- All endpoints except endpoint 0 can be configured as 8, 16, 32, 64 bytes FIFO size
- Endpoint 0 supports control transfer

### 1.3.18. Two Pulse Width Modulator (PWM)

- 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

### 1.3.19. ADC With 8-Channel

- High performance
  - 12-bits, 10-bits, 8-bits or 6-bits configurable resolution
  - ADC conversion time: 1.0  $\mu$ s for 12-bits resolution (1 MHz), 0.88  $\mu$ s conversion time for 10-bits resolution, faster conversion times can be obtained by lowering resolution.
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support
- Low power
  - PLCK frequency can be reduced for low power operation while still keeping optimum ADC performance. For example, 1.0  $\mu$ s conversion time is kept, whatever the frequency of PLCK.
  - Wait mode: to prevent ADC overrun in applications under 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
- 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

### 1.3.20. Two Analog Comparators

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

**1.3.21. Power Management Unit (PMU)**

- Support on-chip 1.2V LDO with maximum load current 100mA
- 1.2V LDO support two mode: lower power, high power

**1.3.22. Programmable Voltage Detector**

- Programmable voltage detector

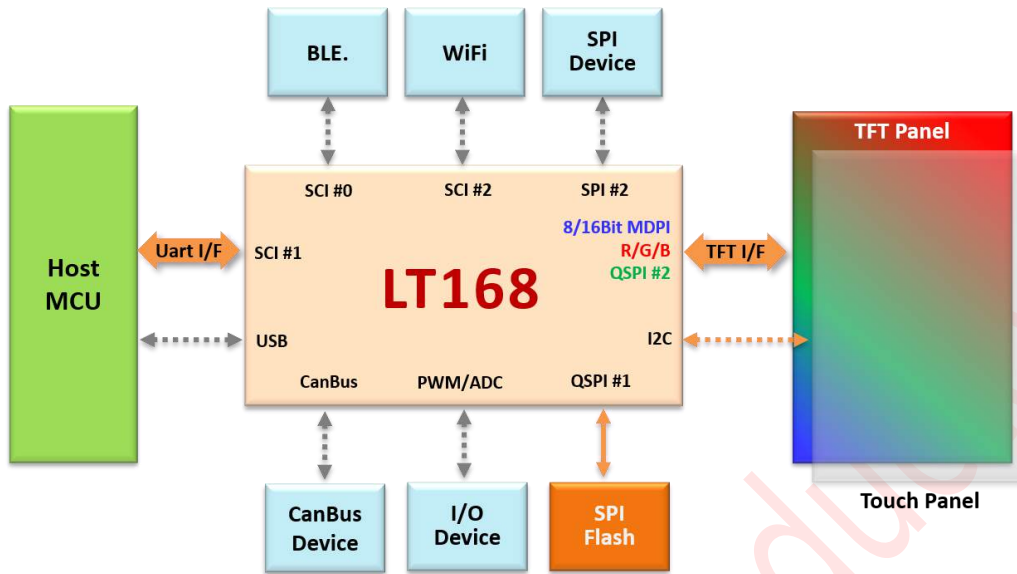
**1.3.23. Internal Oscillator**

- 128KHz on-chip oscillator clock for watchdog and PMU
- PLL clock which can be used for system clock
- 48MHz USB PLL clock which can be used for USB SIE

**1.3.24. External Crystal Oscillator**

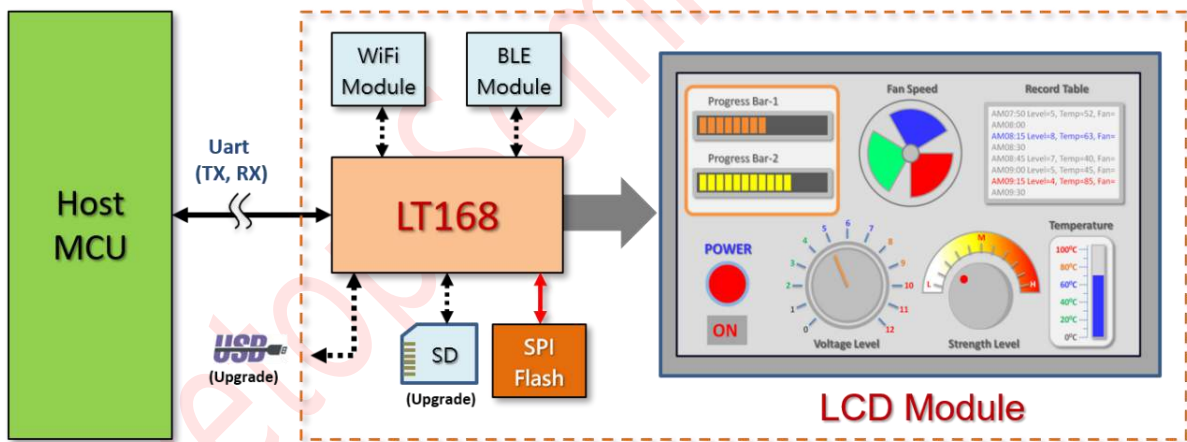
- Up to 20Mhz external crystal Oscillator clock which can be used for system clock
- 32.768Khz external crystal Oscillator clock which can be used for RTC

**1.4. System Block**



**Figure 1-3: System Block of LT168**

**1.5. Application Diagram**



**Figure 1-4: Application Diagram of TFT LCD Module**

## 2. Signal Description

### 2.1. Introduction

LT168 is available in the following packages:

- QFN48 (6\*6 mm<sup>2</sup>) – LT168A
- QFN68 (8\*8 mm<sup>2</sup>) – LT168B

### 2.2. Pin Assignment

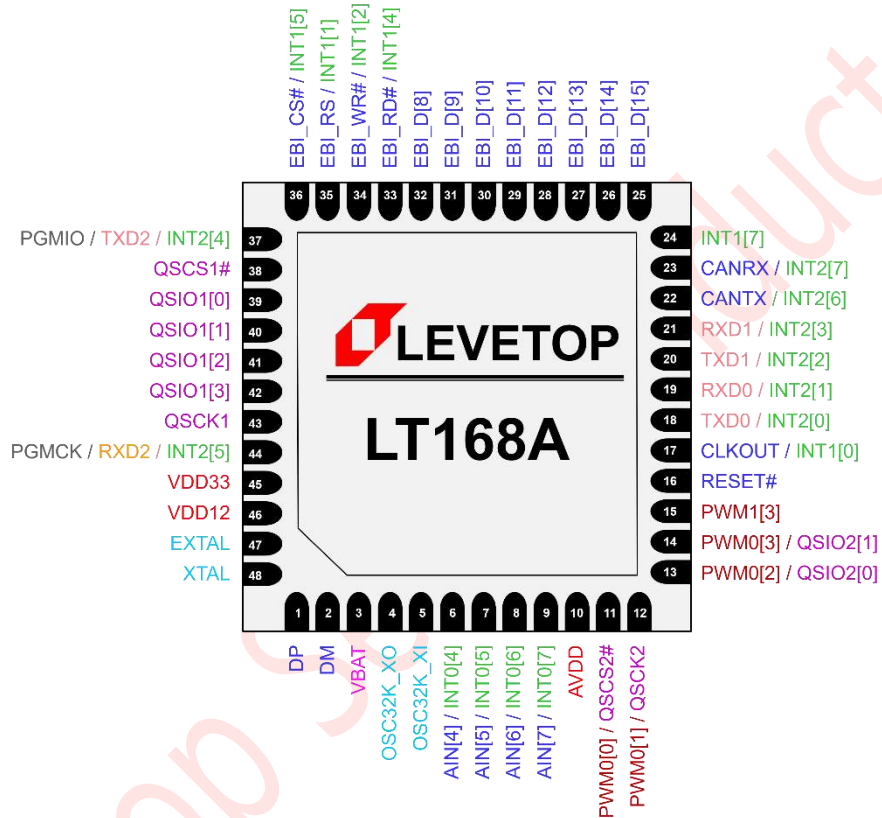
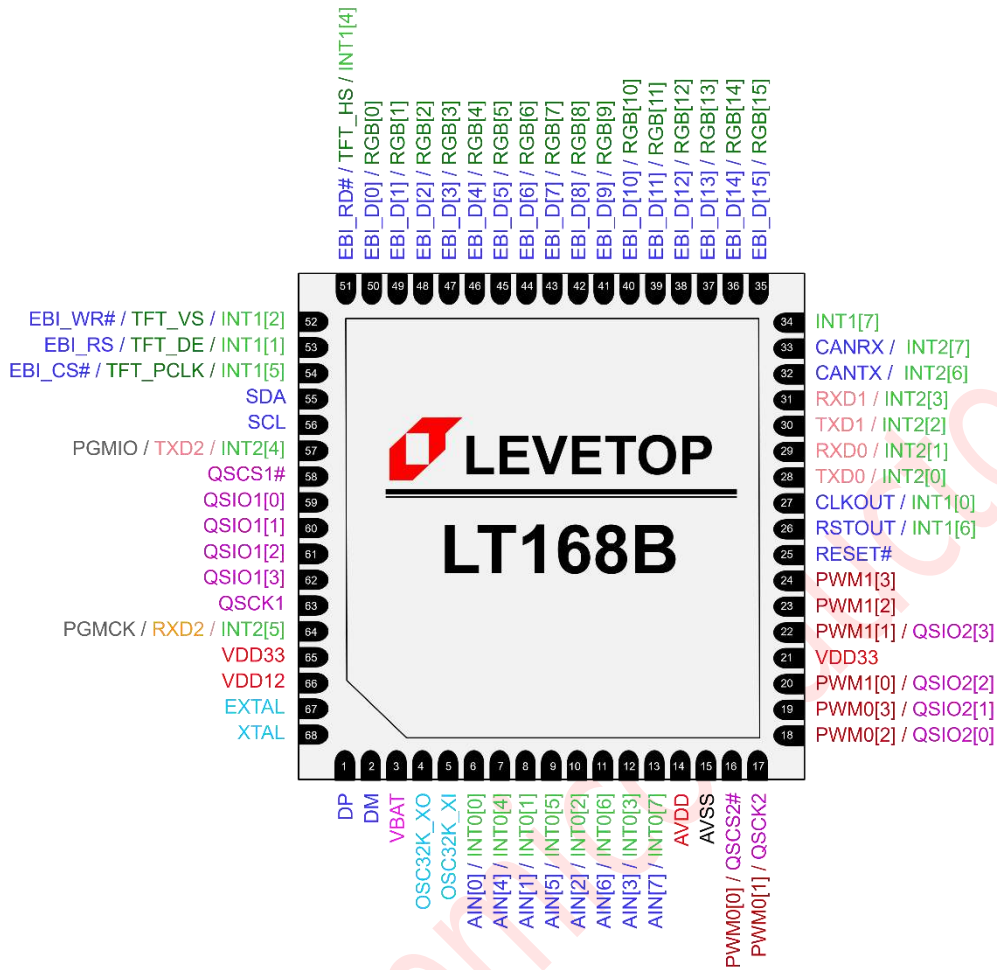


Figure 2-1: LT168A (QFN48) Pin Assignment



**Figure 2-2: LT168B (QFN68) Pin Assignment**

## 2.3. Signal Properties Summary

**Table 2-1: Signal Properties**

Name	Alternate 0	Alternate 1	Pin #		Qty.	Dir.	Default Dir *2	Pullup *3	IO Type *4
			LT168A	LT168B					
<b>SCI (6)</b>									
<b>RXD0</b> *1	INT2[1]	-	19	29	1	I/O	I	PullUp	PBCUL16R
<b>TXD0</b>	INT2[0]	-	18	28	1	I/O	O(H)	-	PBCUL16R
<b>RXD1</b>	INT2[3]	-	21	31	1	I/O	I	PullUp	PBCUL16R
<b>TXD1</b>	INT2[2]	-	20	30	1	I/O	O(H)	-	PBCUL16R
RXD2	<b>PGMCK</b>	INT2[5]	44	64	1	I/O	I	PullUp	PBCUL16R
TXD2	<b>PGMIO</b>	INT2[4]	37	57	1	I/O	I	PullUp	PBCUL16R
<b>USB (2)</b>									
DP	-	-	1	1	1	Analog	Hiz	-	-
DM	-	-	2	2	1	Analog	Hiz	-	-
<b>I2C (2)</b>									
<b>SCL</b>	GPIO_SCL	-	-	56	1	I/O	I	PullUp	PBCUL16R
<b>SDA</b>	GPIO_SDA	-	-	55	1	I/O	I	PullUp	PBCUL16R
<b>QSPI (18)</b>									
QSIO2[3]	<b>PWM1[1]</b>	-	-	22	1	I/O	I	PullDown	PBCD24R
QSIO2[2]	<b>PWM1[0]</b>	-	-	20	1	I/O	I	PullDown	PBCD24R
QSIO2[1]	<b>PWM0[3]</b>	-	14	19	1	I/O	I	PullDown	PBCD24R
QSIO2[0]	<b>PWM0[2]</b>	-	13	18	1	I/O	I	PullDown	PBCD24R
QSCK2	<b>PWM0[1]</b>	-	12	17	1	I/O	I	PullDown	PBCD24R
QSCS2#	<b>PWM0[0]</b>	-	11	16	1	I/O	I	PullDown	PBCD24R
QSIO1[3]	-	-	42	62	1	I/O	I	PullUp	PBCU24R
QSIO1[2]	-	-	41	61	1	I/O	I	PullUp	PBCU24R
QSIO1[1]	-	-	40	60	1	I/O	I	PullUp	PBCU24R
QSIO1[0]	-	-	39	59	1	I/O	I	PullUp	PBCU24R
QSCK1	-	-	43	63	1	O	O(L)	-	PBCU24R
QSCS1#	-	-	38	58	1	O	O(H)	-	PBCU24R
QSIO0[3] *5	-	-	-	-	1	I/O	I	PullUp	PBCU24R
QSIO0[2]	-	-	-	-	1	I/O	I	PullUp	PBCU24R
QSIO0[1]	-	-	-	-	1	I/O	I	PullUp	PBCU24R
QSIO0[0]	-	-	-	-	1	I/O	I	PullUp	PBCU24R
QSCK0	-	-	-	-	1	O	O(L)	-	PBCU24R
QSCS0#	-	-	-	-	1	O	O(H)	-	PBCU24R
<b>PWM0 (4)</b>									
<b>PWM0[3]</b>	QSIO2[1]	-	14	19	1	I/O	I	PullDown	PBCD24R
<b>PWM0[2]</b>	QSIO2[0]	-	13	18	1	I/O	I	PullDown	PBCD24R



Name	Alternate 0	Alternate 1	Pin #		Qty.	Dir.	Default Dir *2	Pullup *3	IO Type *4
			LT168A	LT168B					
PWM0[1]	QSCK2	-	12	17	1	I/O	I	PullDown	PBCD24R
PWM0[0]	QSCS2#	-	11	16	1	I/O	I	PullDown	PBCD24R
<b>PWM1 (4)</b>									
PWM1[3]	-	-	15	24	1	I/O	I	PullDown	PBCD24R
PWM1[2]	-	-	-	23	1	I/O	I	PullDown	PBCD24R
PWM1[1]	QSIO2[3]	-	-	22	1	I/O	I	PullDown	PBCD24R
PWM1[0]	QSIO2[2]	-	-	20	1	I/O	I	PullDown	PBCD24R
<b>ADC (8)</b>									
AIN[7]	INT0[7]	-	9	13	1	Analog	Hiz	-	PVDD1ANPR
AIN[6]	INT0[6]	-	8	11	1	Analog	Hiz	-	PVDD1ANPR
AIN[5]	INT0[5]	-	7	9	1	Analog	Hiz	-	PVDD1ANPR
AIN[4]	INT0[4]	-	6	7	1	Analog	Hiz	-	PVDD1ANPR
AIN[3]	INT0[3]	-	-	12	1	Analog	Hiz	-	PVDD1ANPR
AIN[2]	INT0[2]	-	-	10	1	Analog	Hiz	-	PVDD1ANPR
AIN[1]	INT0[1]	-	-	8	1	Analog	Hiz	-	PVDD1ANPR
AIN[0]	INT0[0]	-	-	6	1	Analog	Hiz	-	PVDD1ANPR
<b>Edge Port 0 (8)</b>									
INT0[7]	AIN[7]	-	9	13	1	I/O	Hiz	-	PBCUL16R
INT0[6]	AIN[6]	-	8	11	1	I/O	Hiz	-	PBCUL16R
INT0[5]	AIN[5]	-	7	9	1	I/O	Hiz	-	PBCUL16R
INT0[4]	AIN[4]	-	6	7	1	I/O	Hiz	-	PBCUL16R
INT0[3]	AIN[3]	-	-	12	1	I/O	Hiz	-	PBCUL16R
INT0[2]	AIN[2]	-	-	10	1	I/O	Hiz	-	PBCUL16R
INT0[1]	AIN[1]	-	-	8	1	I/O	Hiz	-	PBCUL16R
INT0[0]	AIN[0]	-	-	6	1	I/O	Hiz	-	PBCUL16R
<b>Edge Port 1 (7)</b>									
INT1[7]	-	-	24	34	1	I/O	I	PullUp	PBCUL16R
INT1[6]	RSTOUT	-	-	26	1	I/O	O	-	PBCUL16R
INT1[5]	EBI_CS#	TFT_PCLK	36	54	1	I/O	O(H)	-	PBCU24R
INT1[4]	EBI_RD#	TFT_HS	33	51	1	I/O	O(H)	-	PBCU24R
INT1[2]	EBI_WR#	TFT_VS	34	52	1	I/O	O(H)	-	PBCU24R
INT1[1]	EBI_RS	TFT_DE	35	53	1	I/O	O(L)	-	PBCU24R
INT1[0]	CLKOUT	-	17	27	1	I/O	O	-	PBCUL16R
<b>Edge Port 2 (8)</b>									
INT2[7]	CANRX	-	23	33	1	I/O	I	PullUp	PBCUL16R
INT2[6]	CANTX	-	22	32	1	I/O	O(H)	-	PBCUL16R

Name	Alternate 0	Alternate 1	Pin #		Qty.	Dir.	Default Dir *2	Pullup *3	IO Type *4
			LT168A	LT168B					
INT2[5]	RXD2	PGMCK	44	64	1	I/O	I	PullUp	PBCUL16R
INT2[4]	TXD2	PGMIO	37	57	1	I/O	I	PullUp	PBCUL16R
INT2[3]	RXD1	-	21	31	1	I/O	I	PullUp	PBCUL16R
INT2[2]	TXD1	-	20	30	1	I/O	O(H)	-	PBCUL16R
INT2[1]	RXD0	-	19	29	1	I/O	I	PullUp	PBCUL16R
INT2[0]	TXD0	-	18	28	1	I/O	O(H)	-	PBCUL16R
<b>Programming Port (2)</b>									
PGMCK	RXD2	INT2[5]	44	64	1	I/O	I	PullUp	PBCUL16R
PGMIO	TXD2	INT2[4]	37	57	1	I/O	I	PullUp	PBCUL16R
<b>CLOCK (5)</b>									
EXTAL	-	-	47	67	1	I/O	I	-	PXWE2R
XTAL	-	-	48	68	1	I/O	O	-	PXWE2R
OSC32K_XI	-	-	5	5	1	I/O	I	-	PVDD1ANPR
OSC32K_XO	-	-	4	4	1	I/O	O	-	PVDD1ANPR
CLKOUT	INT1[0]	-	17	27	1	I/O	O	-	PBCUL16R
<b>RESET (2)</b>									
RESET#	-	-	16	25	1	-	I	PullUp	PISUR
RSTOUT	INT1[6]	-	-	26	1	I/O	O	-	PBCUL16R
<b>CAN (2)</b>									
CANRX	INT2[7]	-	23	33	1	I/O	I	PullUp	PBCUL16R
CANTX	INT2[6]	-	22	32	1	I/O	O(H)	-	PBCUL16R
<b>EBI (20)</b>									
EBI_CS# *6	TFT_PCLK	INT1[5]	36	54	1	I/O	O(H)	-	PBCU24R
EBI_RS	TFT_DE	INT1[1]	35	53	1	I/O	O(L)	-	PBCU24R
EBI_RD#	TFT_HS	INT1[4]	33	51	1	I/O	O(H)	-	PBCU24R
EBI_WR#	TFT_VS	INT1[2]	34	52	1	I/O	O(H)	-	PBCU24R
EBI_D[15]	TFT_RGB[15]	GPIO_D[15]	25	35	1	I/O	I	PullUp	PBCU24R
EBI_D[14]	TFT_RGB[14]	GPIO_D[14]	26	36	1	I/O	I	PullUp	PBCU24R
EBI_D[13]	TFT_RGB[13]	GPIO_D[13]	27	37	1	I/O	I	PullUp	PBCU24R
EBI_D[12]	TFT_RGB[12]	GPIO_D[12]	28	38	1	I/O	I	PullUp	PBCU24R
EBI_D[11]	TFT_RGB[11]	GPIO_D[11]	29	39	1	I/O	I	PullUp	PBCU24R
EBI_D[10]	TFT_RGB[10]	GPIO_D[10]	30	40	1	I/O	I	PullUp	PBCU24R
EBI_D[9]	TFT_RGB[9]	GPIO_D[9]	31	41	1	I/O	I	PullUp	PBCU24R
EBI_D[8]	TFT_RGB[8]	GPIO_D[8]	32	42	1	I/O	I	PullUp	PBCU24R
EBI_D[7]	TFT_RGB[7]	GPIO_D[7]	-	43	1	I/O	I	PullUp	PBCU24R
EBI_D[6]	TFT_RGB[6]	GPIO_D[6]	-	44	1	I/O	I	PullUp	PBCU24R
EBI_D[5]	TFT_RGB[5]	GPIO_D[5]	-	45	1	I/O	I	PullUp	PBCU24R

Name	Alternate 0	Alternate 1	Pin #		Qty.	Dir.	Default Dir *2	Pullup *3	IO Type *4
			LT168A	LT168B					
EBI_D[4]	TFT_RGB[4]	GPIO_D[4]	-	46	1	I/O	I	PullUp	PBCU24R
EBI_D[3]	TFT_RGB[3]	GPIO_D[3]	-	47	1	I/O	I	PullUp	PBCU24R
EBI_D[2]	TFT_RGB[2]	GPIO_D[2]	-	48	1	I/O	I	PullUp	PBCU24R
EBI_D[1]	TFT_RGB[1]	GPIO_D[1]	-	49	1	I/O	I	PullUp	PBCU24R
EBI_D[0]	TFT_RGB[0]	GPIO_D[0]	-	50	1	I/O	I	PullUp	PBCU24R
<b>Power Supply</b>									
VDD33	-	-	45	21, 65	2	P	-	-	PVDD2R
VDD12	-	-	46	66	1	P	-	-	PWR
AVDD	-	-	10	14	1	P	-	-	PVDD1ANPR
VBAT	-	-	3	3	1	P	-	-	PWR
VSS	-	-	49*7	69*7	1	G	-	-	PWR
AVSS	-	-	-	15	1	G	-	-	PWR

### Notes:

- The **Red** signal name is the default signal for the multi-purpose pin.
- "Default Dir" is referred to direction after Reset. "I" stands for Input, "O" stands for Output, "O(H)" stands for Output High, "O(L)" stands for Output Low, and "Hiz" means Input and Output are all disabled and pullup/pulldown is also dis-abled.
- All pullups and pulldowns are disconnected when the signal is programmed as an output.
- Output Driver Type:
  - PBCU24R = CMOS Tri-state output pad with controllable input and controllable pull-up,
  - PBCUL16R = CMOS Tri-state output pad with controllable input and controllable pullup and limited slew rate,
  - PBCD24R = CMOS Tri-state output pad with enable controlled input and enable controlled pull down,
  - PISUR = schmitt trigger enable controlled input pad with pull-up,
  - PVDD1ANPR = VDD analog PAD with digital power domain,
  - PXWE2R = Crystal oscillator with internal resistor and active high enable,
  - PVDD2R = VDD power pad for I/O post driver,
  - The suffix of cell means the drive strength and x can be 2, 16 and 24. For example, PBCU24R means the drive strength is 24mA.
- These QSPI0 signals are connected directly to the embedded SPI Flash memory.
- The control signals of the "EBI Bus" are dedicated hardware interface and cannot be replaced by other GPIO pins.
- This is the back thermal pad of the LT168x. It must connect to ground (VSS/GND) directly.

## 2.4. Signal Descriptions

This section provides a brief description of the signals. For more detailed information, refer to the specific module section.

**Table 2-2: Signal Description**

Pin Name	Pin Number		Pin Description
	LT168A	LT168B	
<b>Serial Communications Interface 0 Module Signals (SCI0)</b>			
<b>RXD0</b>	19	29	<b>Receive Data</b> This signal can be configured as the SCI0 receiver data input, or be configured as GPIO (INT2[1]).
<b>TXD0</b>	18	28	<b>Transmit Data</b> This signal can be configured as the SCI0 transmitter data output, or be configured as GPIO (INT2[0]).
<b>Serial Communications Interface 1 Module Signals (SCI1)</b>			
<b>RXD1</b>	21	31	<b>Receive Data</b> This signal can be configured as SCI1 receiver data input, or be configured as GPIO (INT2[3]) .
<b>TXD1</b>	20	30	<b>Transmit Data</b> This signal can be configured as SCI1 transmitter data output, or be configured as GPIO (INT2[2]) .
<b>Serial Communications Interface 2 Module Signals (SCI2)</b>			
<b>RXD2</b>	44	64	<b>Receive Data</b> This signal can be configured as SCI2 receiver data input, GPIO (INT2[5]) or PGMCK
<b>TXD2</b>	37	57	<b>Transmit Data</b> This signal can be configured as SCI2 transmitter data output, GPIO (INT2[4]) or PGMIO.
<b>CAN Module Signals</b>			
<b>CANRX</b>	23	33	<b>Receive Data</b> This pin is the receive pin from the CANBus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'. This signal can also be used as GPIO (INT2[7]) when not configured for Canbus operation.
<b>CANTX</b>	22	32	<b>Transmit Data</b> This pin is the transmit pin to CANBus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'. This signal can also be used as GPIO (INT2[6]) when not configured for Canbus operation.

Pin Name	Pin Number		Pin Description
	LT168A	LT168B	
<b>RGB Controller Module Signals</b>			
These signals are available for LT168B only, and are used to drive RGB type TFT panel.			
TFT_PCLK	-	54	<b>TFT Pixel Clock</b> This is pixel clock used to drive the display panel.
TFT_HS	-	51	<b>TFT Horizontal Synchronous Signal</b> This is horizontal synchronous signal, indicating the beginning of a new line.
TFT_VS	-	52	<b>TFT Vertical Synchronous Signal</b> This is vertical synchronous signal, indicating the beginning of a new frame.
TFT_DE	-	53	<b>TFT Data Enable</b> This is data enable signal for RGB interface operation.
TFT_RGB[15:0]	-	35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50	<b>TFT RGB Signals</b> This is Red, Green and Blue data output.
<b>External Bus Interface (EBI) Signals</b>			
The External Bus Interface (EBI) is responsible for controlling the transfer of the information between the internal bus and the external 8080 Parallel MCU Panel. LT168A supports 8-bits EBI, and LT168B supports 8-bits or 16-bits EBI.			
EBI_CS#	36	54	<b>EBI Chip Select</b> LT168 provides External Bus Interface (EBI) to drive MCU type display panel. This signal is the chip select of External Bus Interface (EBI).
EBI_RS	35	53	<b>EBI Register Select</b> This signal is the register select of EBI.
EBI_RD#	33	51	<b>EBI Data Read Control</b> This signal is the data read control signal of EBI.
EBI_WR#	34	52	<b>EBI Data Write Control</b> This signal is the data write control signal of EBI.
EBI_D[15:8]	25, 26, 27, 28, 29, 30, 31, 32	35, 36, 37, 38, 39, 40, 41, 42	<b>EBI High Byte Data Bus</b> These data signals are drive to MCU type display panel.
EBI_D[7:0]	-	43, 44, 45, 46, 47, 48, 49, 50	<b>EBI Low Byte Data Bus</b> These data signals are available for LT168B only, and are used to drive MCU type display panel.
<b>Universal Serial Bus Module Signals (USB)</b>			
DP	1	1	<b>USB Data Positive</b> This signal is used by the USB module.
DM	2	2	<b>USB Data Negative</b> This signal is used by the USB module.

Pin Name	Pin Number		Pin Description
	LT168A	LT168B	
<b>I2C Module Signals</b>			
<b>SCL</b>	-	56	<b>I2C Clock</b> This signal can be configured as the I2C clock line signal, or be configured as GPIO.
<b>SDA</b>	-	55	<b>I2C Data</b> This signal can be configured as the I2C data line signal, or be configured as GPIO.
<b>Quad Serial Peripheral Interface Module 0 (QSPI0)</b> The QSPI0 are used for embedded Flash. These signals are connected to internal QSPI Flash's pins directly.			
<b>QSIO0[3:0]</b>	-	-	<b>QSPI0 Master Input/Out</b> These signals are the serial data output or input from the QSPI0 in master mode.
<b>QSCS0#</b>	-	-	<b>QSPI0 Master Chip Select Output</b> This signal is the chip select signal from the QSPI0 in master mode and low active.
<b>QSCK0</b>	-	-	<b>QSPI0 Master Serial Clock Output</b> This signal is the serial clock output from the QSPI0 in master mode
<b>Quad Serial Peripheral Interface Module 1 (QSPI1)</b>			
<b>QSIO1[3:0]</b>	42, 41, 40, 39	62, 61, 60, 59	<b>QSPI1 Master Input/Out</b> These signals are the serial data output or input from the QSPI1 in master mode.
<b>QSCS1#</b>	38	58	<b>QSPI1 Master Chip Select Output</b> This signal is the chip select signal from the QSPI1 in master mode and low active.
<b>QSCK1</b>	43	63	<b>QSPI1 Master Serial Clock Output</b> This signal is the serial clock output from the QSPI1 in master mode
<b>Quad Serial Peripheral Interface Module 2 (QSPI2)</b>			
<b>QSIO2[3:0]</b>	13, 12, 11, -	22, 20, 19, 18	<b>QSPI2 Master Input/Out</b> These signals are the serial data output or input from the QSPI2 in master mode. These signals can also be configured as PWM1[1], PWM1[0], PWM0[3], PWM0[2] when not configured as QSPI2.
<b>QSCS2#</b>	-	16	<b>QSPI2 Master Chip Select Output</b> This signal is the chip select signal from the QSPI2 in master mode and low active. This signal can also be configured as PWM0[0] when not configured as QSPI2.

Pin Name	Pin Number		Pin Description
	LT168A	LT168B	
QSCK2	-	17	<b>QSPI2 Master Serial Clock Output</b> This signal is the serial clock output from the QSPI2 in master mode. This signal can also be configured as PWM0[1] when not configured as QSPI2.
<b>Edge Port 0 Signals</b>			
INT0[7:0]	9, 8, 7, 6, -, -, -, -	13, 11, 9, 7, 12, 10, 8, 6	<b>Interrupt Input</b> These bidirectional signals can be configured as either external interrupt sources or GPIO. These signals can also be configured as AIN[7:0] when not configured as interrupt sources or GPIO.
<b>Edge Port 1 Signals</b>			
INT1[7:0]	24, -, 36, 33, -, 34, 35, 17	34, 26, 54, 51, -, 52, 53, 27	<b>Interrupt Input</b> These bidirectional signals can be configured as either external interrupt sources or GPIO.
<b>Pulse Width Modulator 0 Signals</b>			
PWM0[3:0]	11, -, -, -	19, 18, 17, 16	These signals can be configured as either PWM 0 output or GPIO.
<b>Pulse Width Modulator 1 Signals</b>			
PWM1[3:0]	15, 14, 13, 12	24, 23, 22, 20	These signals can be configured as either PWM 1 output or GPIO.
<b>Analog-to-Digital Converter Signals</b>			
AIN[7:0]	9, 8, 7, 6, -, -, -, -	13, 11, 9, 7, 12, 10, 8, 6	<b>Analog Input</b> These analog signals can be configured as ADC analog channels. These signals can also be configured as INT0[7:0] when not configured as analog input.
<b>Programming Debug Signals</b>			
PGMCK	44	64	<b>Test Clock</b> This input signal is the test clock used to synchronize the Programming debug logic.
PGMIO	37	57	<b>Test Data Input/Output</b> This input/output signal is the serial input/output for testing instructions and data.
<b>Clock Signals</b>			
EXTAL	47	67	<b>Fast Oscillator Pad Input</b> The signal is the input of fast Oscillator Pad.
XTAL	48	68	<b>Fast Oscillator Pad Output</b> The signal is the output fast Oscillator pad.
OSC32K_XI	5	5	<b>32.768Khz Oscillator Pad Input</b> The signal is the input of 32.768Khz Oscillator pad.
OSC32K_XO	4	4	<b>32.768Khz Oscillator Pad Output</b> The signal is the output 32.768Khz Oscillator pad.

Pin Name	Pin Number		Pin Description
	LT168A	LT168B	
<b>CLKOUT</b>	17	27	<b>Clock Out</b> This output signal reflects the internal system clock. This signal can also be configured as INT1[0] when not configured as Clock output.
<b>Reset Signals</b>			
<b>RESET#</b>	16	25	<b>Reset In</b> This active-low input signal is used as the external reset request. The reset signal will place the CPU in supervisor mode with default settings for all register bits except for the register bits that can only be reset by POR. 0 = External Reset Assert 1 = External Reset Desert
<b>RSTOUT</b>	-	26	<b>Reset Out</b> This active-low output signal indicates that the internal reset controller has reset the chip. 0 = Chip is at reset status 1 = Chip is not at reset status This signal can also be configured as INT1[6] when not configured as Reset output.
<b>Power and Ground Signals</b>			
These signals provide system power and ground to the chip. Multiple signals are provided for adequate current capability. All power supply signals must have adequate bypass capacitance for high-frequency noise suppression.			
<b>VDD33</b>	45	21, 65	<b>3.3V Power</b> This signal supplies 3.3V positive power to the I/O pads and LDO.
<b>VDD12</b>	46	66	<b>1.2V LDO Output</b> This 1.2V LDO output signal is used to supply the power of the core logic. 1uF ceramic bypass capacitor is required to externally connect between the pad and VSS.
<b>AVDD</b>	10	14	This signal supplies 3.3V positive power to Analog module.
<b>VBAT</b>	3	3	This signal supplies battery power to RTC module.
<b>VSS</b>	49 <sup>(1)</sup>	69 <sup>(1)</sup>	This signal supplies 3.3V negative supply (ground) to the I/O pads and LDO.
<b>AVSS</b>	-	15	This signal supplies 3.3V negative supply (ground) to the Analog Module.

**Note (1)** : This is the thermal pad and must be grounded directly to VSS or GND.



**2.5. LT168A vs. LT168B**
**Table 2-3: LT168x Comparison**

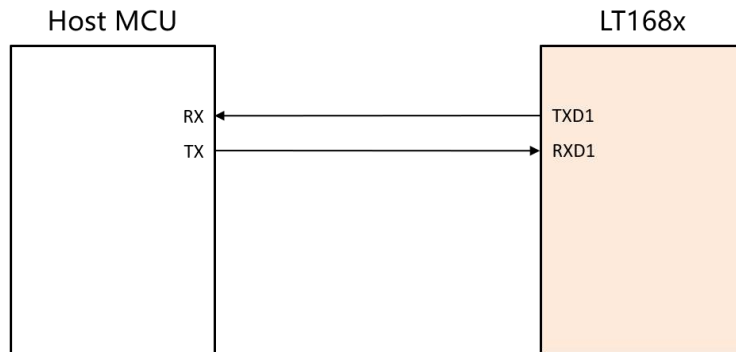
Function		LT168A	LT168B
Items	Description		
TFT LCD Panel	RGB Interface	--	V (480*480 max)
	16bit 8080 Interface	--	V (800*480 max)
	8bit 8080 Interface	V (480*320 max)	V (480*320 max)
	QSPI Panel	--	V
MCU Core & Memory	Core	32-bits RISC	32-bits RISC
	Speed	200MHz	200MHz
	Flash Capacity	512KB	2MB
	SRAM Capacity	256KB	256KB
	Display RAM <sup>(1)</sup>	512KB	512KB
	External PSRAM	--	V
Other Interface	Uart Port	V (x3)	V (x3)
	SPI I/F	V (x1)	--
	QSPI I/F	V (x1)	V (x2)
	USB 2.0	V	V
	SD Card	V	V
	PWM Output	V (x5)	V (x8)
	Can Bus	V (x1)	V (x1)
	ADC Input	V (x4)	V (x8)
	RTP I/F	V	V
	CTP I2C I/F	V	V
	GPIO	V (x10)	V (x19)
	RTC	V	V
	Appliaction and Upgrade	UI_Editor-II	V
UI_Emulator-II		V	V
USB Port Upgrade		V	V
Uart Port Upgrade		V	V
SD Card Upgrade		V	V
Support 2 <sup>nd</sup> Devetop		V	V
Power and Package	Power	3.3V	3.3V
	Package	QFN-48	QFN-68

**Note (1) :** The Display RAM can be used as the normal SRAM, which means the maximum normal SRAM is up to 768KB.

### 3. Hardware Interface

#### 3.1. Host Communication Interface

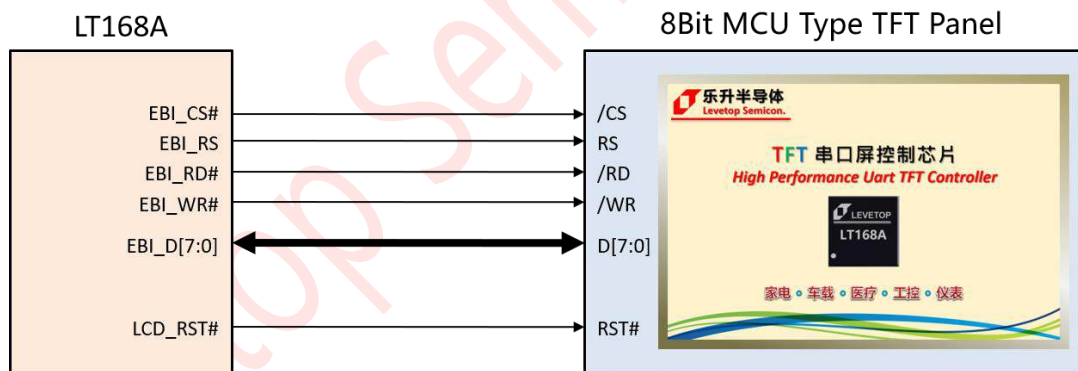
LT168 is designed to communicate with the Host MCU through UART interface. Please refer to UI\_Editor-II user manual for communication protocol and related settings.



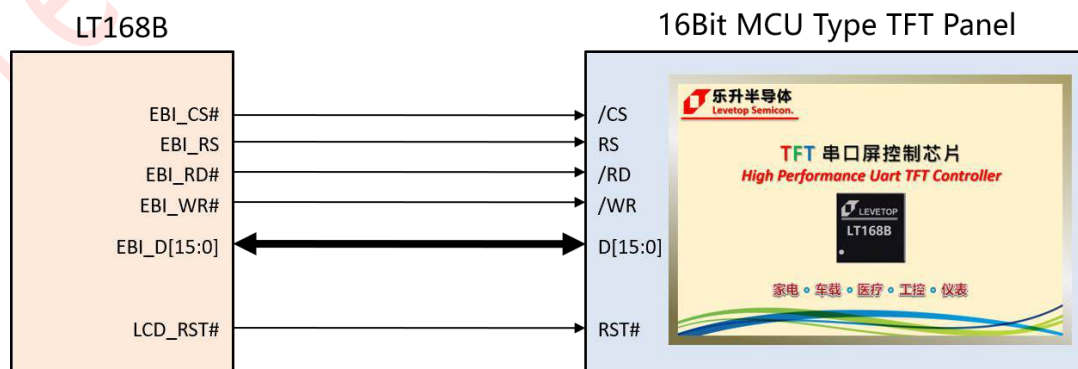
**Figure 3-1: UART Connection between LT168x and Host MCU**

#### 3.2. TFT LCD Panel Interface

LT168 provides an External Bus Interface (EBI) to drive TFT panels with parallel 8080 interface. The data bus of this parallel interface is either 8 bits or 16 bits. LT168B can drive both 8bits/16bits parallel MCU panels, whereas LT168A can drive 8bits parallel MCU panels only. The reference connections are as shown in Figure 3-2 and 3-3.



**Figure 3-2: LT168A Connect to 8-bits MCU Panel**



**Figure 3-3: LT168B Connect to 16-bits MCU Panel**

The LT168B is also available in RGB LCD interface mode. The reference schematic is shown in the figure below. The external PSRAM in the figure is reserved for display RAM data scratch area. If you want to achieve richer display effects such as image overlay, you can choose to add an external PSRAM chip. **Note:** LT168B supports RGB panels with maximum 480\*480 resolution.

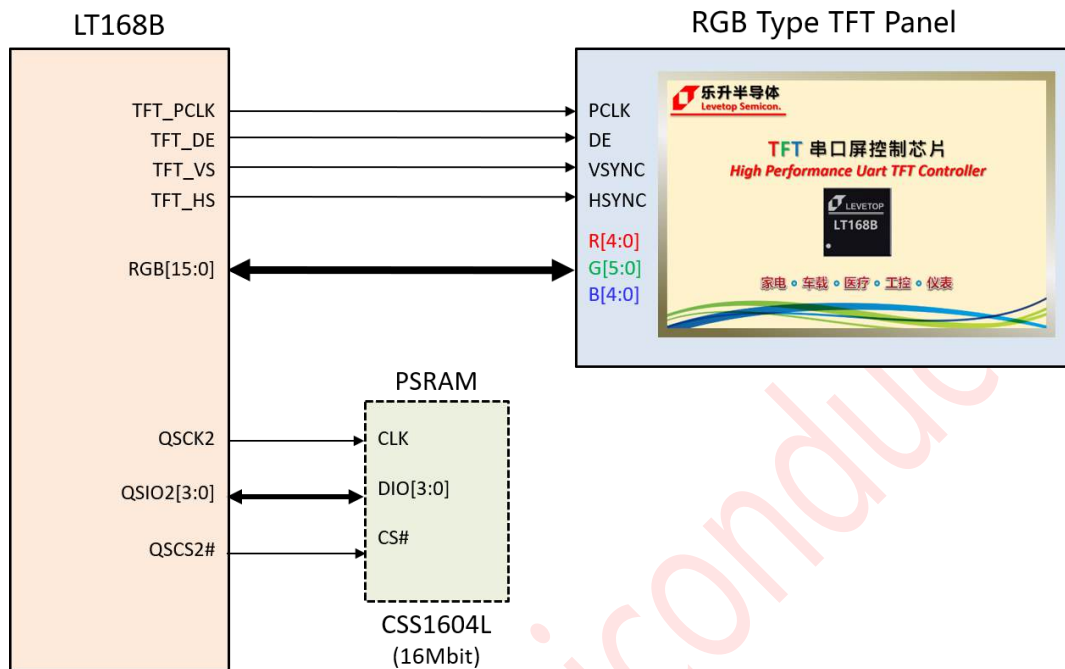
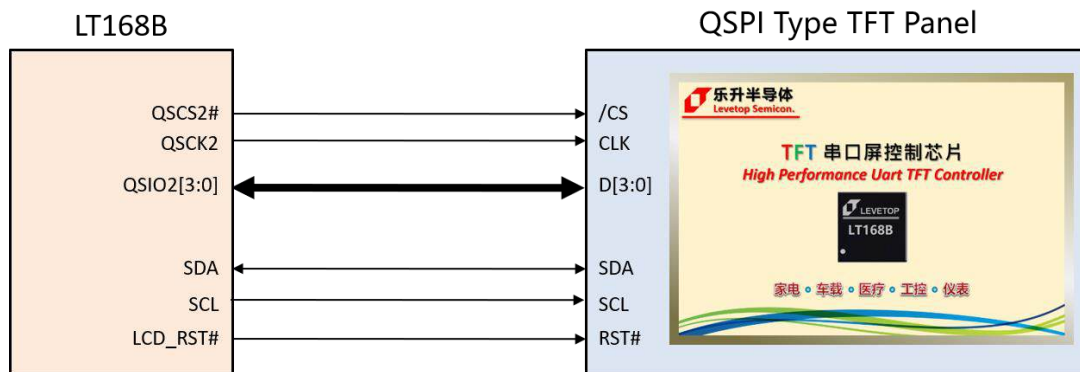


Figure 3-4: LT168B Connect to RGB Type TFT Panel

Table 3-1: The differences of LT168B for RGB Panel with or without PSRAM

Resolution	480 x 272		480 x 480	
	Without PSRAM	With PSRAM	Without PSRAM	With PSRAM
Functions	Without PSRAM	With PSRAM	Without PSRAM	With PSRAM
Page buffer refresh	Support	Support	Not Support	Support
The size of the limit display by PNG overlay	480 x 272	480 x 272	200 x 200	480 x 480

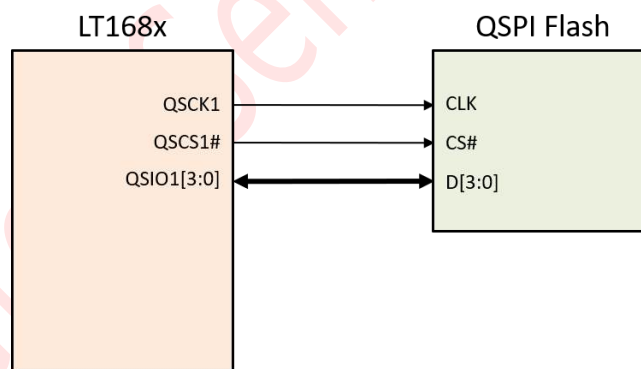
There is also a kind of TFT panel with QSPI interface on the TFT panel market. This kind of panel needs the display chip to provide high-speed QSPI signal and constantly refresh the display data in order to achieve good use effect. The LT168B also supports this kind of TFT display. The reference circuit diagram is as follows:



**Figure 3-5: LT168B Connect to QSPI Panel**

### 3.3. QSPI Flash Interface

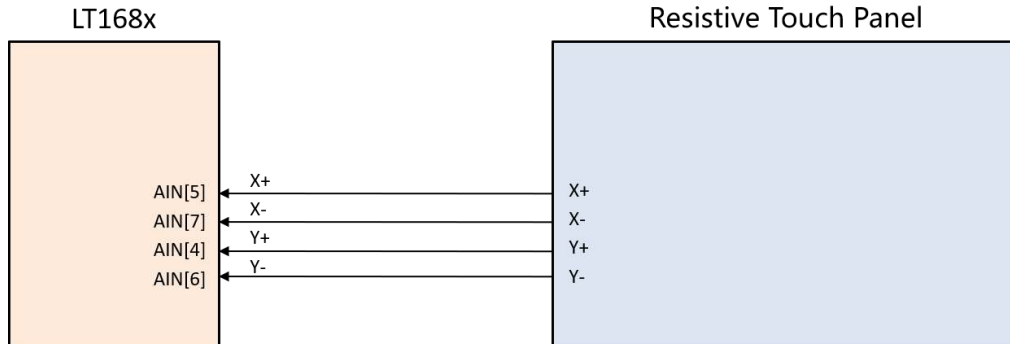
LT168 has two sets of QSPI interfaces, and one is for connecting to an external QSPI Flash. The external Flash is used to store images, animations, fonts and other information. Host MCU can send commands to request LT168 to retrieve the data from the QSPI Flash and transmit the data onto the connected TFT panel for display. The image data can be programmed to the external SPI Flash by several means. Please refer to UI\_Editor-II user manual for more detail. Another set of QSPI interface can be used to connect to other QSPI devices. See Section 3.12 SPI/Uart Interface for more detail.



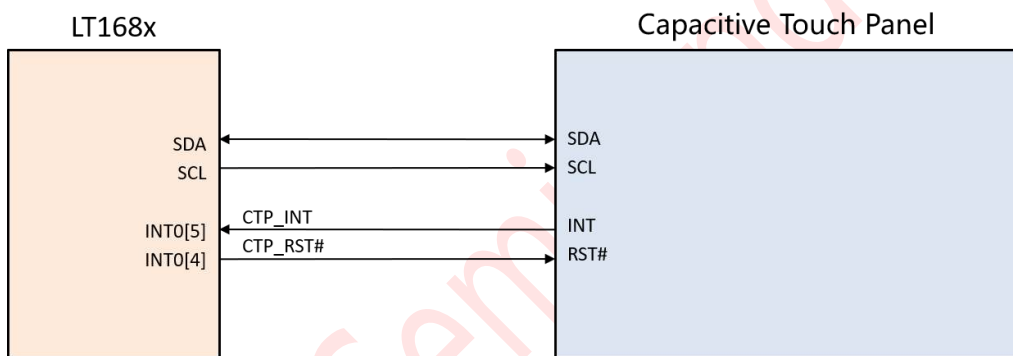
**Figure 3-6: LT168x Connect to QSPI Flash**

### 3.4. Touch Panel Interface

LT168 has an ADC analog input and I2C interface that can be used to interface directly to resistive or capacitive touch panel. Upon receiving the touch information, LT168 will process it and transmit it to the Host MCU. The reference schematics are as followings:



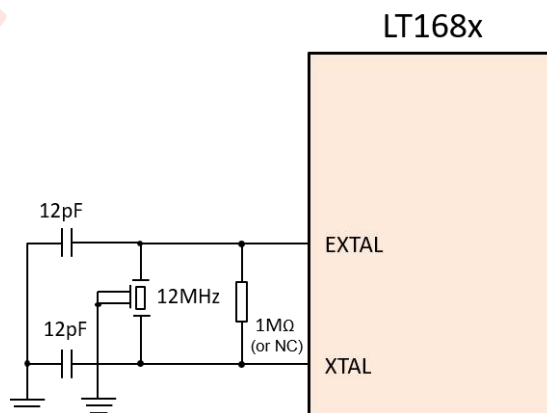
**Figure 3-7: LT168x Connect to Resistive Touch Panel**



**Figure 3-8: LT168x Connect to Capacitive Touch Panel**

### 3.5. Clock Interface

LT168x needs an external 12MHz Crystal Oscillator as the internal PLL and USB module clock source.



**Figure 3-9: External Clock Circuit**

### 3.6. Backlight Control Circuit

LT168 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 follows:

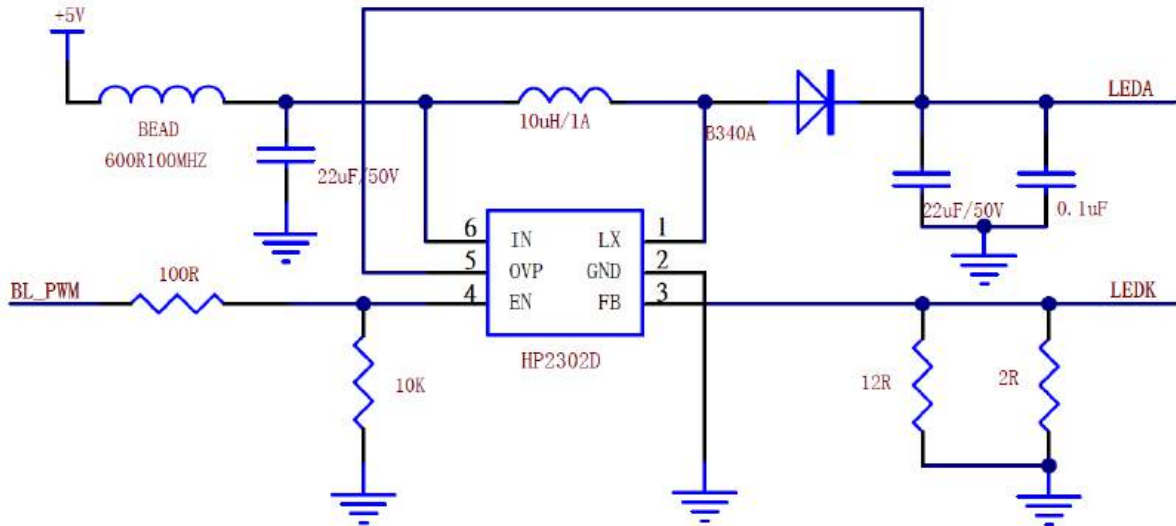


Figure 3-10: TFT LCD Backlight Circuit Example 1

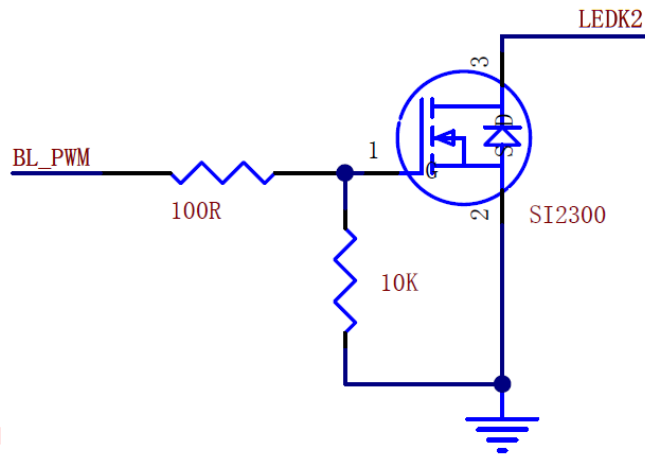
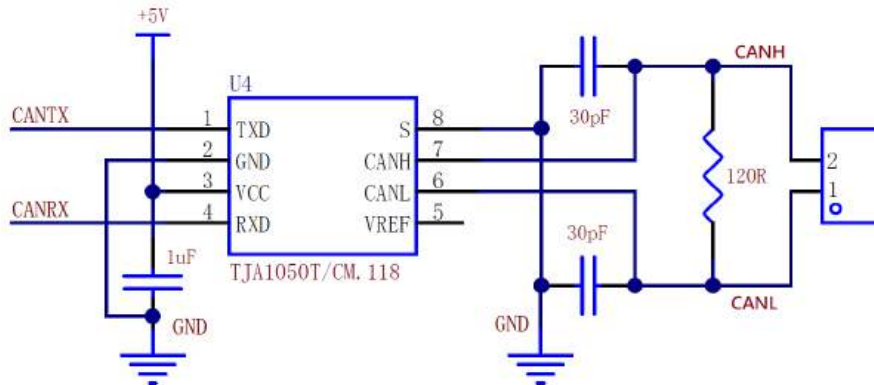


Figure 3-11: TFT LCD Backlight Circuit Example 2

### 3.7. Can Bus Interface

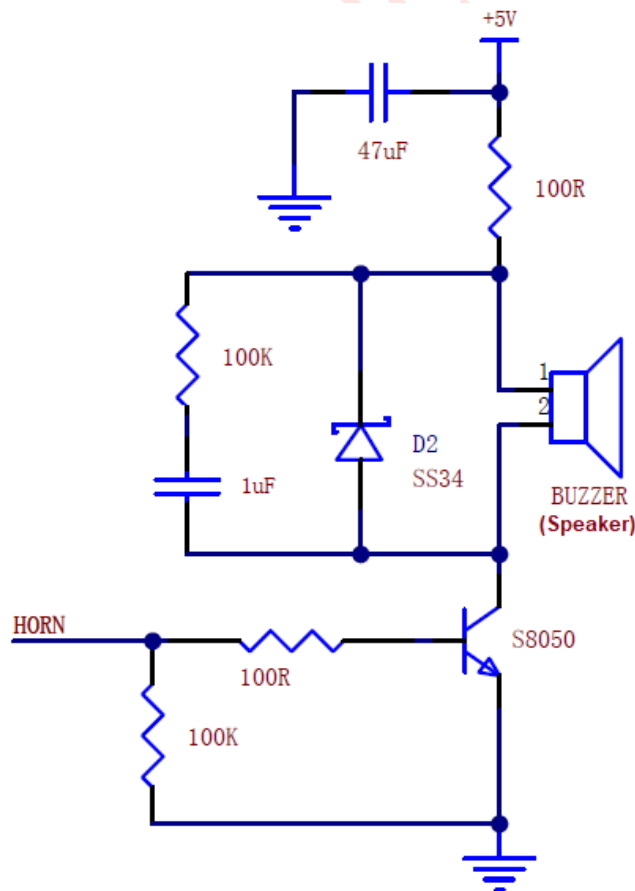
LT168 provides a CAN Bus interface. The reference schematic is as following:



**Figure 3-12: Canbus Circuit Example**

### 3.8. Audio Interface

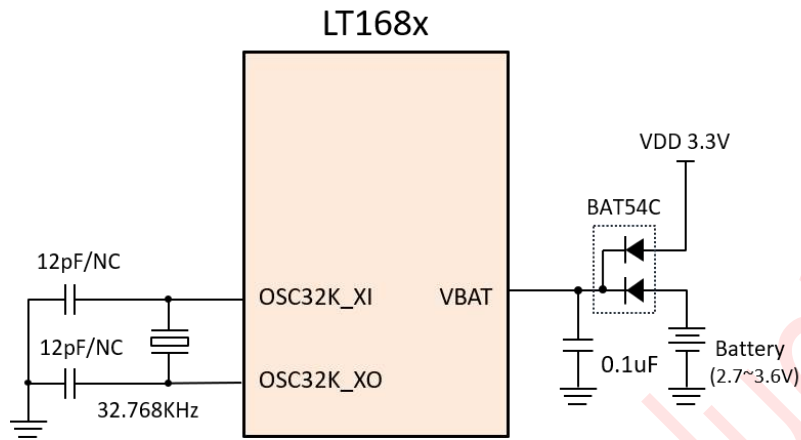
LT168 uses PWM1[2] to provide an analog sound output signal - "HORN", which can be used as a sound playback or to push a buzzer. The reference schematic is as following:



**Figure 3-13: Audio Output Application Circuit**

### 3.9. Real Time Clock Circuit

LT168 contains RTC (Real Time Clock) inside. To utilize the RTC, a 32.768KHz crystal circuit must be provided. The reference schematic is as following:

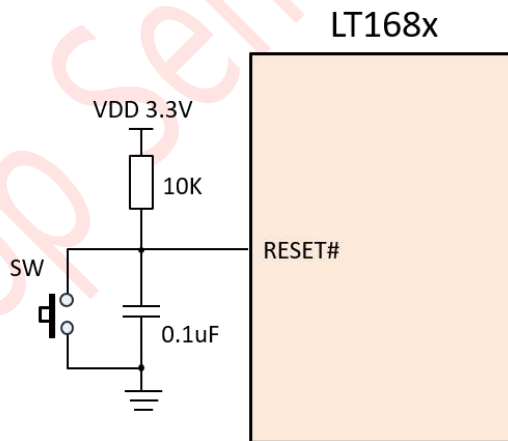


**Figure 3-14: RTC Application Circuit**

### 3.10. Reset Interface

There are two hardware reset sources for LT168, both of which are synchronized by the internal clock:

- Power on Reset
- External Reset Pin (RESET#)

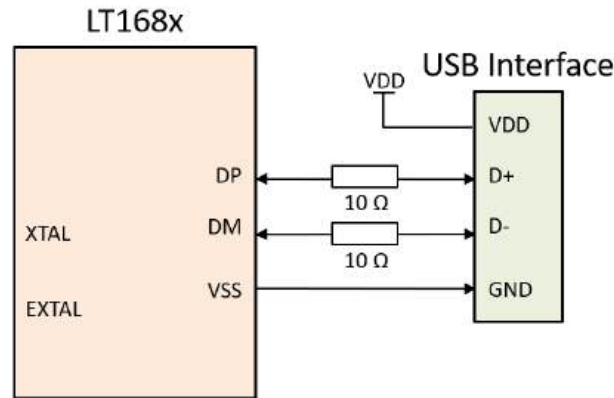


**Figure 3-15: External Reset Circuit**



### 3.11. USB Interface

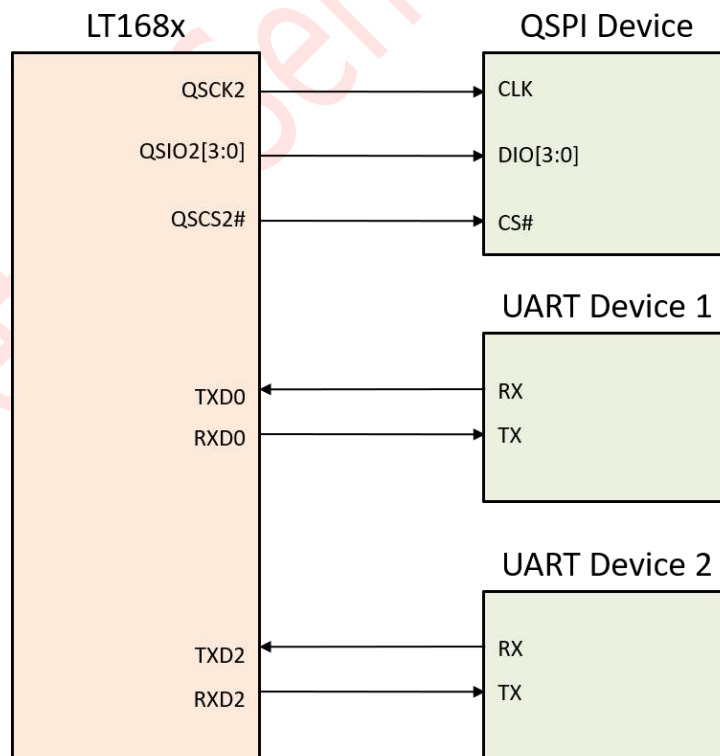
LT168 provides a USB interface with USB Slave functionality. This USB interface can be used to connect with a PC for updating the internal MCU program of LT168, and the data of the external SPI Flash. Please refer to UI\_Editor-II user manual for more detail.



**Figure 3-16: LT168x Connect to USB Connector**

### 3.12. SPI/Uart Interface

In addition to the necessary pins for the TFT panel, LT168x also provides some pins for external SPI/QSPI components. If QSPI2 is connected to QSPI PSRAM (Pseudo SRAM), it can be used to temporarily store display image materials or data, which is convenient for the MCU to perform image overlay and a large amount of data processing. The rest of the Uart interfaces for connecting to devices such as WiFi module, Bluetooth module, etc. The reference schematic is as following:



**Figure 3-17: Extension for SPI and Uart Device**

## 4. System Memory Map

### 4.1. Introduction

The embedded memory and address map of LT168 are listed below:

- Up to 128M Bytes of External SPI/QSPI Flash Memory
- 512K Bytes or 2M Bytes of Internal QSPI Flash Memory
- 8K Bytes of Internal Boot ROM Memory
- 768K Bytes of Internal Static Random-Access Memory (SRAM)
  - System RAM: the Upper 256K Bytes and Address Range from 0x00800000
  - Display RAM: the Lower 512K Bytes and Address Range from 0x00840000
- Internal Memory Mapped Registers

### 4.2. Address Map

0xFFFF_FFFF	Core Registers
0xE000_0000	
0x8FFF_FFFF	QSPI2 Flash
0x8000_0000	
0x7FFF_FFFF	QSPI1 Flash
0x7000_0000	
0x6FFF_FFFF	QSPI0 Flash
0x6000_0000	
0x40FF_FFFF	Registers
0x4000_0000	
0x2FFF_FFFF	EBI (8080)
0x2000_0000	
0x008B_FFFF	Internal SRAM
0x0080_0000	
0x0000_1FFF	ROM
0x0000_0000	

**Figure 4-1: Address Map**

The LT168's Register Map and SPI's Address Map are show as following table:

**Table 4-1: Register Address Location Map**

Base Address	Maximum Size	Usage Module	Chapter
0x4000_0000	64Kbyte	Direct Memory Access Controller (DMAC)	14
0x4001_0000	64Kbyte	Chip Configuration Module (CCM)	8
0x4002_0000	64Kbyte	Reset Control Module (RCM)	10
0x4003_0000	64Kbyte	Clock and Power Control Module (CLKPWRM)	9
0x4004_0000	64Kbyte	Programmable Interrupt Timer (PIT) - PIT0	20
0x4005_0000	64Kbyte	Programmable Interrupt Timer (PIT) – PIT1	20
0x4006_0000	64Kbyte	Programmable Interrupt Timer (PIT) – PIT2	20
0x4007_0000	64Kbyte	Programmable Interrupt Timer (PIT) – PIT3	20
0x4008_0000	64Kbyte	Serial Communication Interface (SCI) - SCI1	25
0x4009_0000	64Kbyte	Serial Communication Interface (SCI) - SCI0	25
0x400A_0000	64Kbyte	Analog Comparator (COMP) - COMP0	29
0x400B_0000	64Kbyte	Analog Comparator (COMP) - COMP1	29
0x400C_0000	64Kbyte	Serial Communication Interface (SCI) - SCI2	25
0x400D_0000	64Kbyte	Pulse Width Modulator (PWM) - PWM0	28
0x400E_0000	64Kbyte	Pulse Width Modulator (PWM) - PWM1	28
0x400F_0000	64Kbyte	Edge Port Module (EPORT) – EPORT0	23
0x4010_0000	64Kbyte	Edge Port Module (EPORT) – EPORT1	23
0x4011_0000	64Kbyte	Analog-to-Digital Converter (ADC)	31
0x4012_0000	64Kbyte	Efuse Control Module (EFM) Option Byte (OPB)	15 16
0x4013_0000	64Kbyte	Watchdog Timer (WDT)	21
0x4014_0000	64Kbyte	Real Time Controller (RTC)	22
0x4015_0000	64Kbyte	Inter-Integrated Circuit (I2C)	27
0x4016_0000	64Kbyte	USB2.0 Full-Speed Device Controller (USBC)	30
0x4017_0000	64Kbyte	Crossbar Switch (XBAR)	13
0x4018_0000	64Kbyte	External Bus Interface (EBI)	19
0x4019_0000	64Kbyte	Cache Module (CACHM)	12
0x401A_0000	64Kbyte	RGB Controller Module (RGBC)	17
0x401B_0000	64Kbyte	Blender Controller (BLDC)	18
0x401C_0000	64Kbyte	CANBus Controller(CANBC)	24
0x401D_0000	64Kbyte	Edge Port Module (EPORT) - EPORT2	23
0x6000_0000	64Kbyte	Synchronous Serial Interface (SSI) - SSI0 / QSPI0	26
0x7000_0000	64Kbyte	Synchronous Serial Interface (SSI) - SSI1 / QSPI1	26
0x8000_0000	64Kbyte	Synchronous Serial Interface (SSI) - SSI2 / QSPI2	26
0xE000_0000	4Kbyte	Embedded Interrupt Controller(EIC)	5
0xE000_1000	4Kbyte	Embedded Programmable Timer (EPT)	7

**Note:**

See module sections for details of how much of each block is being decoded. Accessing to addresses outside the module memory maps (and also the reserved area 0x00CB\_0000 ~ 0x00CF\_FFFF) will not be responded to and will result in a bus monitor transfer error exception.

## 5. Embedded Interrupt Controller(EIC)

This section describes the Embedded Interrupt Controller of the 32-bits RISC.

### 5.1. Introduction

LT168 has an Interrupt Controller that collects requests from multiple interrupt sources and provides an interface to the CPU interrupt logic.

### 5.2. Features

The features of the Interrupt Controller include:

- Configurable interrupt sources, 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

### 5.3. Memory Map and Registers

This subsection describes the memory map (see **Table 5-1**) and registers. The Embedded Interrupt Controller base address is **0xE000\_0000**.

#### 5.3.1. Memory Map

EIC module base address(EIC\_BASEADDR) is defined in 32-bits RISC internal parameter. The default value is **0xE0000000**. 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 5-1: Interrupt Controller Module Memory Map**

Offset Address	Bits[31:24]	Bits[23-16]	Bits[15-8]	Bits[7-0]	Access <sup>(1)</sup>
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 <sup>(2)</sup>				-
Priority level select registers (PLSR0-PLSR31)					
0x0000_0040	PLSR3	PLSR2	PLSR1	PLSR0	S/U

Offset Address	Bits[31:24]	Bits[23-16]	Bits[15-8]	Bits[7-0]	Access <sup>(1)</sup>
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 <sup>(2)</sup>				-

**Notes:**

- (1) In 32-bits RISC, the register can be accessed in any case.
- (2) Accessing to unimplemented address locations have no effect and will result in a cycle termination transfer error.

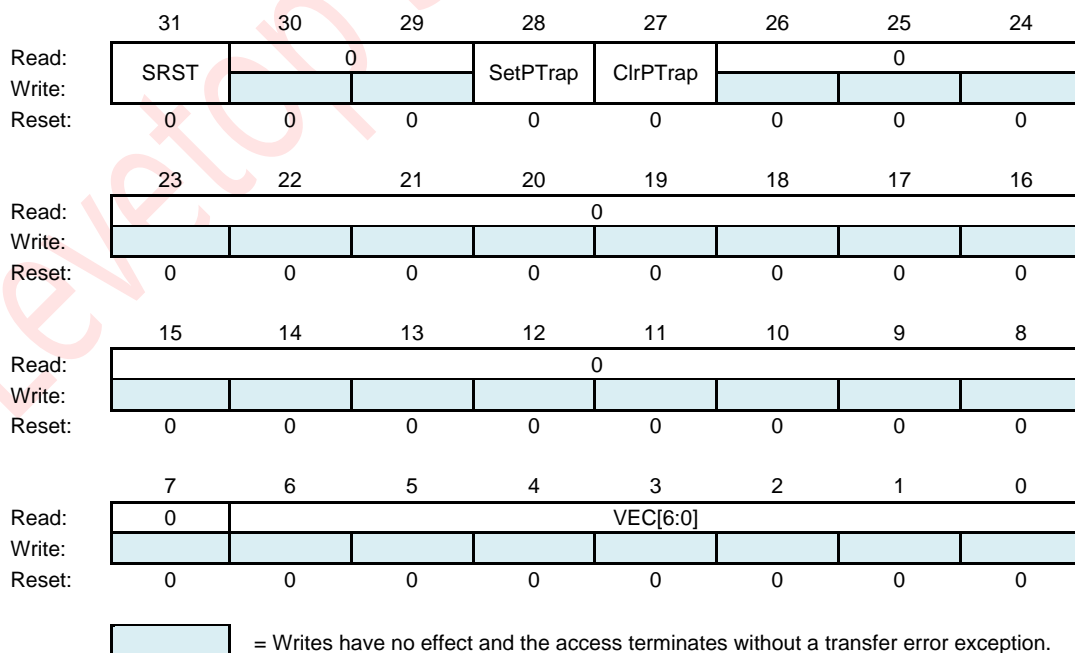
**5.3.2. Register Descriptions**

This subsection contains a description of the interrupt controller module registers. The Embedded Interrupt Controller base address (**EIC\_BASEADDR**) is 0xE000\_0000.

**5.3.2.1. Interrupt Control Status Register (ICSR)**

The 32-bits interrupt control register (ICSR) reflects the state of the interrupt controller

**Address: EIC\_BASEADDR+0x0000\_0000**



**Figure 5-1: 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

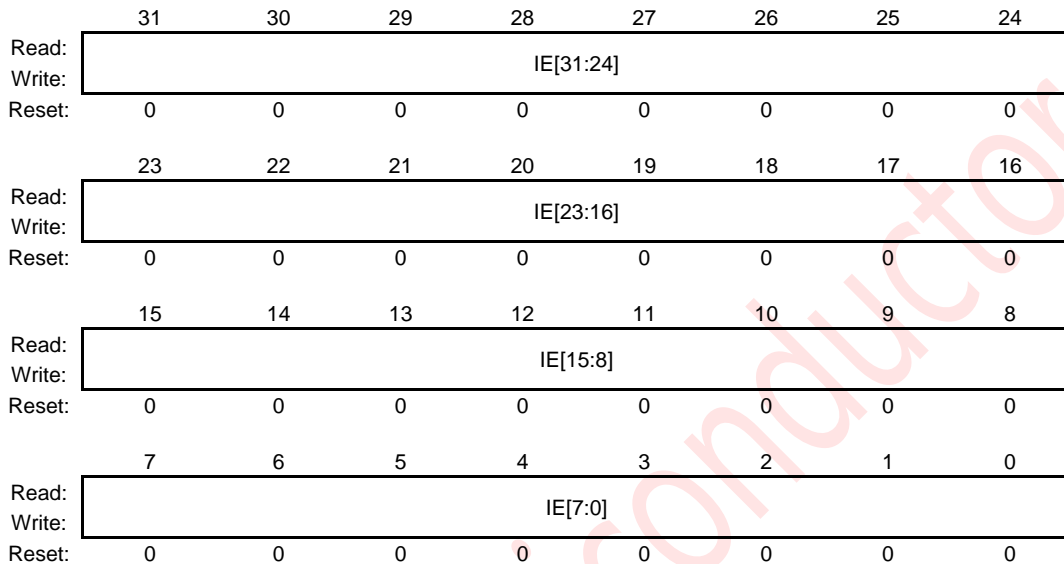
**VEC[6:0]** — Interrupt Vector Number Field

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

**5.3.2.2. Interrupt Enable Register (IER)**

The read/write, 32-bits 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.

**Address: EIC\_BASEADDR+0x0000\_0010**



**Figure 5-2: Interrupt Enable Register (IER)**

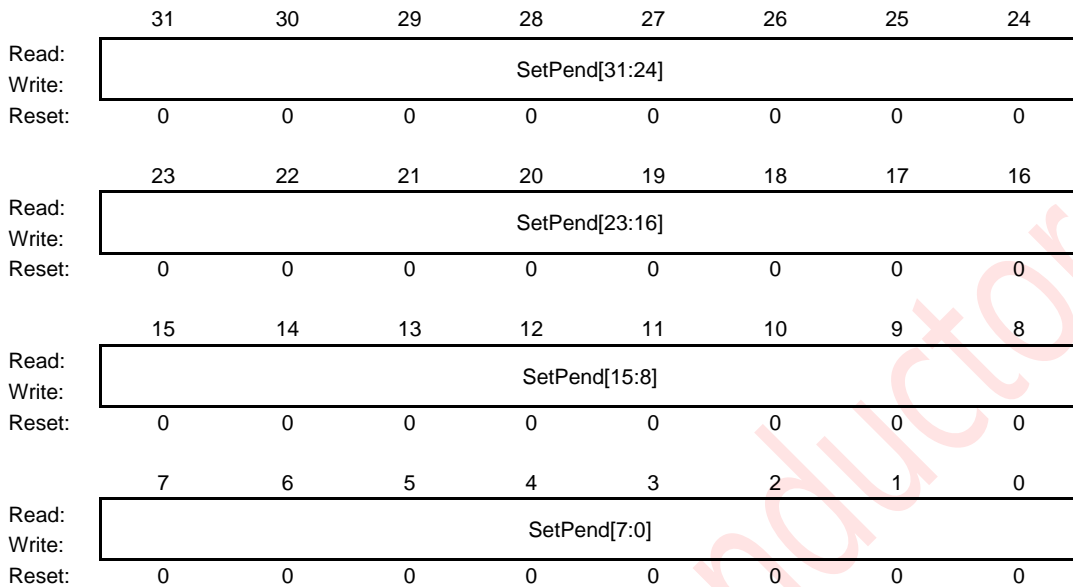
**IE[31:0]** — Interrupt Enable Field

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 is enabled
- 0 = interrupt request disabled

**5.3.2.3. Interrupt Pending Set Register (IPSR)**

**Address: EIC\_BASEADDR+0x0000\_0018**



**Figure 5-3: Interrupt Pending Set Register (IPSR)**

**SetPend[31:0]** — Interrupt Pending Set Field

The read/write SetPend[31:0] field set pending 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

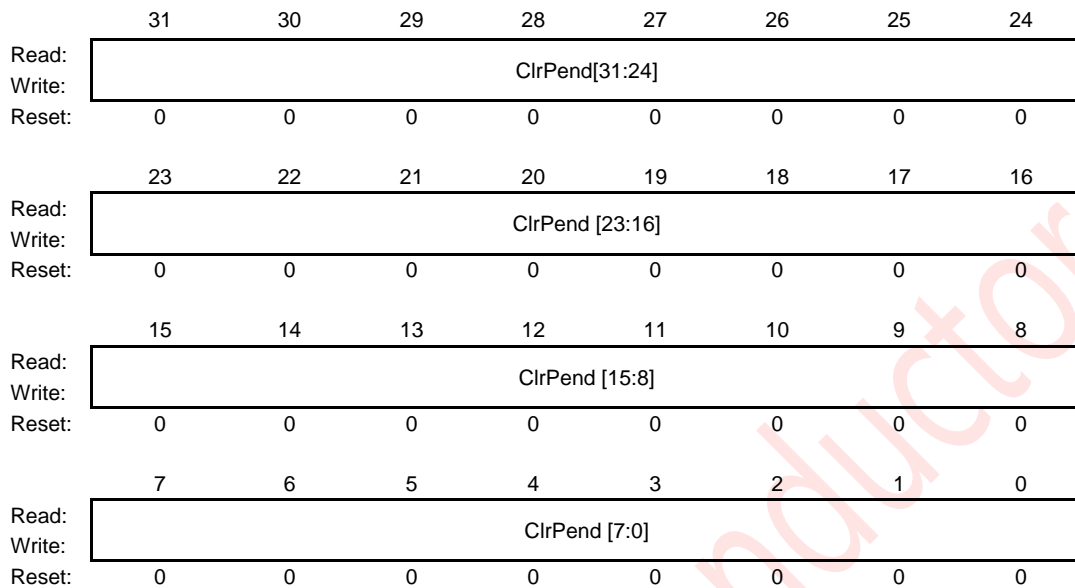
On writes:

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



**5.3.2.4. Interrupt Pending Clear Register (IPCR)**

**Address: EIC\_BASEADDR+0x0000\_001C**



**Figure 5-4: Interrupt Pending Clear Register (IPCR)**

**ClrPend[31:0]** — Interrupt Pending Clear Field

The read/write ClrPend[31:0] field clear pending 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

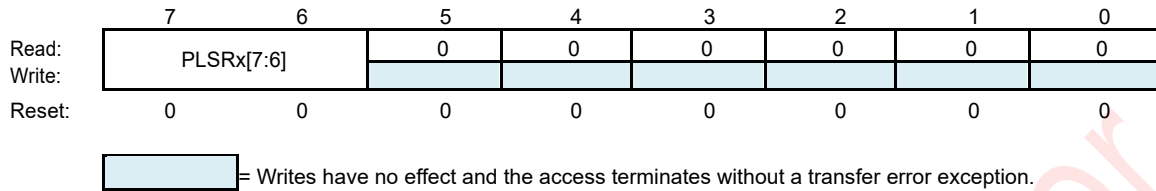
On writes:

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

**5.3.2.5. Priority Level Select Registers (PLSR)**

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

**Address: EIC\_BASEADDR+0x0000\_0040 through EIC\_BASEADDR+0x0000\_005C**



**Figure 5-5: 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. This 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 5-2: Priority Value Adjustment**

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

5.3.2.6. System Priority Level Select Registers (SYSPLSR)

Address: EIC\_BASEADDR+0x0000\_0060

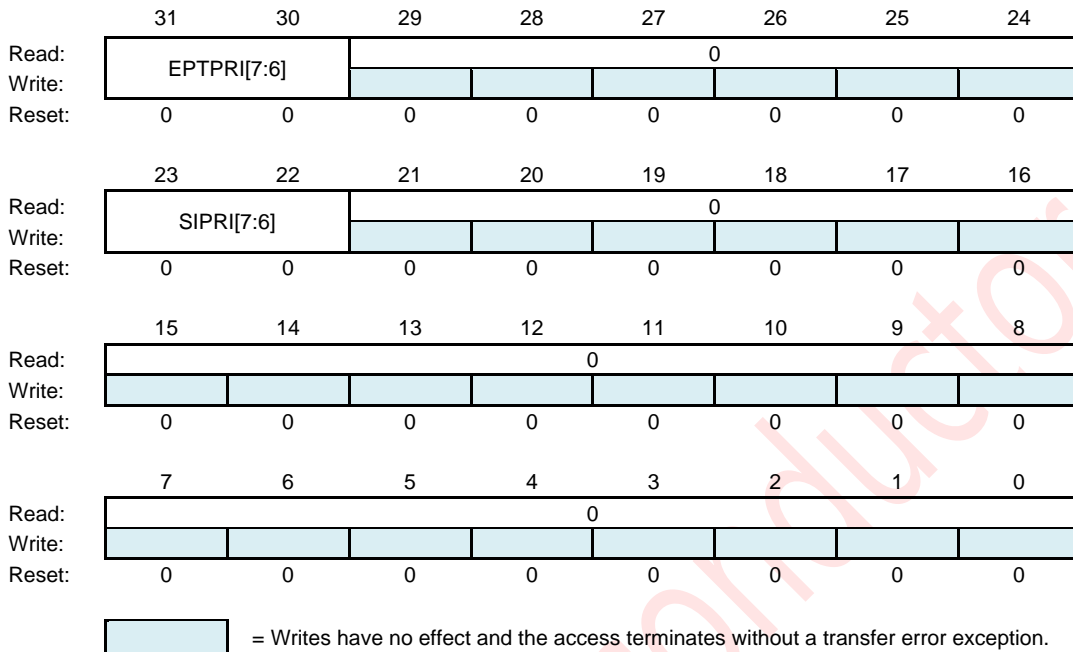


Figure 5-6: System Priority Level Select Registers (SYSPLSR)

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

EPT interrupt's default priority is -2. This means EPT interrupt priority is higher than other normal IRQs and system software interrupt(PendTrap) as default. The lower the value, the higher the priority. However, user can set EPTPRI[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 5-3: Priority Value Adjustment

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

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

Software interrupt(PendTrap) default priority is -1 . This means EPT interrupt priority is higher than other normal IRQs as default. The lower the value, the higher the priority. However, users 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 5-4: Priority Value Adjustment**

SIPRI[7:6]	Priority Value to be increased
00	0
01	64
10	128
11	192

## 5.4. Function Description

EIC supports both level-sensitive and pulse interrupts. The interrupt source number is from 1 to 32.

The interrupt will become pending by one of the following reasons:

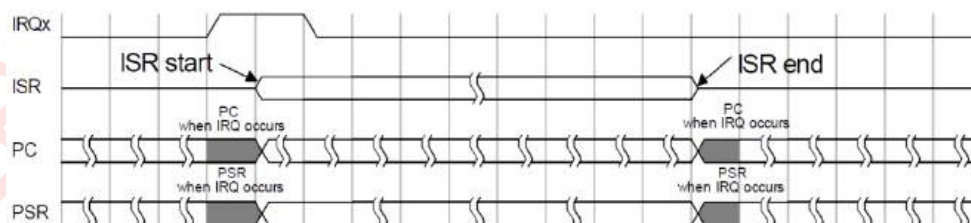
- The EIC detects that the interrupt signal is active and the corresponding interrupt is not active
- The EIC detects a rising edge on the interrupt signal

The pending interrupt remains pending until one of the followings:

- 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 will monitor the interrupt signal continuously. If the interrupt signal 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 Pending Clear Register bit.

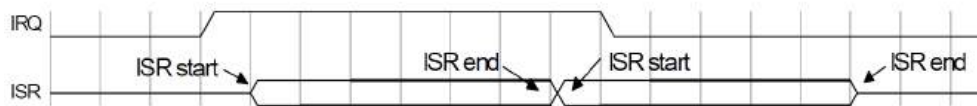
### 5.4.1. Interrupt Handling Without Confliction

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 5-7: One Pulse Interrupt without Confliction**

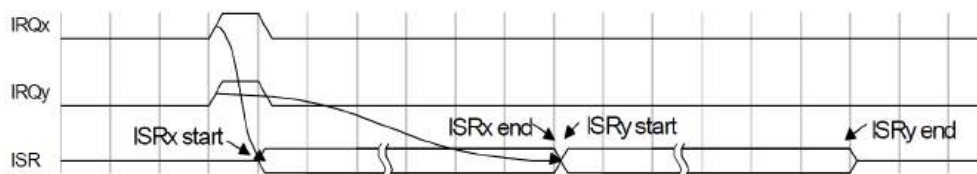
For a level-sensitive interrupt, the state of the interrupt changes to pending if the signal is asserted. Without confliction, the interrupt causes the processor to immediately enter the ISR. When the processor returns from the ISR, EIC continues to sample the interrupt signal. If the signal is not cleared,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



**Figure 5-8: Level-sensitive Interrupt without Conflict**

**5.4.2. Interrupt With Confliction**

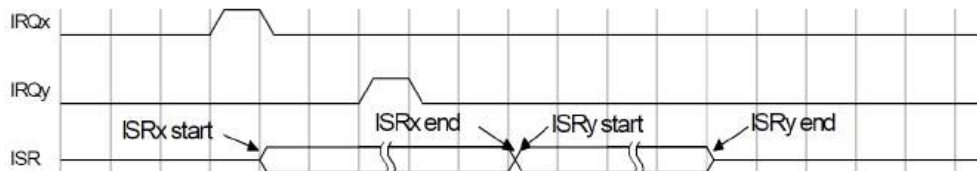
When two interrupt signals are asserted at the same time, the Interrupt Arbiter will judge which one has the greater priority. For instance, if the priority of IRQx is greater than IRQy, the processor will enter ISRx and IRQy becomes pending. After the processor returns from ISRx, the processor will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



**Figure 5-9: Two Interrupts Occur at the Same Time**

If an interrupt signal is asserted during another interrupt handling, then there will be 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 5-10: A lower Priority Interrupt Asserted with Confliction**



**Figure 5-11: A higher Priority Interrupt Asserted with Confliction**

**5.4.3. 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 .

## 5.5. Interrupts

The interrupt controller assigns a number to each interrupt source, as **Table 5-5** shows.

**Table 5-5: Interrupt Source Assignment**

Source	Module	Flag	Source Description	Flag Clearing Mechanism
0	ADC			
1	QSPI0	XR XOIS	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	COMP1	CPRIF		
		CPFIF		
5	DMAC	DONE[0]		Write DONE[0] = 1
		DONE[1]		Write DONE[1] = 1
		DONE[14]		Write DONE[14] = 1
		DONE[15]		Write DONE[15] = 1
		DMA_ESR[GPE]	Group Priority Error	Write channel number to CERR[6:0] to clear error status

Source	Module	Flag	Source Description	Flag Clearing Mechanism
		DMA_ESR[CPE]	Channel Priority Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[SAE]	Source Address Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[SOE]	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[SGE]	Scatter/Gather Configuration Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[SBE]	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	PWM1	PIFR[0]		
		PIFR[1]		
		PIFR[2]		
		PIFR[3]		
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	PIT2	PIF	PIT Flag	Writing a 1 to it or writing to PMR
12	PIT3	PIF	PIT Flag	Writing a 1 to it or writing to PMR
13	RTC	Day_intf	Day pulse flag	
		Hou_intf	Hour pulse flag	
		Min_intf	Minute pulse flag	
		Sec_intf	Second pulse flag	
		Ala_intf	Alarm flag	
		1KHz_intf	1KHz pulse flag	
		32KHz_intf	32KHz pulse flag	
14	USB_DEV	USB_DEV Flag	USB_DEV Flag	
15	I2C	I2C Flag	I2C Flag	
16	EPORT2	EPF0	Edge port 2 flag 0	Write EPF0 = 1
		EPF1	Edge port 2 flag 1	Write EPF1 = 1
		EPF2	Edge port 2 flag 2	Write EPF2 = 1

Source	Module	Flag	Source Description	Flag Clearing Mechanism
		EPF3	Edge port 2 flag 3	Write EPF3 = 1
		EPF4	Edge port 2 flag 4	Write EPF4 = 1
		EPF5	Edge port 2 flag 5	Write EPF5 = 1
		EPF6	Edge port 2 flag 6	Write EPF6 = 1
		EPF7	Edge port 2 flag 7	Write EPF7 = 1
17	PVD	PVDO	PVD Flag	
18	CANBUS	CAN_IFRH[BUF63:BUF0]	CAN buffers 63–0 interrupts	This bit is cleared by writing it to '1'
19	CANBUS	CAN_ESR[BOFF_INT]	CANBus 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	BLENDE	INT_END	Transfer Complete	
		INT_OQ	One quarter of the blending operation of the currently data source is completed	
		INT_HALF	Halt of the blending operation of the currently data source is completed	
25	RGBC	INTV	The completion flag of one frame transmission	
		INTN	Transmission completion flag for the specified number of frames	
26	QSPI1	XR XOIS	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	
27	QSPI2	XR XOIS	XIP Receive FIFO Overflow Interrupt	
		RXFIS	Receive FIFO Full Interrupt	
		RXOIS	Receive FIFO Overflow Interrupt	
		RXUIS	Receive FIFO Underflow Interrupt	



Source	Module	Flag	Source Description	Flag Clearing Mechanism
		TXOIS	Transmit FIFO Overflow Interrupt	
		TXEIS	Transmit FIFO Empty Interrupt	
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	
		OR	Receiver Overrun Flag	
		NF	Noise Flag	
		FE	Framing Error Flag	
		PF	Parity Error Flag	
		RXUF	Receiver Buffer Underflow Flag	
		29	SCI2	TDRE
TC	Transmission Complete			
TXOF	Transmitter Buffer Overflow Flag			
LBKDIF	LIN Break Detect Interrupt Flag			
IDLE	Idle Line Flag			
RXEDGIF	RXD2 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			
30	EPORT0	EPF0	Edge port 0 flag 0	Write EPF0 = 1
		EPF1	Edge port 0 flag 1	Write EPF1 = 1

Source	Module	Flag	Source Description	Flag Clearing Mechanism
		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
EPF1	Edge port 1 flag 1			Write EPF1 = 1
EPF2	Edge port 1 flag 2			Write EPF2 = 1
EPF3	Edge port 1 flag 3			Write EPF3 = 1
EPF4	Edge port 1 flag 4			Write EPF4 = 1
EPF5	Edge port 1 flag 5			Write EPF5 = 1
EPF6	Edge port 1 flag 6			Write EPF6 = 1
EPF7	Edge port 1 flag 7			Write EPF7 = 1

## 6. 32-bits RISC Introduction

This section describes the functionality of the 32-bits RISC 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, 32-bits RISC is built on a new 3-stage pipeline von Neumann architecture.

The 32-bits RISC 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 32-bits RISC design. By leveraging these configurable options, tradeoff among performance, functionality and cost are more flexible. The gate count of 32-bits RISC varies from 12K to 20K with different configurations.

### 6.1. Features

The main features of the 32-bits RISC are as follows:

- 32-bits load/store reduced instruction set computer (RISC) architecture with fixed 16-bits instruction length
- 16 entry 32-bits 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 byte/halfword/word memory accesses
- Embedded interrupt controller, support nested vector interrupts and low power mode wakeup
- Single-cycle 32-bits x 32-bits hardware integer multiplier array
- 3~13 cycles hardware integer divider array
- AHB-lite external bus

### 6.2. Microarchitecture Summary

The 32-bits RISC utilizes a 3-stage pipeline for instruction execution. The instruction fetch, instruction decode/register file read, and 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 32-bits 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 (32-bits) 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 be executed 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 32-bits. 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.

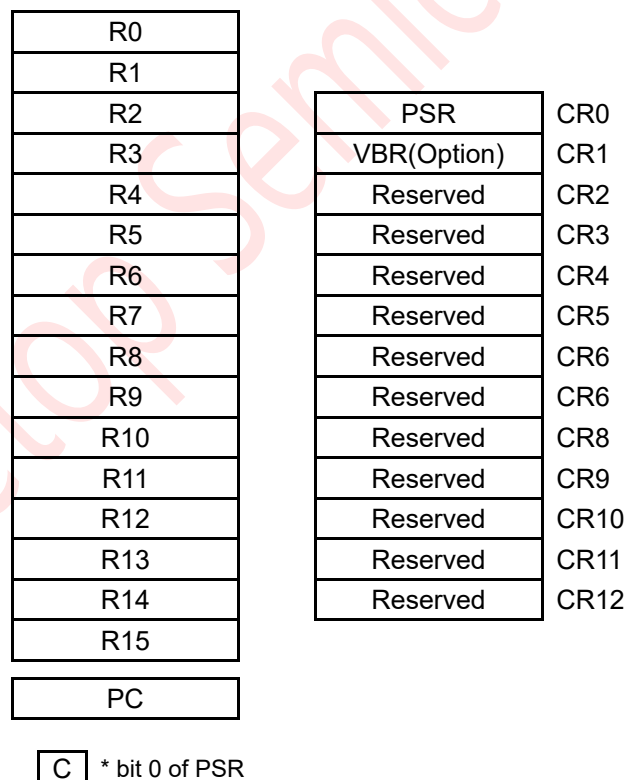
### 6.3. Programming Model

The 32-bits RISC 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 can be executed 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.

The registers depicted in the programming model (see **Figure 6-1**) provide operand storage and control. The user programming model consists of 16 general-purpose 32-bits registers, the 32-bits 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.

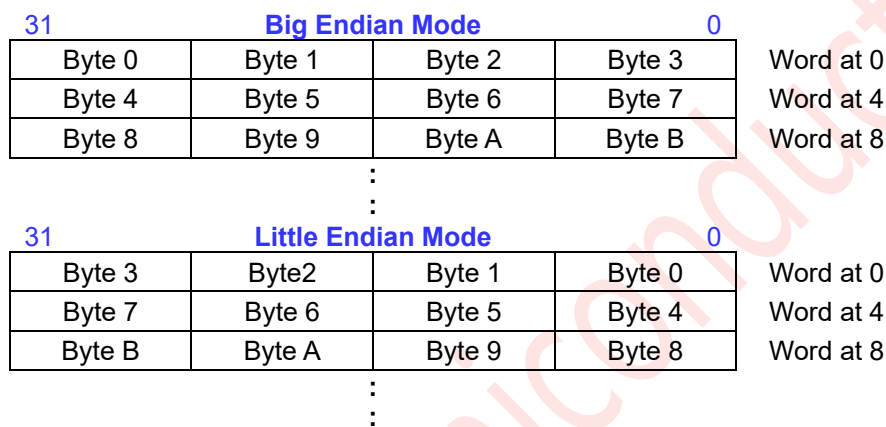


**Figure 6-1: Programming Model**

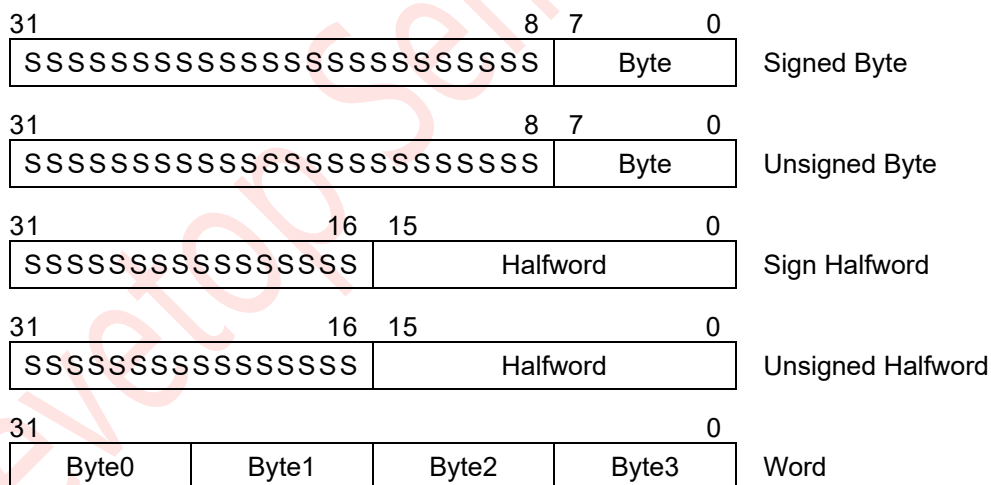
### 6.4. Data Format Summary

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 32-bits of the source operand(s) and generate a 32-bits result.

Memory may be viewed from either a Big Endian or Little Endian byte ordering perspective depending on the processor configuration (see **Figure 6-2**). 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 byte of word 0 is located at address 3. Within registers, bits are numbered within a word starting with bit 31 as the most significant bit (see **Figure 6-3**). 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 6-2: Data Organization in Memory**



**Figure 6-3: Data Organization in Registers**

### 6.5. Operand Addressing Capabilities

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

### 6.6. 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 6-1** provides an alphabetized listing of the 32-bits RISC instruction set.

**Table 6-1: 32-bits RISC 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
ASRI	Arithmetic Shift Right Immediate
BCLRI	Clear Bit
BF	Branch on Condition False
BGENI	Bit Generate Immediate
BGENR	Bit Generate Register
BKPT	Breakpoint
BMASKI	Bit Mask Immediate
BR	Branch
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
BTSTI	Bit Test Immediate
CLRF	Clear Register on Condition False

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

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

**Note (1)** : Not implemented in the current version.



## 7. Embedded Programmable Timer (EPT)

### 7.1. Introduction

Embedded Programmable Timer(EPT) is a 24-bits timer that provides precise interrupts at regular intervals with minimal processor intervention. The timer can either count down from the reload value, or 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.

### 7.2. Memory Map and Registers

#### 7.2.1. Memory Map

EPT base address is defined as EIC\_BASEADDR + 0x1000. The default based address (EPT\_BASEADDR) is 0xE000\_1000. Table 7-1 shows the offset address of EPT registers. EPT module occupies 4K address area.

**Table 7-1: Programmable Timer Module Memory Map**

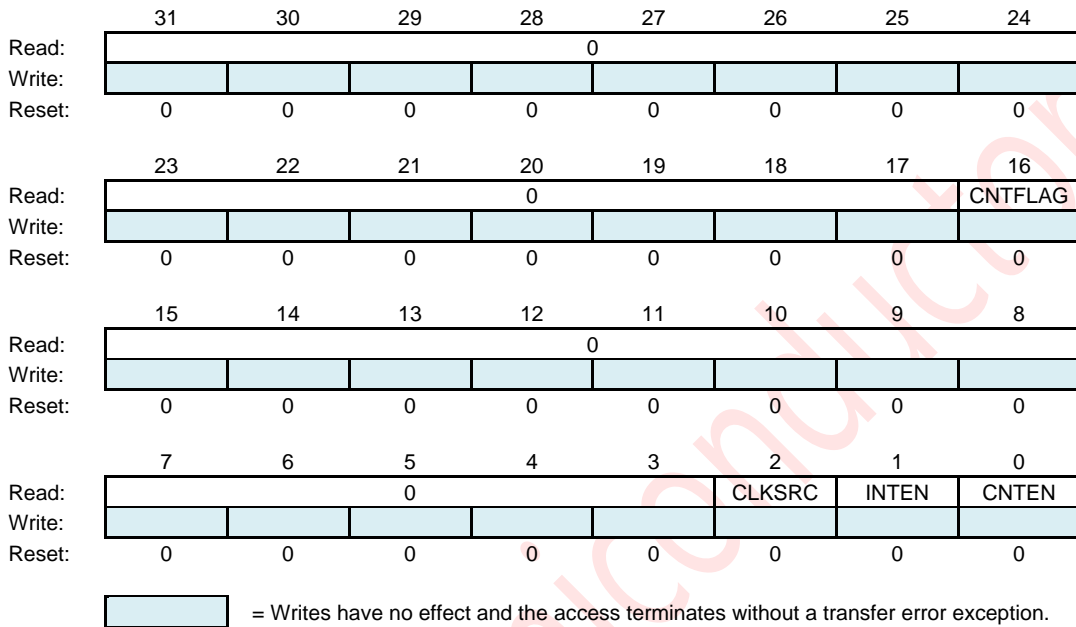
Offset Address	Bits[31:0]	Access
0x0000_0000	EPT Control and Status Register (EPTCSR)	S/U
0x0000_0004	EPT Reload Register (EPTRLD)	S/U
0x0000_0008	EPT Count Register (EPTCNT)	S/U
0x0000_000C	Reserved	S/U

**7.2.2. Register Descriptions**

This subsection contains a description of the EPT module registers.

**7.2.2.1. EPT Control Status Register (EPTCSR)**

**Address: EPT\_BASEADDR+0x0000\_0000**



**Figure 7-1: 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.

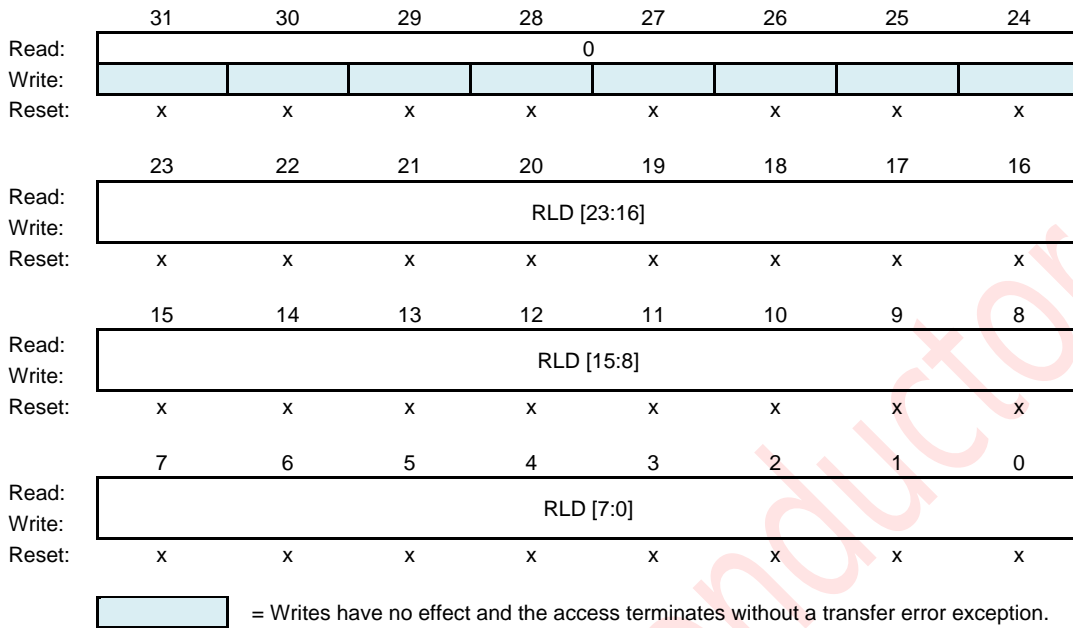
**CNTEN** — Counter Enable

The read/write bit is used to enable EPT's counter. It will be reset by RESET.

- 1 = Counter is enabled
- 0 = Counter is disabled

**7.2.2.2. EPT Reload Register (EPTRLD)**

**Address: EPT\_BASEADDR+0x0000\_0004**



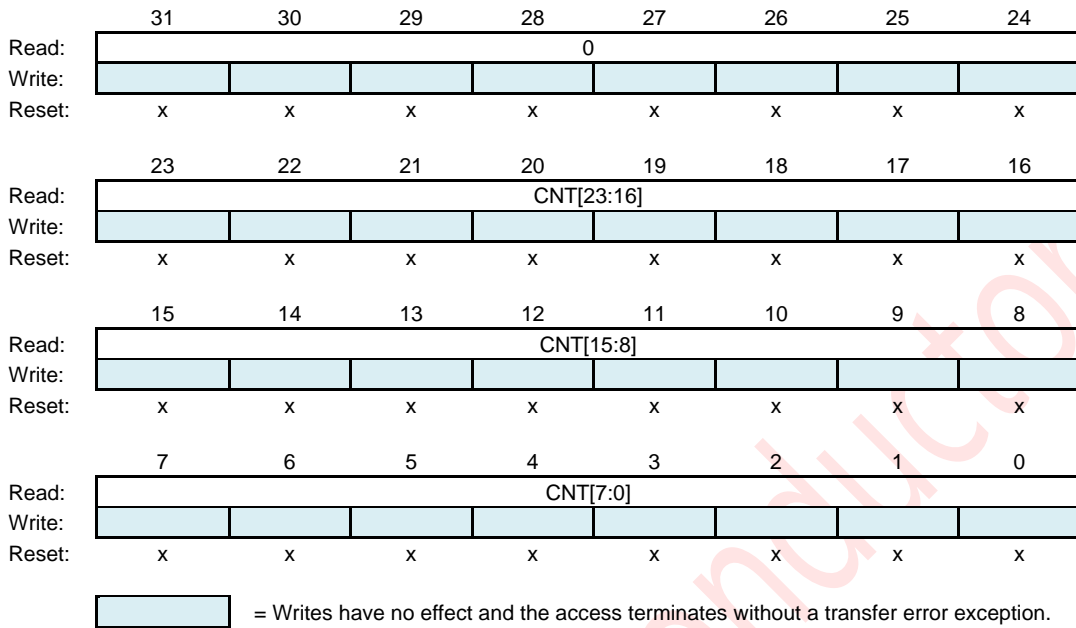
**Figure 7-2: EPT Reload Register (EPTRLD)**

**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 0x00000001 ~ 0x00FFFFFF. Value 0 has no effect. To generate a period timer with N clock cycles, set RLD to N-1.

**7.2.2.3. EPT Count Register (EPTCNT)**

**Address: EPT\_BASEADDR+0x0000\_0008**



**Figure 7-3: 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 no reset value.

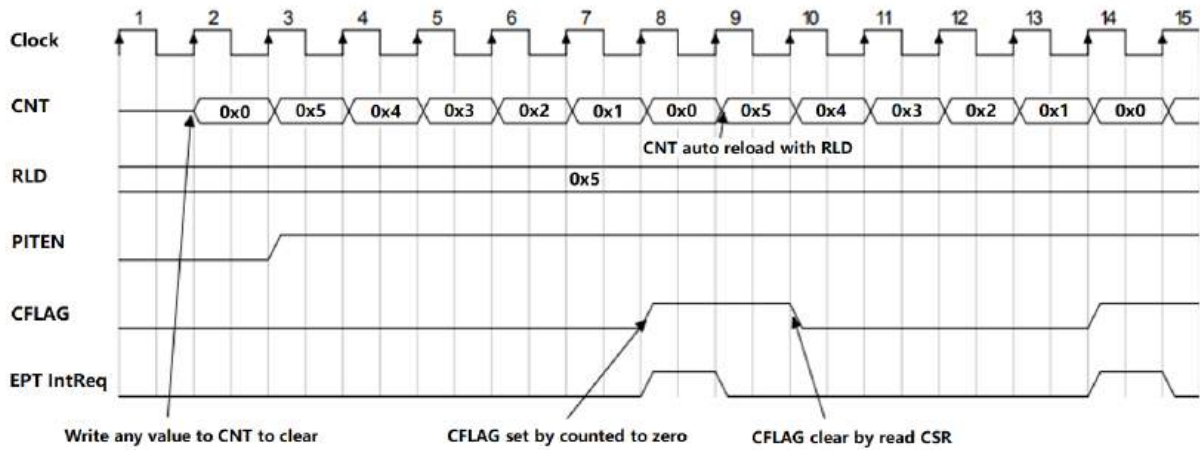
Reads will return the current value of EPT counter. A write of any value to this register will clear the counter value to 0 and also the CNTFLAG to 0.

### 7.3. Function Description

When Enabled, EPT counts 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 zero to RLD disables the counter on next wrap. When EPT counts to zero, the CLFLAG bit will be set to 1, and then the EPT will trigger the EPT interrupt if INTEN is enabled.

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

#### 7.3.1. Count Timing



**Figure 7-4: EPT Count Timing**

## 8. Chip Configuration Module (CCM)

### 8.1. Introduction

The Chip Configuration Module (CCM) of LT168 controls the chip configuration.

### 8.2. Features

The CCM performs below operations:

- Configure wakeup function
- Configure LDO mode
- Configure IO function

### 8.3. Memory Map and Registers

This subsection provides a description of the memory map and registers. The CCM base address is **0x4001\_0000**. **Table 8-1** shows the offset address of CCM registers.

#### 8.3.1. Memory Map

**Table 8-1: CCM Memory Map**

Offset Address	Bits[31:16]	Bits[15:0]	Access <sup>1</sup>
0x0000	WKUPC — Wakeup Configuration Register		S
0x0004	Reserved		S
0x0008	Reserved		S
0x000C	CPPDC — Chip Pin Pull Down Configuration Register		S
0x0010	Reserved		S
0x0014	QSPIXIPCR — QSPI XIP Mode Configuration Register		S
0x0018	Reserved		S
0x001C	QSPIKEYR — QSPI 32-Bit Key Register		S
0x0020	QSPIGPIOCR — QSPI GPIO Configuration Register		S
0x0024	MCURAMPRIOCR — MCU Access RAM Priority Configuration Register		S
0x0028	EPORT2FCR — EPORT2 Function Configuration Register		S

**Note:**

S = supervisor-only access. Accessing supervisor only address locations in user mode has no effect and result in a cycle termination transfer error.

**8.3.2. Register Descriptions**

**8.3.2.1. Wakeup Configuration Register (WKUPC)**

**Address: CCM\_BASEADDR+0x0000\_0000**

	31	30	29	28	27	26	25	24
Read:	WKUPFILT		WKUPSEN [30:24]					
Write:	EREN							
Reset:	0	1	1	1	1	1	1	1
	23	22	21	20	19	18	17	16
Read:	WKUPSEN [23:16]							
Write:								
Reset:	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8
Read:	WKUPSEN [15:8]							
Write:								
Reset:	1	1	1	1	1	1	1	1
	7	6	5	4	3	2	1	0
Read:	WKUPSEN [7:0]							
Write:								
Reset:	1	1	1	1	1	1	1	1

**Figure 8-1: Wakeup Configuration Register (WKUPC)**

**WKUPFILTEREN** — Wakeup Source Filter Enable

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

- 1 = Wakeup source filter is enabled
- 0 = Wakeup source filter is disabled

**WKUPSEN[30:0]** — Wakeup Source Enable

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

**Table 8-2: WKUPSEN and Corresponding Wakeup Source**

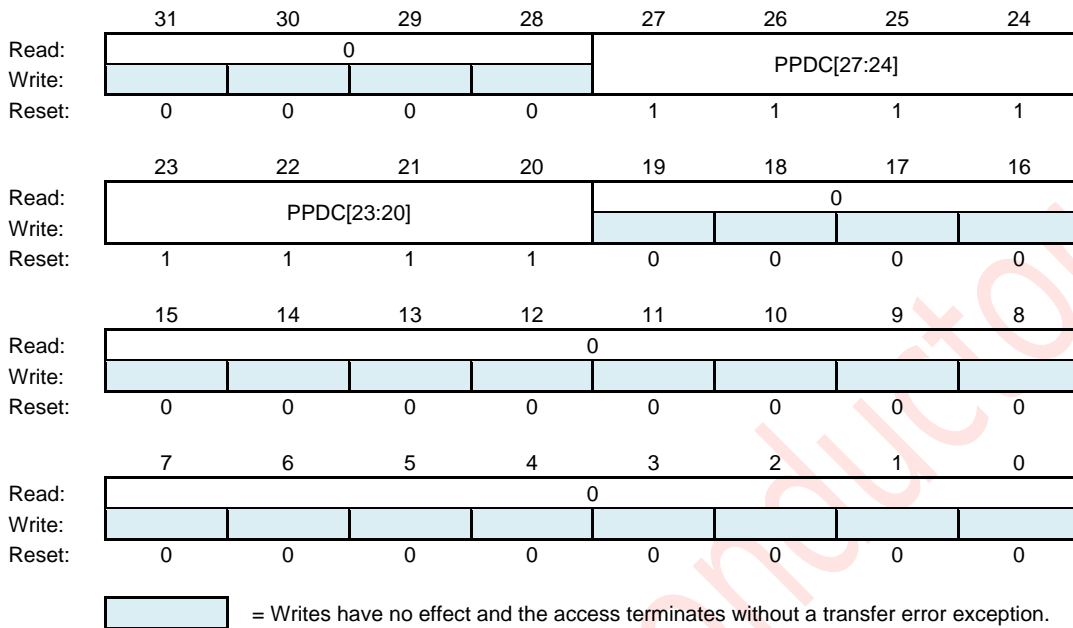
WKUPSEN	Wakeup Source
WKUPSEN[30]	INT2[3]
WKUPSEN[29]	INT2[2]
WKUPSEN[28]	INT2[1]
WKUPSEN[27]	RTC Interrupt
WKUPSEN[26]	PVD Interrupt
WKUPSEN[25]	USB Resume
WKUPSEN[24]	COMP1 Interrupt
WKUPSEN[23]	COMP0 Interrupt
WKUPSEN[22]	INT2[6]
WKUPSEN[21]	INT2[5]
WKUPSEN[20]	I2C

<b>WKUPSEN</b>	<b>Wakeup Source</b>
WKUPSEN[19]	WDT0 Interrupt
WKUPSEN[18]	JTAG POWER ON REQUEST
WKUPSEN[17]	RESET# Pin
WKUPSEN[16]	WDT0 Reset
WKUPSEN[15]	INT1[7]
WKUPSEN[14]	INT1[6]
WKUPSEN[13]	INT1[5]
WKUPSEN[12]	INT1[4]
WKUPSEN[11]	INT1[3]
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]



**8.3.2.2. Chip Pin Pull Down Configuration Register (CPPDC)**

**Address: CCM\_BASEADDR+0x0000\_000C**



**Figure 8-2: Chip Pin Pull Down Configuration Register (CPPDC)**

**PPDC[27:20]** — Pin Pull Down Configuration Field

This read/write field controls the pull down function of the corresponding pin which shows in **Table 8-3**.

- 1 = Pull down function of the corresponding pin is enabled
- 0 = Pull down function of the corresponding pin is disabled

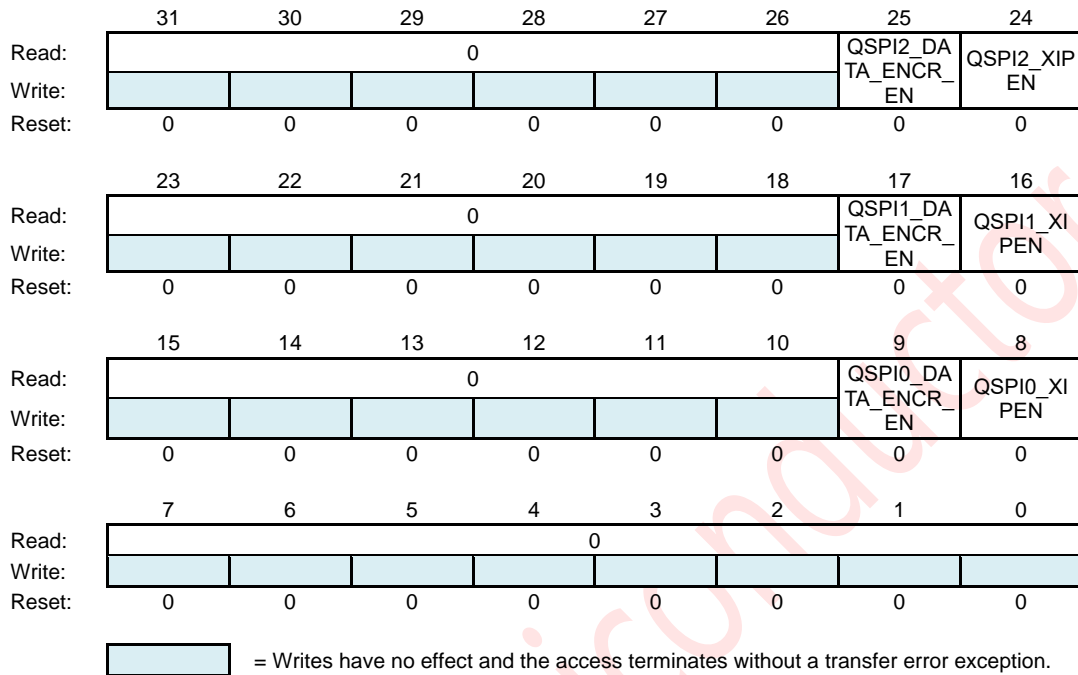
**Table 8-3: Chip Pin Pull Down Configuration**

Pin Name	Pull Down Configuration Bit	Pull Down Function
PWM0[3]	PPDC[27]	0: Disable, 1: Enable
PWM0[2]	PPDC[26]	0: Disable, 1: Enable
PWM0[1]	PPDC[25]	0: Disable, 1: Enable
PWM0[0]	PPDC[24]	0: Disable, 1: Enable
PWM1[3]	PPDC[23]	0: Disable, 1: Enable
PWM1[2]	PPDC[22]	0: Disable, 1: Enable
PWM1[1]	PPDC[21]	0: Disable, 1: Enable
PWM1[0]	PPDC[20]	0: Disable, 1: Enable

**8.3.2.3. QSPI XIP Mode Configuration Register (QSPIXIPMCFR)**

The QSPIXIPMCFR register is a read-writable register.

**Address: CCM\_BASEADDR+0x0000\_0014**



**Figure 8-3: QSPI XIP Mode Configuration Register (QSPIXIPMCFR)**

**QSPi<sub>x</sub>\_DATA\_ENCR\_EN** — QSPi<sub>x</sub> XIP Transfer Data Encrypt function Enable Control bit

- 1 = QSPi<sub>x</sub> XIP Transfer Data Encrypt function is Enabled
- 0 = QSPi<sub>x</sub> XIP Transfer Data Encrypt function is Disabled

**QSPi<sub>x</sub>\_XIPEN** — QSPi<sub>x</sub> XIP Mode Enable Control bit


- 1 = QSPi<sub>x</sub> XIP Mode is Enabled
- 0 = QSPi<sub>x</sub> XIP Mode is Disabled

**8.3.2.4. QSPI 32-Bit Key Register (QSPILKEYR)**

The QSPIKEYR register is a writable register, and always returns 0's when being read..

**Address: CCM\_BASEADDR+0x0000\_001C**

Read:	0							
Write:	QSPI_KEY[31:24]							
Reset:	0	0	0	0	0	0	0	0
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[23:16]							
Reset:	0	0	0	0	0	0	0	0
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[15:8]							
Reset:	0	0	0	0	0	0	0	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 8-4: QSPI 32-Bit Key Register (QSPILKEYR)**

**8.3.2.5. QSPI GPIO Configuration Register (QSPIGPIOCR)**

The QSPIGPIOCR register is a read-writable register.

**Address: CCM\_BASEADDR+0x0000\_0020**

Read:	QSPI2_SS	QSPI1_SS	QSPI0_SS	0		QSPI2_SS	QSPI1_SS	QSPI0_SS
Write:	_GPIOEN	_GPIOEN	_GPIOEN			_PUE	_PUE	_PUE
Reset:	0	0	0	0	0	0	0	0
Read:	0					QSPI2_SS	QSPI1_SS	QSPI0_SS
Write:						_OBE	_OBE	_OBE
Reset:	0	0	0	0	0	0	0	0
Read:	0					QSPI2_SS	QSPI1_SS	QSPI0_SS
Write:						_DO	_DO	_DO
Reset:	0	0	0	0	0	1	1	1
Read:	0					QSPI2_SS	QSPI1_SS	QSPI0_SS
Write:						_DI	_DI	_DI
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: QSPI GPIO Configuration Register (QSPIGPIOCR)**

**QSPiX\_SS\_GPIOEN** — QSPiX CS# Pin GPIO Mode Enable Control bit

- 1 = QSPiX CS# Pin GPIO Mode Is enabled
- 0 = QSPiX CS# Pin GPIO Disabled

**QSPiX\_SS\_PUE** — QSPiX CS# Pin Pullup Enable Control bit in GPIO Input Mode.

**Note:** QSPi2\_SS\_PUE is not valid for LT168.

- 1 = QSPiX CS# Pin GPIO Pullup Is enabled
- 0 = QSPiX CS# Pin GPIO Pullup Disabled

**QSPiX\_SS\_OBE** — QSPiX CS# Pin Output Enable Control bit in GPIO Mode

- 1 = QSPiX CS# Pin GPIO Output Is enabled
- 0 = QSPiX CS# Pin GPIO Output Disabled

**QSPiX\_SS\_DO** — QSPiX CS# Pin Output Data in GPIO Output Mode

- 1 = QSPiX CS# will be driven High in GPIO Output Mode
- 0 = QSPiX CS# will be driven Low in GPIO Output Mode

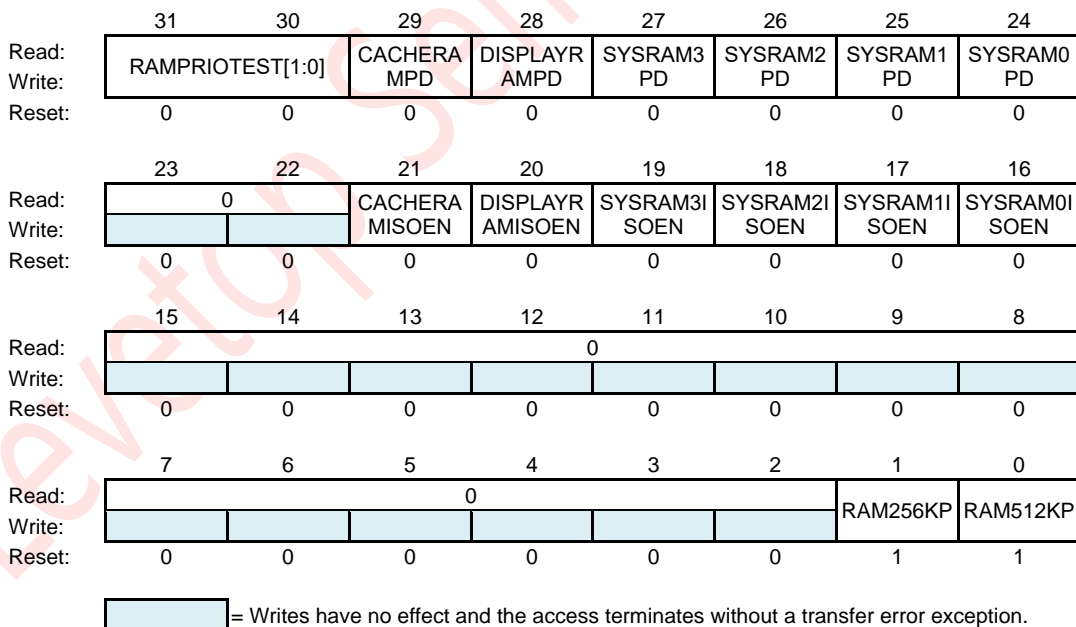
**QSPiX\_SS\_DI** — QSPiX CS# Pin value in GPIO Mode

- 1 = Indicated QSPiX CS# Pin is High level in GPIO Input Mode
- 0 = Indicated QSPiX CS# Pin is Low level in GPIO Input Mode

**8.3.2.6. MCU Access RAM Priority Configuration Register (MCURAMPRIOCR)**

The MCURAMPRIOCR register is a read-writable register.

**Address: CCM\_BASEADDR+0x0000\_0024**



**Figure 8-6: MCU Access RAM Priority Configuration Register (MCURAMPRIOCR)**

**RAMPRIOTEST[1:0]** — MCURAMPRIOCR Write Access Sequence In

The writable bit of MCURAMPRIOCR register cannot be changed, unless the correct sequence is written. The right sequence is: 2'b01 → 2'b10 → 2'b11. After these two bits are written by this sequence, these two bits' value == 2'b11, then the writable bit of MCURAMPRIOCR register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

**CACHERAMPD** — The power supply of Cache RAM will be shut-down when writing 1'b1 to it.

**DISPLAYRAMPD** — Please refer to CACHERAMPD bit description.

**SYSRAM3PD** — Please refer to CACHERAMPD bit description.

**SYSRAM2PD** — Please refer to CACHERAMPD bit description.

**SYSRAM1PD** — Please refer to CACHERAMPD bit description.

**SYSRAM0PD** — Please refer to CACHERAMPD bit description.

**CACHERAMISOEN** — Cache RAM output value will be isolated when writing 1'b1 to it.

**Note:** Cache RAM output should be isolated before the power of Cache RAM is shut-off to avoid leakage current.

**DISPLAYRAMISOEN** — Please refer to CACHERAMISOEN bit description.

**SYSRAM3ISOEN** — Please refer to CACHERAMISOEN bit description.

**SYSRAM2ISOEN** — Please refer to CACHERAMISOEN bit description.

**SYSRAM1ISOEN** — Please refer to CACHERAMISOEN bit description.

**SYSRAM0ISOEN** — Please refer to CACHERAMISOEN bit description.

**RAM256KP** — MCU Accesses System RAM256K Priority Configuration Register

This bit determines whether MCU has higher priority when MCU and Blender access System RAM256K at the same time.

1 = MCU has higher priority when MCU and Blender access System RAM256K at the same time.

0 = MCU has lower priority when MCU and Blender access System RAM256K at the same time.

**RAM512KP** — MCU Accesses Display RAM512K Priority Configuration Register

This bit determines whether MCU has higher priority when MCU and Blender access Display RAM512K at the same time.

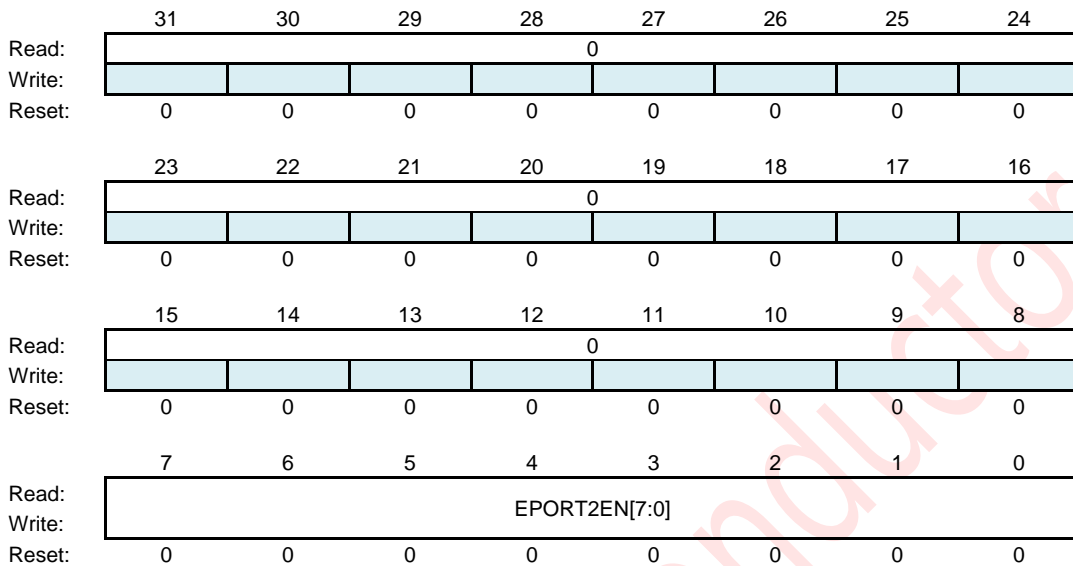
1 = MCU has higher priority when MCU and Blender access Display RAM512K at the same time.

0 = MCU has lower priority when MCU and Blender access Display RAM512K at the same time.

8.3.2.7. EPORT2 Function Configuration Register (EPORT2FCR)

The EPORT2FCR register is a read-writable register.

Address: CCM\_BASEADDR+0x0000\_0028



[shaded box] = Writes have no effect and the access terminates without a transfer error exception.

Figure 8-7: EPORT2 Function Configuration Register (EPORT2FCR)

**EPORT2EN[7:0]** — EPORT2 Function Enable control bit.

Table 8-4: EPORT2 Function Control Bit

EPORT2 Enable Bit	1'b1	1'b0
EPORT2EN[7]	INT2[7]	CANRX
EPORT2EN[6]	INT2[6]	CANTX
EPORT2EN[5]	INT2[5]	RXD2
EPORT2EN[4]	INT2[4]	TXD2
EPORT2EN[3]	INT2[3]	RXD1
EPORT2EN[2]	INT2[2]	TXD1
EPORT2EN[1]	INT2[1]	RXD0
EPORT2EN[0]	INT2[0]	TXD0

## 9. Clock and Power Control Module (CLKPWRM)

### 9.1. Overview

The Clock Module contains:

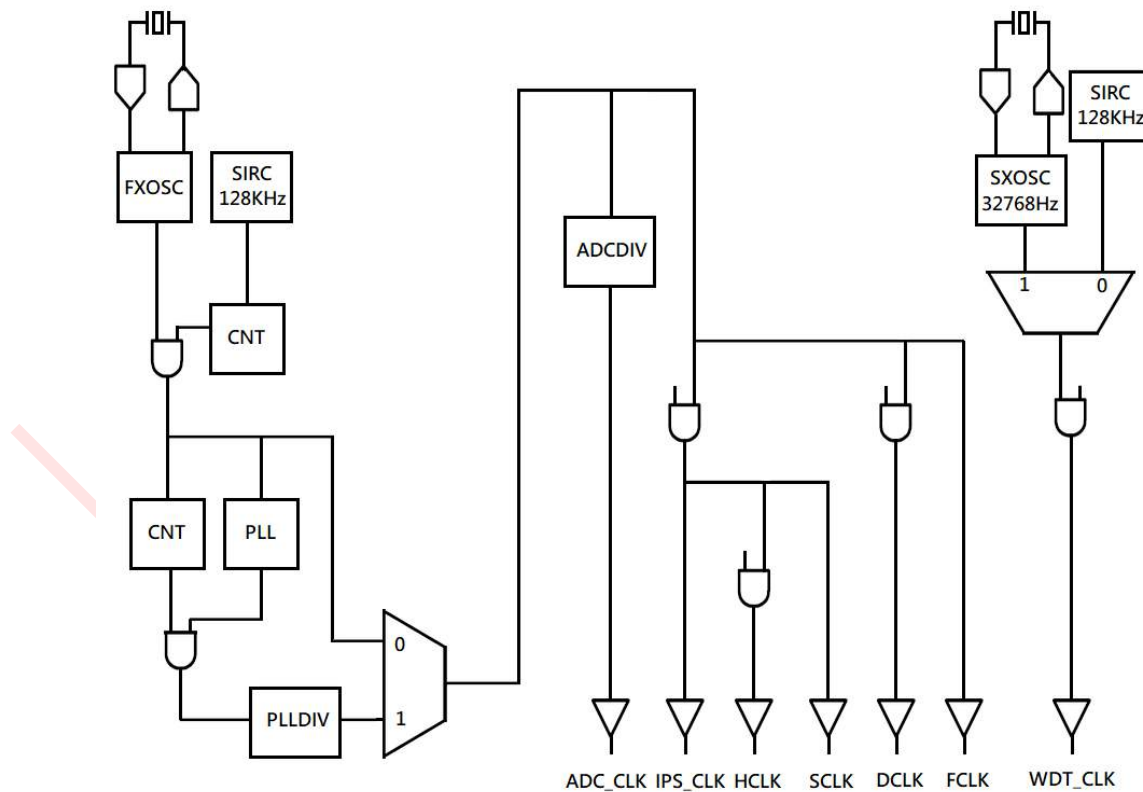
- PLL: Internal VCO PLL
- FXOSC: External Fast Speed Crystal Oscillator(12MHz)
- SIRC: Internal Low Speed 128Khz Oscillator
- SXOSC: External Low Speed Crystal Oscillator(32768Hz)
- Status and Control Registers
- Clock and Power Control Logic

### 9.2. Features

Features of the clock module include:

- Two System Clock Sources
  - Internal PLL Clock
  - External Fast Speed Crystal Oscillator(FXOSC)
- Individual Clock Divider for IPS, System and ADC Clock
- Support Low-power Mode
- Modules can be separately stopped by setting MSCR

### 9.3. Clock Structure



**Figure 9-1: Clock Structure**

LT168\_DS\_ENG / V1.5

## 9.4. Clock Source Select

System Clock Source can be Internal PLL clock or external high speed crystal oscillator(FXOSC). Clock source Select is based on the PLEN bit of SYNCR register. If the PLEN bit is set then internal PLL is the system clock source, otherwise the system clock source is external high speed crystal oscillator(FXOSC).

### 9.4.1. Low-Power Options

#### 9.4.1.1. 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.

#### 9.4.1.2. Stop Mode

In stop mode, all system clocks are disabled.

CAUTION: Do not program or erase EFLASH during stop mode.



## 9.5. Memory Map and Registers

The clock programming module consists of below registers:

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

This subsection describes the memory map and registers of the Clock Module. The Clock Module base address is **0x4003\_0000**. **Table 9-1** shows the offset address of Clock registers.

### 9.5.1. Memory Map

**Table 9-1: Clock Memory Map**

Address	Bits[31:0]	Access <sup>1</sup>
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 Clock Source Enable Control Register(ECSECR)	S
0x0014	OSC Bist Test Configuration Register1(OBTCR1)	S
0x0018	OSC Bist Test Configuration Register2(OBTCR2)	S
0x001C	OSC Bist Test Control Register(OBTCCR)	S
0x0020	OSC BIST Test Counter Register(OBTCNTR)	S
0x0024	OSC BIST Test Result Register(OBTRR)	S

**Note:**

S = supervisor-only access. Accessing supervisor only address locations in user mode has no effect and result in a cycle termination transfer error.

### 9.5.2. Register Description

This subsection provides a description of the clock module registers.

#### 9.5.2.1. Synthesizer Control Register (SYNCR)

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

Address: **CLOCK\_BASEADDR+0x0000\_0000**

	31	30	29	28	27	26	25	24
Read:	SYNCTEST[1:0]		SYSRAM3 LPEN	SYSRAM2 LPEN	SYSRAM1 LPEN	SYSRAM0 LPEN	PLLOCKM	ENLOWPOWER
Write:	[ ]		[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
Reset:	0	0	0	0	0	0	0	1
	23	22	21	20	19	18	17	16
Read:	PLLDIV[5:0]						0	0
Write:	[ ]						[ ]	[ ]
Reset:	0	0	0	0	0	1	0	0
	15	14	13	12	11	10	9	8
Read:	ADCDIV[3:0]				LOSCEN	PLLSRCEN	PLLEN	SLEEP
Write:	[ ]				[ ]	[ ]	[ ]	[ ]
Reset:	0	0	1	0	1	1	0	0
	7	6	5	4	3	2	1	0
Read:	DISPLAYR AMLPEN	CLKOUTS EL	STBYMD[1:0]		CACHERA MLPEN	ADCEN	0	LOSCLPEN
Write:	[ ]	[ ]	[ ]		[ ]	[ ]	[ ]	[ ]
Reset:	1	0	1	1	1	1	0	1

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

Figure 9-2: Synthesizer Control Register (SYNCR)

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

The writable bits of SYNCR register can not be changed, unless the correct sequence is written. The right sequence is: 2'b01 → 2'b10 → 2'b11. After these two bits are written by this sequence, these two bits' value == 2'b11, then the writable bit of SYNCR register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

#### SYSRAM3LPEN — System RAM3 (address range from 0x0083\_0000 to 0x83\_FFFF) Low Power Enable

The low power mode of System RAM3 is achieved by switch-off the power supply to the RAM and the content of the RAM will be lost.

- 1 = Low power mode of SYSRAM3 is enabled when the chip enters low power mode.
- 0 = Low power mode of SYSRAM3 is disabled when the chip enters low power mode.

#### SYSRAM2LPEN — System RAM2 (address range from 0x0082\_0000 to 0x82\_FFFF) Low Power Enable

The low power mode of System RAM2 is achieved by switch-off the power supply to the RAM and the content of the RAM will be lost.

- 1 = Low power mode of SYSRAM2 is enabled when the chip enters low power mode.
- 0 = Low power mode of SYSRAM2 is disabled when the chip enters low power mode.

**SYSRAM1LPEN** — System RAM1(address range from 0x0081\_0000 to 0x81\_FFFF) Low Power Enable  
 The low power mode of System RAM1 is achieved by switch-off the power supply to the RAM and the content of the RAM will be lost.

- 1 = Low power mode of SYSRAM1 is enabled when the chip enters low power mode.
- 0 = Low power mode of SYSRAM1 is disabled when the chip enters low power mode.

**SYSRAM0LPEN** — System RAM0(address range from 0x0080\_0000 to 0x80\_FFFF) Low Power Enable  
 The low power mode of System RAM0 is achieved by switch-off the power supply to the RAM and the content of the RAM will be lost.

- 1 = Low power mode of SYSRAM0 is enabled when the chip enters low power mode.
- 0 = Low power mode of SYSRAM0 is disabled when the chip enters low power mode.

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

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

**ENLOWPOWER** — Enable Enter Low power mode status Bit

Before system enters standby mode, be sure that this bit is set, otherwise the low power mode will not be entered successfully. This bit is set after recovery from low power mode, and cleared by hardware when entering low power mode

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

**PLLDIV[5:0]** — PLL Clock Divider

This field sets the divider value for PLL clock. The default value is 6'b000001 (divide by two). See **Table 9-2** for other divider values.

**Table 9-2: PLL Clock Divider**

PLLDIV[5:0]	Divider Value
000000	Divide-by-2
000001	Divide-by-2
000010	Divide-by-4
000011	Divide-by-6
000100	Divide-by-8
000101	Divide-by-10
.....	.....
.....	.....
111111	Divide-by-126

**Note:** The frequency of the system clock should not be greater than 300MHz.

**ADCDIV[3:0]** — ADC Clock Divider

This field sets the divider value for ADC clock. The default value is 4'b0000(divide by one). See **Table 9-3** for other divider values..

**Table 9-3: ADC Clock Divider**

ADCDIV[3:0]	Divider 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** — This bit determines whether system clock will be stopped or not when PLEN is changed from 0 to 1.

- 1 = System clock will not 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 the same case as the stop instruction because the stop instruction is always valid.

**DISPLAYRAMLPEN** — Display RAM(address range from 0x0084\_0000 to 0x8b\_FFFF) Low Power Enable  
The low power mode of Display RAM is achieved by switch-off the power supply to the RAM and the content of the RAM will be lost.

- 1 = Low power mode of DISPLAYRAM is enabled when the chip enters low power mode.
- 0 = Low power mode of DISPLAYRAM is disabled when the chip enters low power mode.

**CLKOUTSEL** — Clock Out Select Bit

**Table 9-4: CLKOUTSEL Mode**

CLKOUTSEL	CLKOUT
0	System clock
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 9-5**.

**Table 9-5: Sleep Operation Control Bit in Sleep Mode**

STBYMD	ADC Clock	System Clocks	Clock Source	LDO
00	Enable	Disabled	Enable	Normal
01	Disabled	Disabled	Enable	Normal
10	Disabled	Disabled	Disabled	Normal
11	Disabled	Disabled	Disabled	Standby

**CACHERAMPLPEN** — CACHE RAM Low Power Enable Bit.

This bit is enabled when the chip enters low power mode. The low power mode of CACHE RAM is achieved by switch-off the power supply to the RAM and the content of the RAM will be lost.

- 1 = Low power mode of CACHE RAM is enabled when the chip enters low power mode.
- 0 = Low power mode of CACHE RAM is disabled when the chip enters low power mode.

**ADCEN** — Analog-to-digital converter Clock Enable Bit

- 1 = ADC Clock is enable
- 0 = ADC Clock is disable

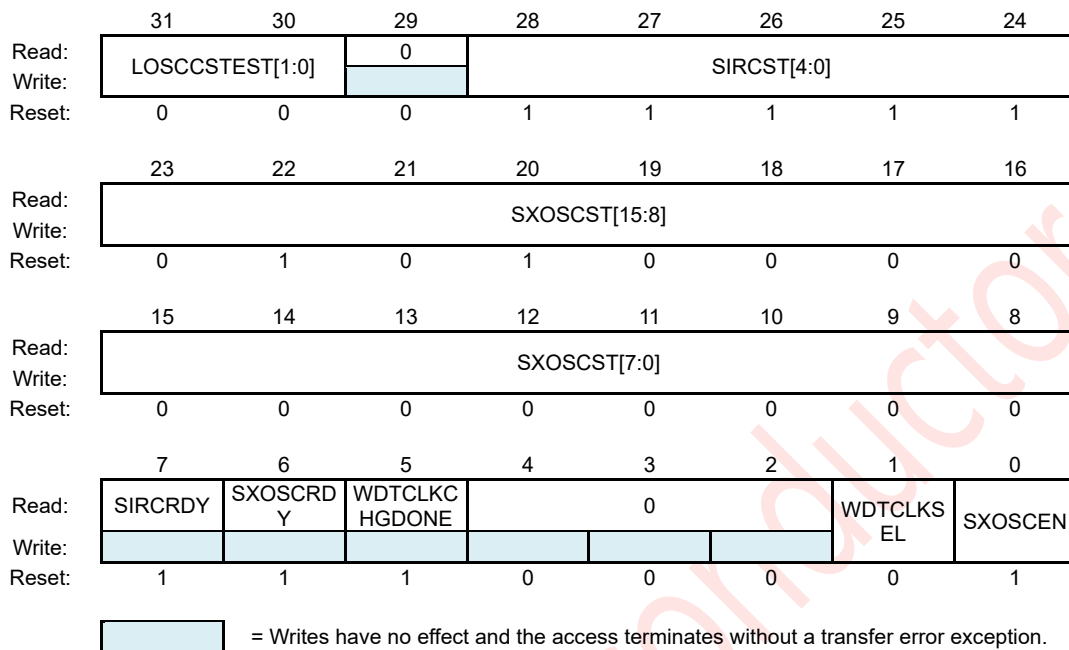
**LOSCLPEN** — Internal Low Speed 128KHz Oscillator Low Power Enable

If the LOSCLPE is set, the internal low speed 128KHz oscillator will be stopped during the 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

**9.5.2.2. Low Speed Oscillator Control and Status Register (LOSCCSR)**

**Address: CLOCK\_BASEADDR+0x0000\_0004**



**Figure 9-3: Low Speed Oscillator Control and Status Register (LOSCCSR)**

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

The writable bit of LOSCCSR register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by the sequence, these two bits' value == 2'b11, then the writable bit of LOSCCSR register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

**SIRCST[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 be ready for output clock after switching 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 be ready for output clock after switching on.

**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).

- 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 changed to low when WDTCLKSEL is changed. When WDTCLK change is done, then this bit will be set.

- 1 = WDT clock switch done and WDT can work normally
- 0 = WDT clock is switching and WDT can not be work normally

**WDTCLKSEL** — WDT clock selection control bit.

When changing WDT clock source from SXOSC to SIRC, the WDTCLKSEL bit should be cleared before turning off SXOSC.

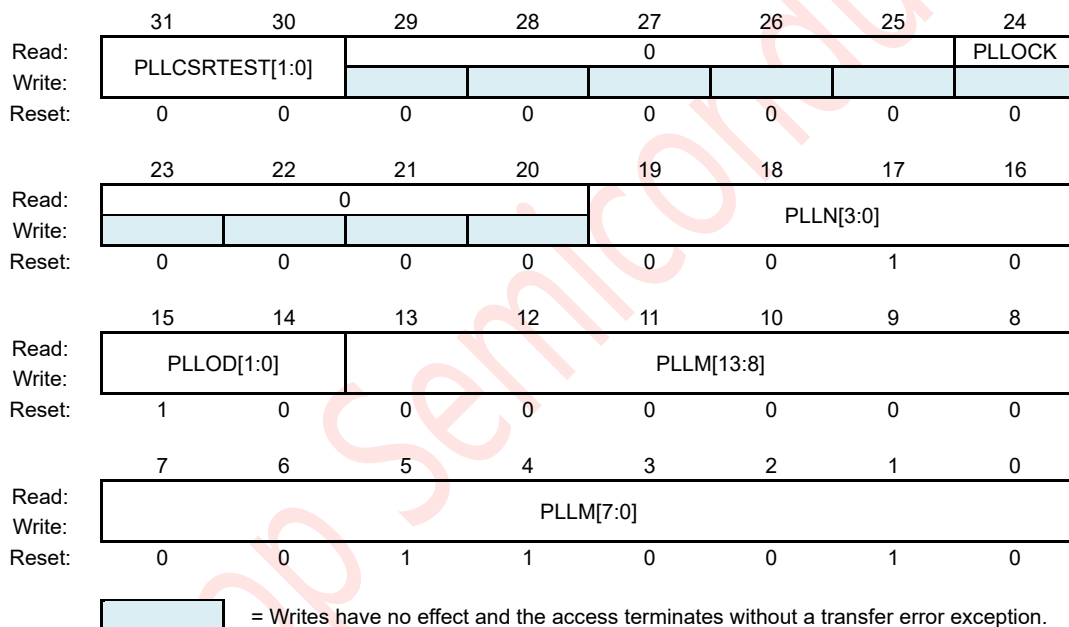
- 1 = WDT clock source is SXOSC(32.768KHz)
- 0 = WDT clock source is SIRC 128Khz

**SXOSCEN** — SXOSC Enable Setting

- 1 = SXOSC is enabled for WDT application.
- 0 = SXOSC crystal is disabled for WDT application., In addition, WDTCLKSEL should be cleared now, otherwise WDT clock will be lost.

**9.5.2.3. PLL Configuration And Status Register (PLLCSR)**

**Address: CLOCK\_BASEADDR+0x0000\_0008**



**Figure 9-4: PLL Configuration and Status Register (PLLCSR)**

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

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

The writable bit of PLLCSR register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of PLLCSR register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

CAUTION: Please keep these conditions during usage:

1. 1MHz <= XIN/N <= 50MHz
2. 200MHz <= XIN\*M/N <= 400MHz
3. M≥4
4. N≥1

**PLLOCK** — PLL Lock Flag

Once PLL is enabled, PLLOCK will be set after waiting for the number of cycles set in PLLOCKCR[PLLST].

1 = PLL is locked

0 = PLL is not locked

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

This field sets the input divider value for PLL clock. The default value is 4'b0000. See **Table 9-6** for the available divider values.

**Table 9-6: PLL Input Divider**

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

**PLLOD[1:0]** — Output divider control bits

This field sets the output divider value for PLL VCO clock. The default value is 2'b00. See **Table 9-7** for the available divider values.

**Table 9-7: 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

**PLLM[13:0]** — Feedback 14-bits divider control bits

This field sets the PLL feedback divider value . The default value is 14'b0. See **Table 9-8** for the available divider values.



Table 9-8: PLL Feedback Divider Value

PLLM[13:0]	M
14'b00_0000_0000_0000	0
14'b00_0000_0000_0001	1
14'b00_0000_0000_0010	2
14'b00_0000_0000_0011	3
14'b00_0000_0000_0100	4
14'b00_0000_0000_0101	5
.....	.....
.....	.....
14'b11_1111_1111_1111	16383

9.5.2.4. Module Stop Control Register (MSCR)

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

Address: CLOCK\_BASEADDR+0x0000\_000C

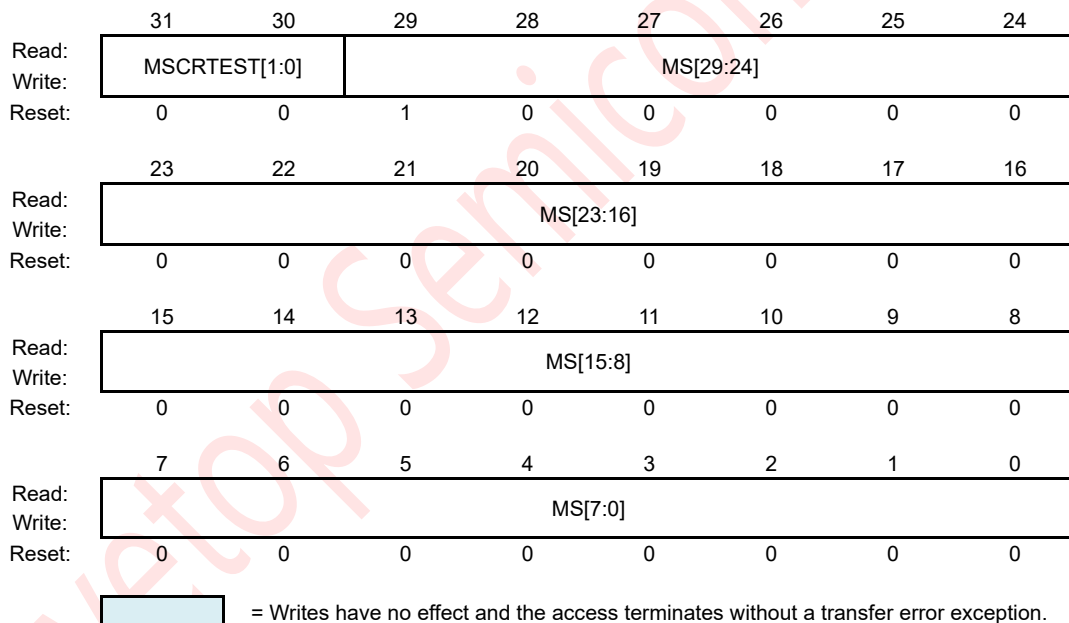


Figure 9-5: Module Stop Control Register (MSCR)

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

The writable bit of MSCR register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of MSCR register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values 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. (refer to **Table 9-9** MS[29:0] Bits Corresponding Modules).

1 = Module Clock is disabled

0 = Module Clock is enabled

**Table 9-9: MS[29:0] Bits Corresponding Modules**

MS Bit	Corresponding Module
0	Blender
1	COMP0
2	COMP1
3	ADC
4	PIT0
5	PIT1
6	PIT2
7	PIT3
8	RTC
9	DMA
10	PWM0
11	PWM1
12	EPORT0
13	EPORT1
14	XBAR
15	OPTION
16	RESET
17	WDT
18	SCI0
19	CCM
20	I2C
21	SCI1
22	SCI2
23	CAN
24	EPORT2
25	QSPI0
26	QSPI1
27	QSPI2
28	RGB
29	USBC

9.5.2.5. EPT External Clock Source Enable Control Register (ECSECR)

Address: CLOCK\_BASEADDR+0x0000\_0010

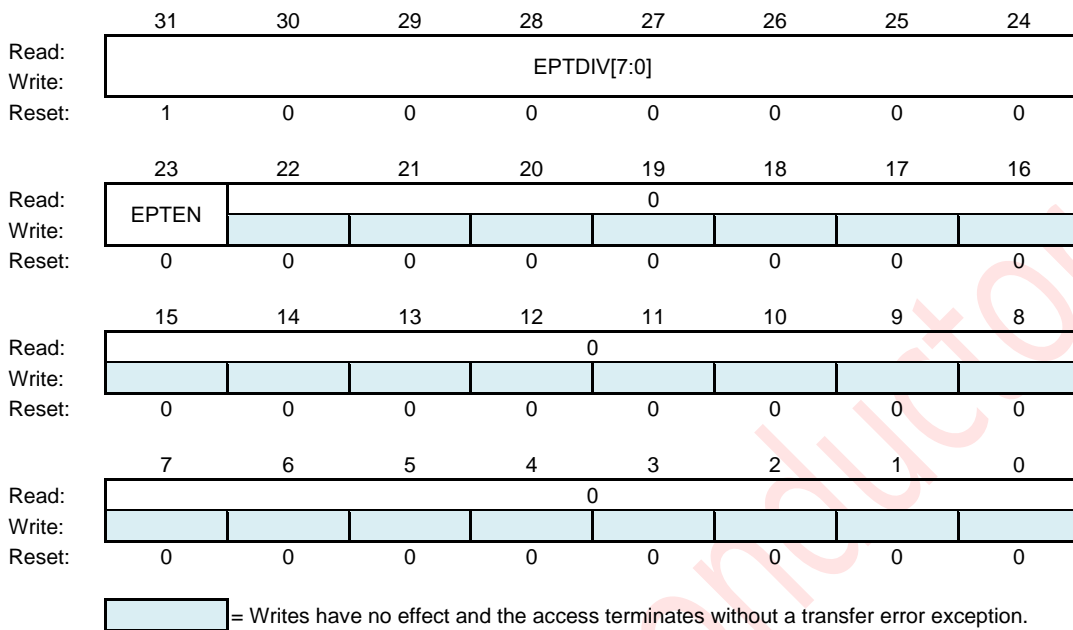


Figure 9-6: EPT External Clock Source Enable Control Register (ECSECR)

**EPTDIV[7:0]** — EPT Clock Divider

This field sets the divider value for EPT clock. The default value is 8'h80. See Table 9-10 for other available divider values.

Table 9-10: EPT Clock Divider

EPTDIV[7:0]	Divider 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

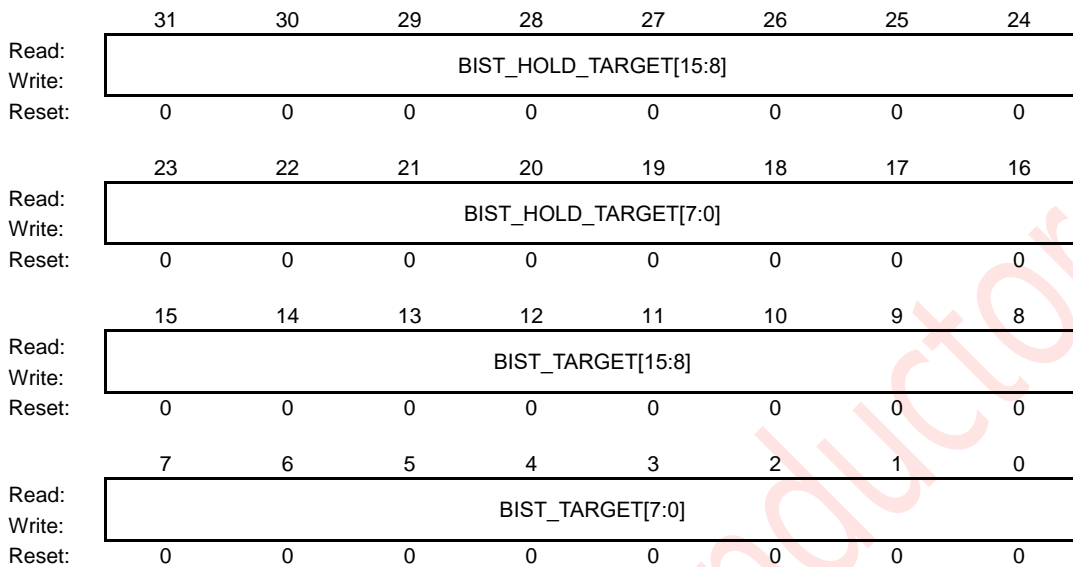
**EPTEN** — EPT Clock Enable Bit

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

- 1 = EPT clock is enable
- 0 = EPT clock is disable

**9.5.2.6. OSC Bist Test Configuration Register1 (OBTCR1)**

**Address: CLOCK\_BASEADDR+0x0000\_0014**



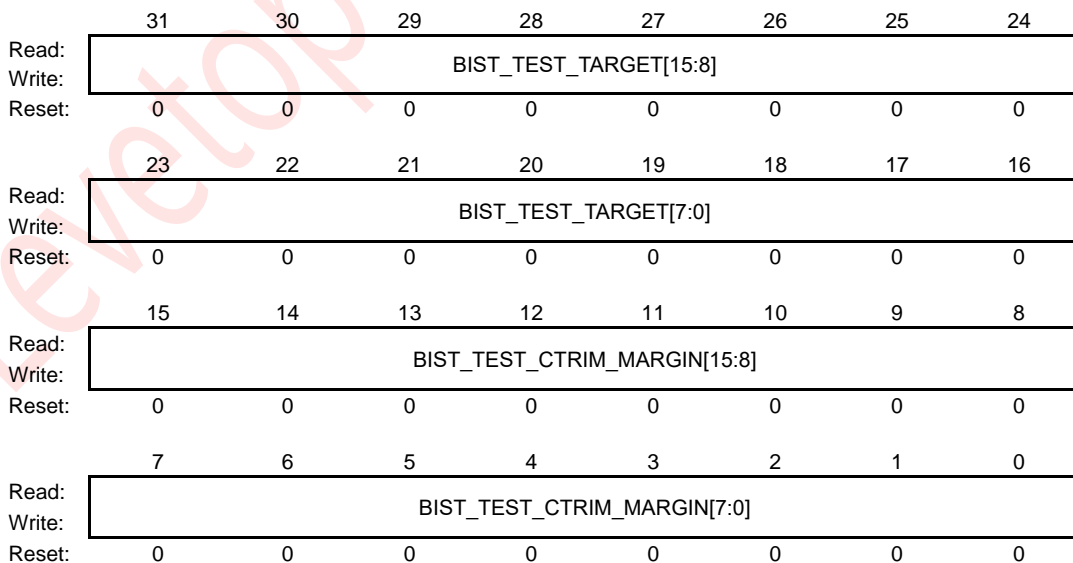
**Figure 9-7: OSC Bist Test Configuration Register1 (OBTCR1)**

**BIST\_HOLD\_TARGET[15:0]** — Bist clk hold count target value  
 These bits set the the amount of wait clk under test stable after the trim value is changed.

**BIST\_TARGET[15:0]** — Bist clk count target value.

**9.5.2.7. OSC Bist Test Configuration Register2 (OBTCR2)**

**Address: CLOCK\_BASEADDR+0x0000\_0018**



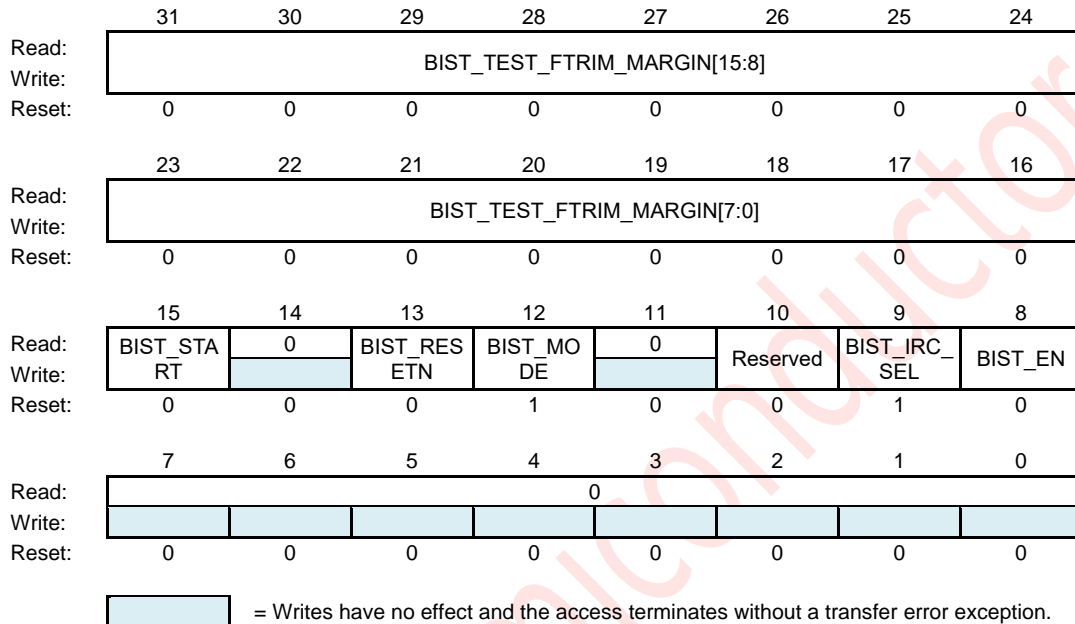
**Figure 9-8: OSC Bist Test Configuration Register2 (OBTCR2)**

**BIST\_TEST\_TARGET[15:0]** — Reserved. Not available to users.

**BIST\_TEST\_CTRIM\_MARGIN[15:0]** — Reserved. Not available to users.

**9.5.2.8. OSC Bist Test Control Register (OBTCTLR)**

**Address: CLOCK\_BASEADDR+0x0000\_001C**



**Figure 9-9: OSC Bist Test Control Register (OBTCTLR)**

**BIST\_TEST\_FTRIM\_MARGIN[15:0]** — Reserved. Not available to users.

**BIST\_START** — Bist Start

- 1 = start
- 0 = stop

**BIST\_RESETN** — Bist Reset Negate

- 1 = Negate Bist reset
- 0 = Assert Bist Reset

**BIST\_MODE** — Bist Mode

- 1 = trace mode(for measure the frequency of PLL or 128K clock)
- 0 = trim mode

**Note:** Only trace mode is implemented on this chip.

**BIST\_IRC\_SEL** — PLL or 128KHz clock selection

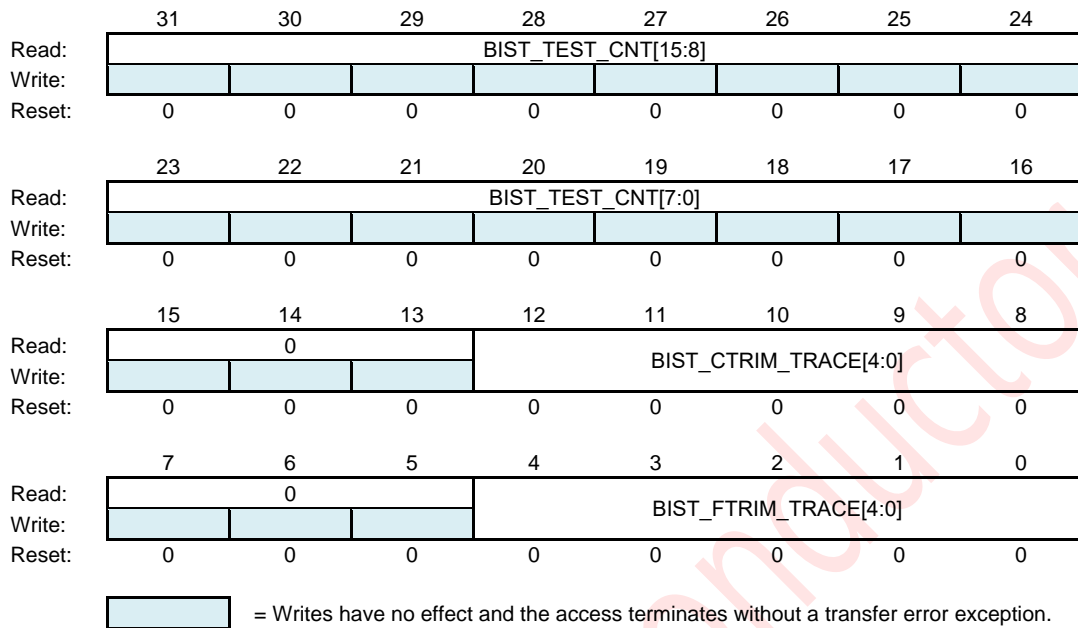
- 1 = PLL Clock is selected
- 0 = 128K Clock is selected

**BIST\_EN** — Reference Clock Enable

- 1 = enable
- 0 = disable

**9.5.2.9. OSC BIST Test Counter Register (OBTCNTR)**

**Address: CLOCK\_BASEADDR+0x0000\_0020**



**Figure 9-10: OSC BIST Test Counter Register (OBTCNTR)**

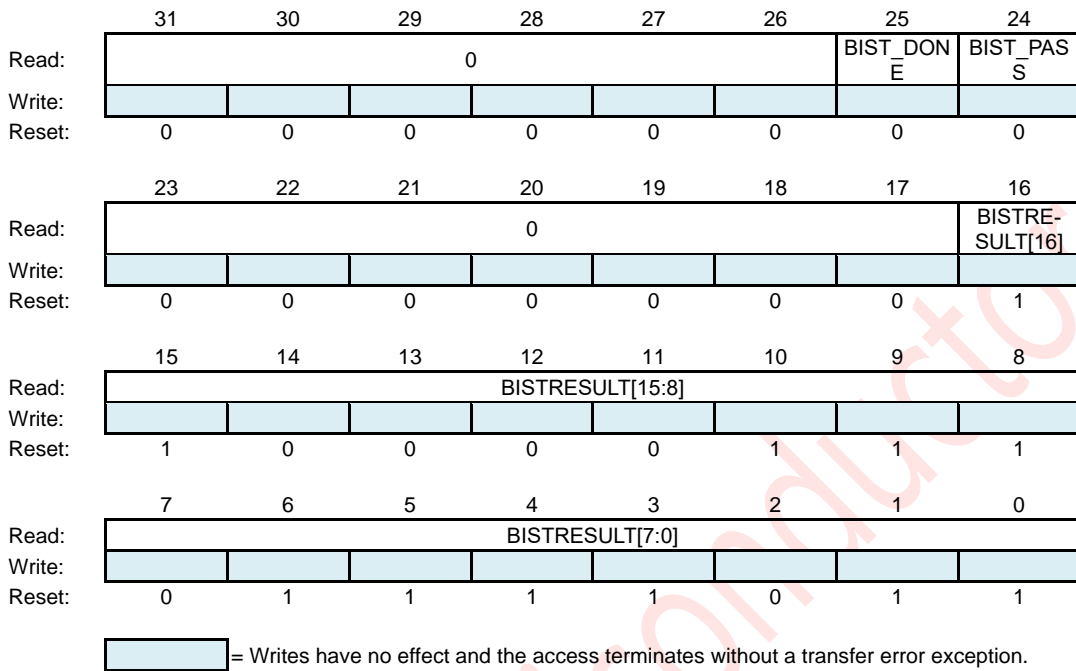
**BIST\_TEST\_CNT[15:0]** — Bist Test Counter Value

**BIST\_CTRIM\_TRACE[4:0]** — Bist Ctrim Trace Value

**BIST\_FTRIM\_TRACE[4:0]** — Bist Ftrim Trace Value

**9.5.2.10. OSC BIST Test Result Register (OBTRR)**

**Address: CLOCK\_BASEADDR+0x0000\_0024**



**Figure 9-11: OSC BIST Test Result Register (OBTRR)**

**BIST\_DONE** — Bist Done

- 1 = done
- 0 = not done

**BIST\_PASS** — Bist Pass

- 1 = PASS
- 0 = FAIL

**BISTRESULT[16:0]** — Bist Trim Result

VALID WHEN bist\_done = 1 & bist\_pass = 1

## 9.6. Functional Description

### 9.6.1. Turning on PLL

The following steps are suggested:

1. Configuring SYNCR[PLLSRCEN] first.
2. Configuring PLLCSR[PLLOD]/PLLCSR[PLLN]/PLLCSR[PLLM] according to the target VCO frequency.
3. Configuring SYNCR[PLEN] and open PLL.
4. Waiting for SYNCR[PLLOCK] status and system clock will be changed to PLL clock.
5. Configuring SYNCR[PLLDIV] according to the target system frequency.

### 9.6.2. The Frequency of PLL Measurement

The following steps are suggested:

1. Setting BIST\_RESETN bit to 0 for assert reset.
2. Setting BIST\_TARGET counter value.
3. Setting BIST\_MODE bit to 1.
4. Setting IRC\_BIST\_SEL bit to 1 for PLL measurement.
5. Setting BIST\_START bit to 1.
6. Setting BIST\_RESETN bit to 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:

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

### 9.6.3. The Frequency of 128KHz Measurement

The following steps are suggested:

1. Setting BIST\_RESETN bit to 0 for assert reset.
2. Setting BIST\_TARGET counter value.
3. Setting BIST\_MODE bit to 1.
4. Setting IRC\_BIST\_SEL bit to 0 for 128KHz clock measurement.
5. Setting BIST\_START bit to 1.
6. Setting BIST\_RESETN bit to 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:

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



## 10. Reset Control Module (RCM)

### 10.1. Overview

Reset Control Module is provided to determine the cause of reset, assert the appropriate reset signals to the system, and then to keep a history of reset causes.

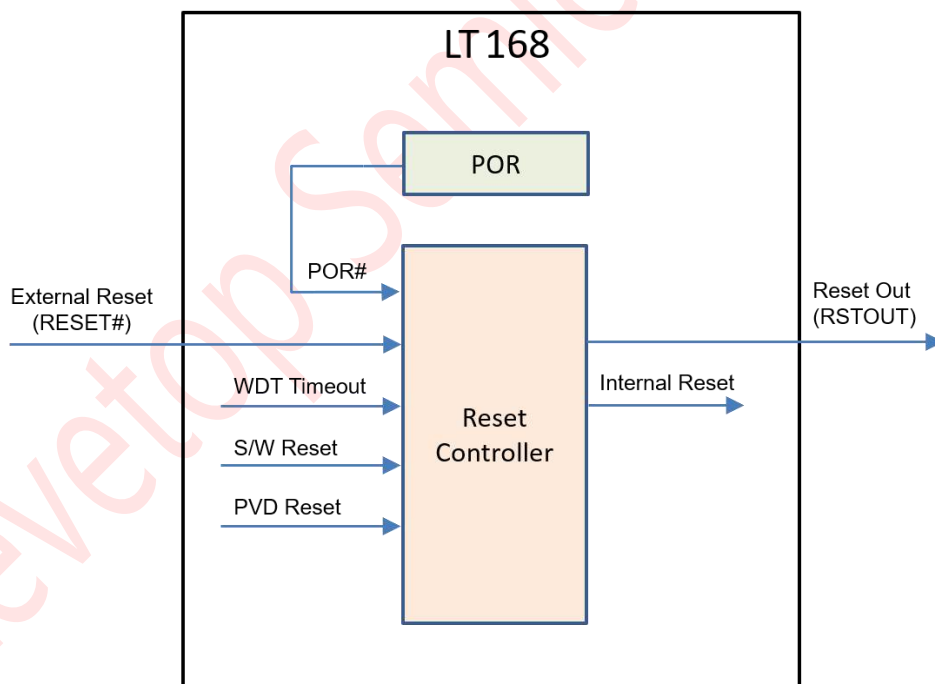
### 10.2. Features

Module features include:

- Five Sources of Reset:
  - Power on Reset
  - External Reset Pin (RESET#)
  - Software Reset
  - Watchdog Timer Reset
  - Programmable Voltage Detect Reset
- Software-readable status flags indicating the cause of the last reset

### 10.3. Block Diagram

Figure 10-1 illustrates the reset controller



**Figure 10-1: Reset Controller Block Diagram**

## 10.4. Memory Map and Registers

### 10.4.1. Memory Map

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

This subsection describes the memory map and registers of Reset Controller Module. The Reset Controller base address is 0x4002\_0000. See Table 10-1 for the address map and the following paragraphs for a description of the registers.

Table 10-1: Reset Controller Address Map

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

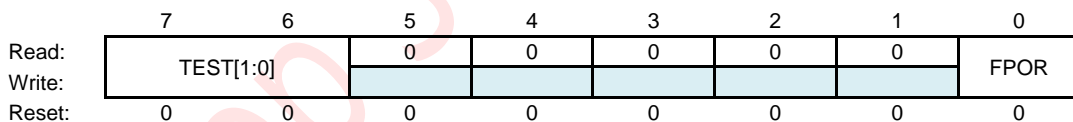
Note: S/U = Supervisor or user mode access.

### 10.4.2. Register Description

#### 10.4.2.1. Reset Test Register (RTR)

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

Address: RCM\_BASEADDR+0x0000\_0001




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

Figure 10-2: Reset Test Register (RTR)

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

The writable bit of FPOR register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01->2'b10->2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of FPOR bit can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

#### FPOR — Force Power On Reset

Writing 0x5B to the RTR register, and then setting this bit will result in system power on reset.

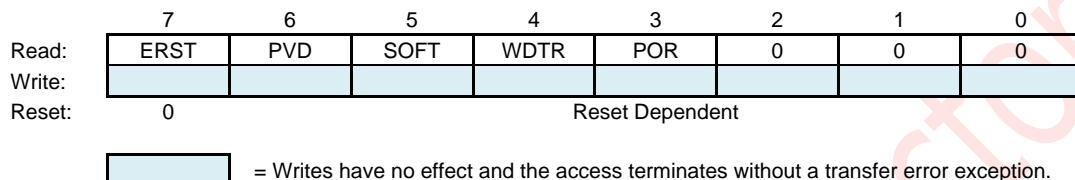
The reset will result in the chip trimming again.

**10.4.2.2. Reset Status Register (RSR)**

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.

**Address: RCM\_BASEADDR+0x0000\_0002**



**Figure 10-3: 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 a PVD reset.

- 1 = Last reset state was caused by a PVD reset
- 0 = Last reset state was not caused by a 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.

**WDTR** — Watchdog Timer Reset Flag

This bit indicates that the last reset state was caused by a watchdog timer timeout.

- 1 = Last reset state was caused by a watchdog timer timeout.
- 0 = Last reset state was not caused by a watchdog timer timeout.

**POR** — Power-On Reset Flag

This bit indicates that the last reset state was caused by power-on reset.

- 1 = Last reset state was caused by power-on reset.
- 0 = Last reset state was not caused by power-on reset.

10.4.2.3. Reset Control Register (RCR)

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

Address: RCM\_BASEADDR+0x0000\_0003

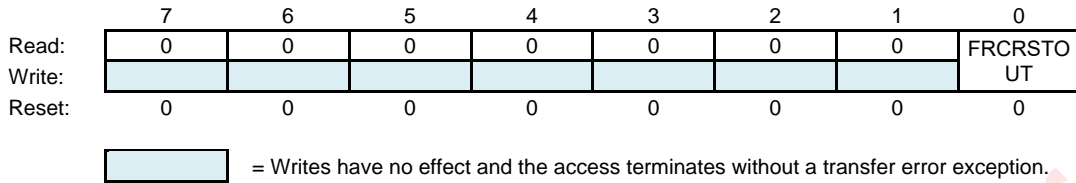


Figure 10-4: Reset Control Register (RCR)

**FRCRSTOUT** — Force RSTOUT Pin

The FRCRSTOUT bit allows software to drive the external RSTOUT pin to Low or High State.

1 = Assert RSTOUT pin. Writing “1” to this bit will drive RSTOUT pin to Low.

0 = Negate RSTOUT pin. Writing “0” to this bit will drive RSTOUT pin to High.

10.5. Functional Description

10.5.1. Reset Sources

Table 10-2 defines the sources of reset and the signals driven by the reset controller.

Table 10-2: 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, and 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.

10.5.1.1. 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.

10.5.1.2. Watchdog Timer Reset

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

**10.5.1.3. Software Reset**

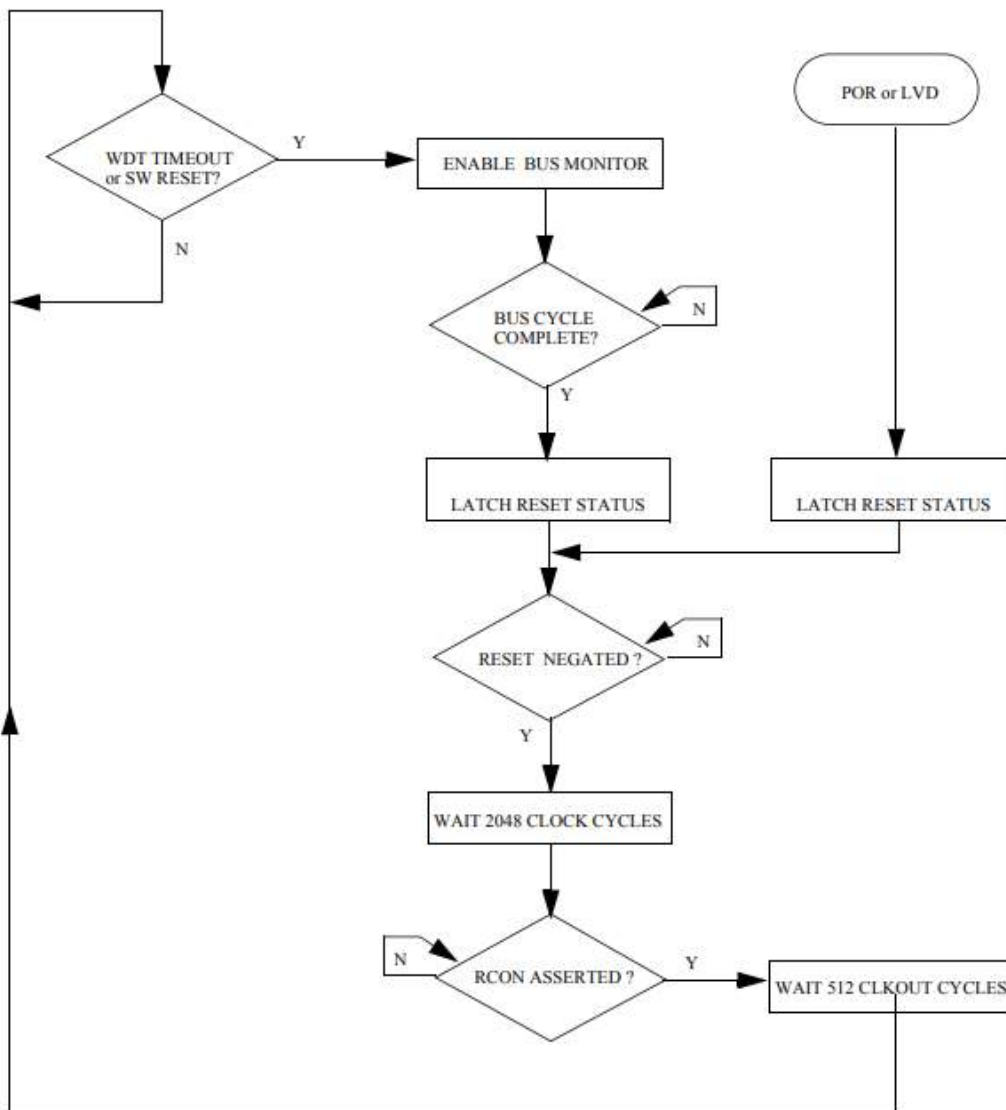
If the SYSRESTEQ bit in 32-bits RISC Embedded Interrupt Controller (EIC) is set, the software reset will be generated. The reset controller asserts system reset for approximately 2048 cycles. Then the chip exits reset and resumes operation.

**10.5.1.4. 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.

**10.5.2. Reset Control Flow**

The reset logic control flow is shown in **Figure 10-5**. All cycle counts given are approximate.



**Figure 10-5: Reset Control Flow**

## **11. Static Random Access Memory (SRAM)**

### **11.1. Introduction**

The features of the LT168's Static Random Access Memory (SRAM) include:

- On-chip 64K Bytes SRAM
- Fixed Address Space
- Byte, Half-word (16-bits), or Word (32-bits) Read/Write Accesses
- One Clock Per access (including bytes, Half-words, and Words)
- Supervisor or User Mode Access

### **11.2. Modes Of Operation**

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

### **11.3. Low-Power Modes**

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

### **11.4. 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.

### **11.5. Interrupts**

The SRAM module does not generate interrupt requests.

## 12. Cache Module (CACHEM)

### 12.1. Introduction

Cache Module provides the processor with tightly-coupled processor-local memories and bus paths to EBI and QSPI0/1/2 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 reduce 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 accessing to the adjacent locations (for example, sequential instruction execution or the utilization of a data structure).
- Temporal locality — An access to an area of memory is likely to be repeated within a short period of time (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 are loaded into a cache, the access time for subsequent loads and stores are reduced. This will enhance overall performance. 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 32-bits AHB-bus based application that desires a cache function. This cache function can enhance performance by providing fast access to recently used code or data.

The local memory controller supports three operation modes:

1. Write-through — access to address spaces with this cache mode is 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 is 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.

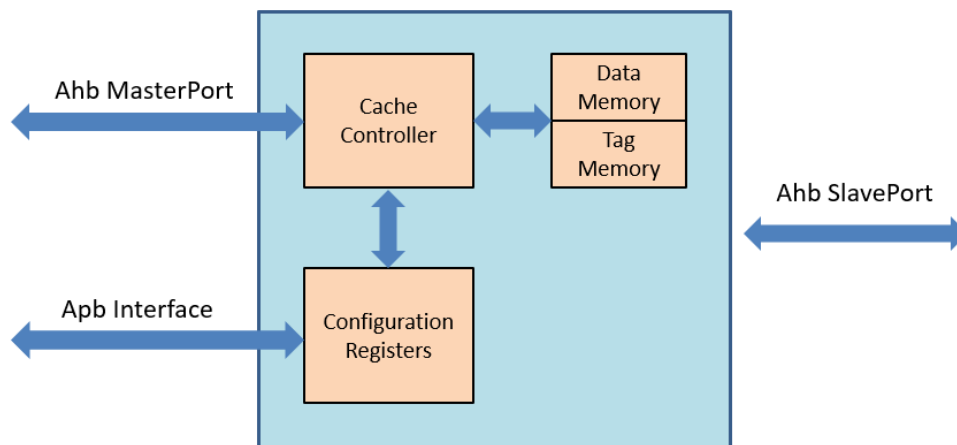
- A write-back write miss will do a "read-to-write" (allocate on write miss policy for write-back mode spaces). A line read on the output bus of a 16 byte aligned memory address containing the desired write address is performed. This miss data is loaded into the cache and marked as valid and modified; and the write data will then update the appropriate cache data locations.
3. Non-cacheable — access to address spaces with this cache mode is 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 decode 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 32-bits 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 32-bits wrapping burst accesses with the critical word accessed first for maximum performance.

## 12.2. Block Diagram

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



**Figure 12-1: Cache Module Block Diagram**



### 12.3. Memory Map and Registers

#### 12.3.1. Memory Map

This subsection describes the Cache Module memory map and registers. The Cache Module base address is 0x4019\_0000. Table 12-1 shows the offset address of these registers.

Table 12-1: Cache Module Memory Map

Address Offset	Bits[31:0]	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

**Notes:**

1. S = CPU supervisor mode access only, U = CPU user mode access only
2. Accessing to supervisor-only address locations in user mode has no effect and result in a cycle termination transfer error

#### 12.3.2. Register Description

This subsection provides a description of the Cache module.

##### 12.3.2.1. Cache Control Register (LMEM\_CCR)

Address: CACHE\_BASEADDR+0x0000\_0000

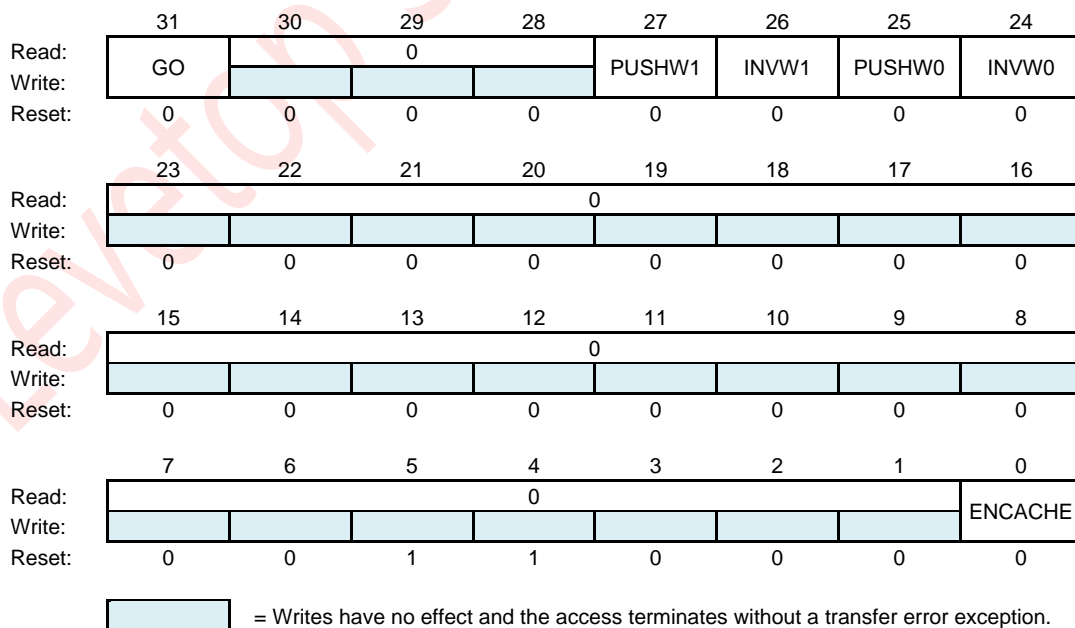


Figure 12-2: Cache Control Register (LMEM\_CCR)

**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.

**PUSHW1** — Push way 1

1 = When setting the GO bit, push all modified lines in way 1

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

**PUSHW0** — Push way 0

1 = When setting the GO bit, push all modified lines in way 0

0 = no operation

**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

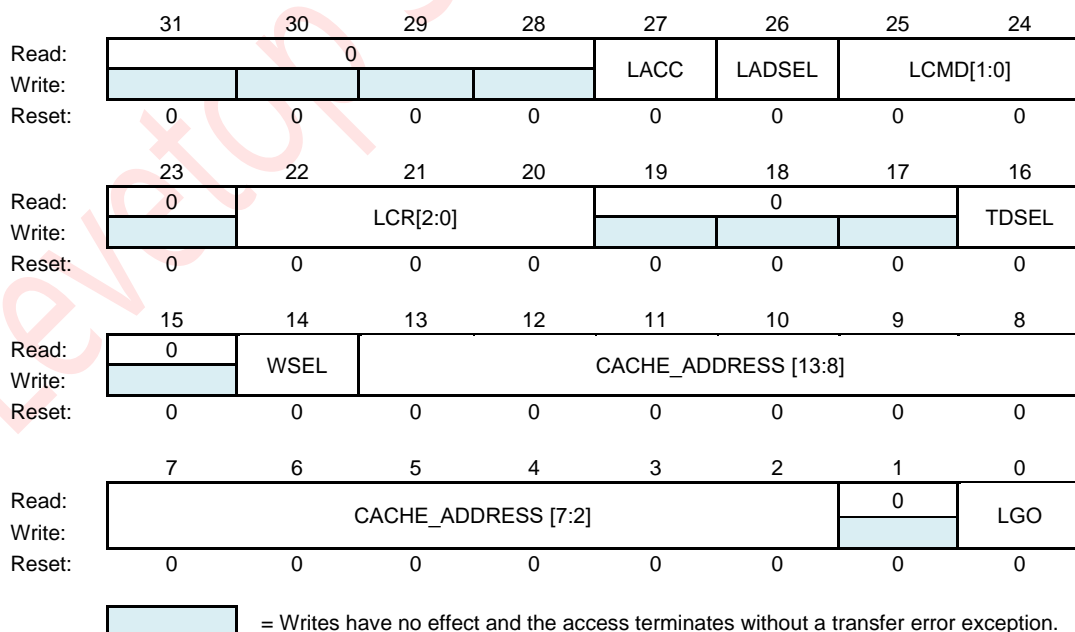
**ENCACHE** — Cache enable.

1 = cache is enabled

0 = cache disabled

**12.3.2.2. Cache Line Control Register (LMEM\_CLCR)**

**Address: CACHE\_BASEADDR+0x0000\_0004**



**Figure 12-3: Cache Line Control Register (LMEM\_CLCR)**

**LACC** — Line access type.

- 0 = Read
- 1 = Write

**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

**LCMD[1:0]** — Line command.

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

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

“

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

- 1 = Tag.
- 0 = Data.

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

- 1 = way 1.
- 0 = way 0.

**CACHE\_ADDRESS[13:2]** — Cache address.

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

**LGO** — Initiate Cache Line Command.

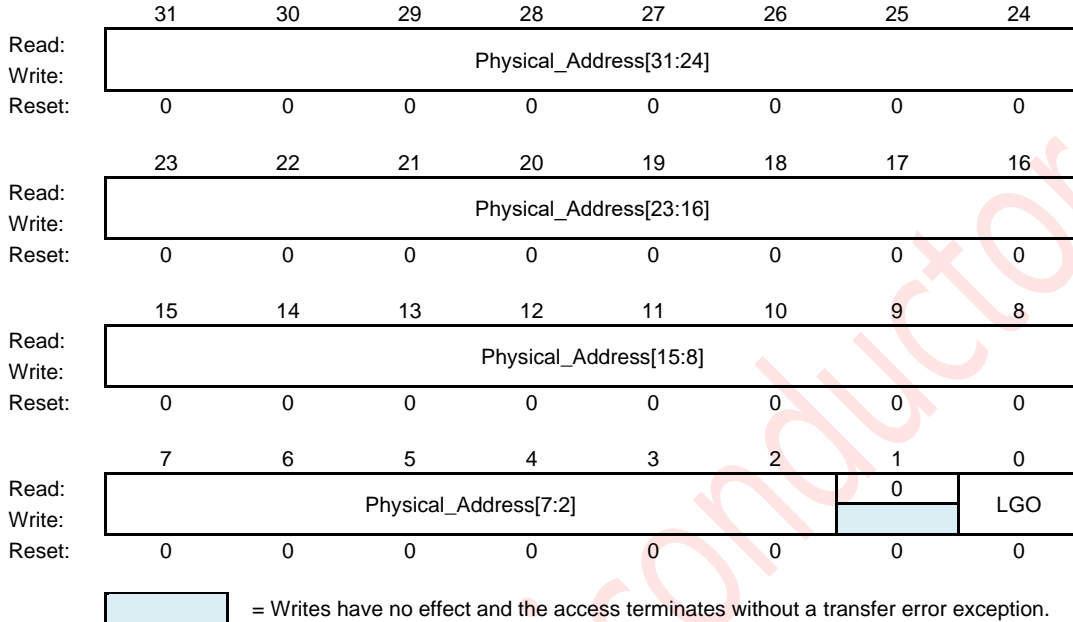
Setting this bit initiates the cache line command indicated by LMEM\_CLCR[27-24]. Reading this bit indicates if a line command is active. This bit stays set until the command completes. Writing zero has no effect. This bit is shared with CSAR[LGO].

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

**12.3.2.3. 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.

**Address: CACHE\_BASEADDR+0x0000\_0008**



**Figure 12-4: 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.

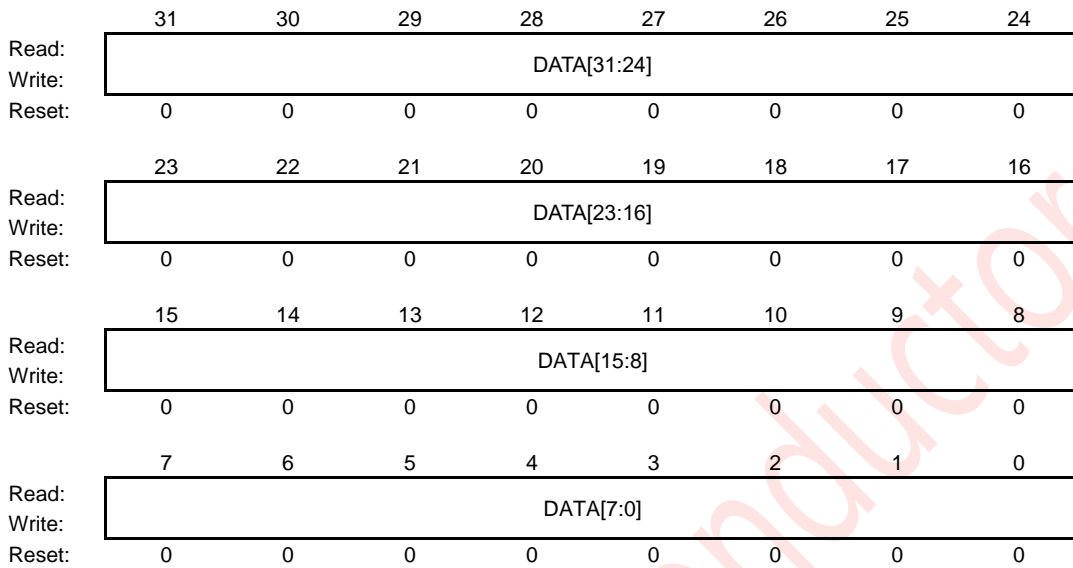
This bit is shared with CLCR[LGO]

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

**12.3.2.4. 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.

**Address: CACHE\_BASEADDR+0x0000\_000C**



**Figure 12-5: Cache Read/Write Value Register (LMEM\_CCVR)**

**DATA[31:0]** — 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

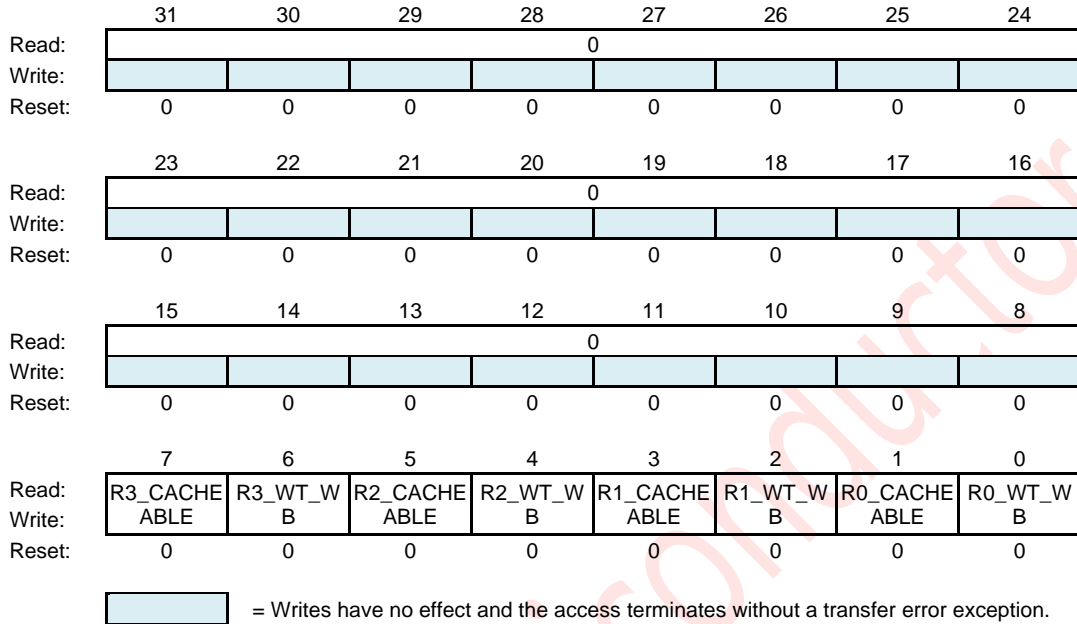
For data search, read, or write:

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

**12.3.2.5. Cache Access Register (LMEM\_ACRG)**

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

**Address: CACHE\_BASEADDR+0x0000\_0020**



**Figure 12-6: 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

12.3.2.6. Cache Page Invalidation Base Address Register (LMEM\_PAGE\_INV\_BADDR)

Address: CACHE\_BASEADDR+0x0000\_0180

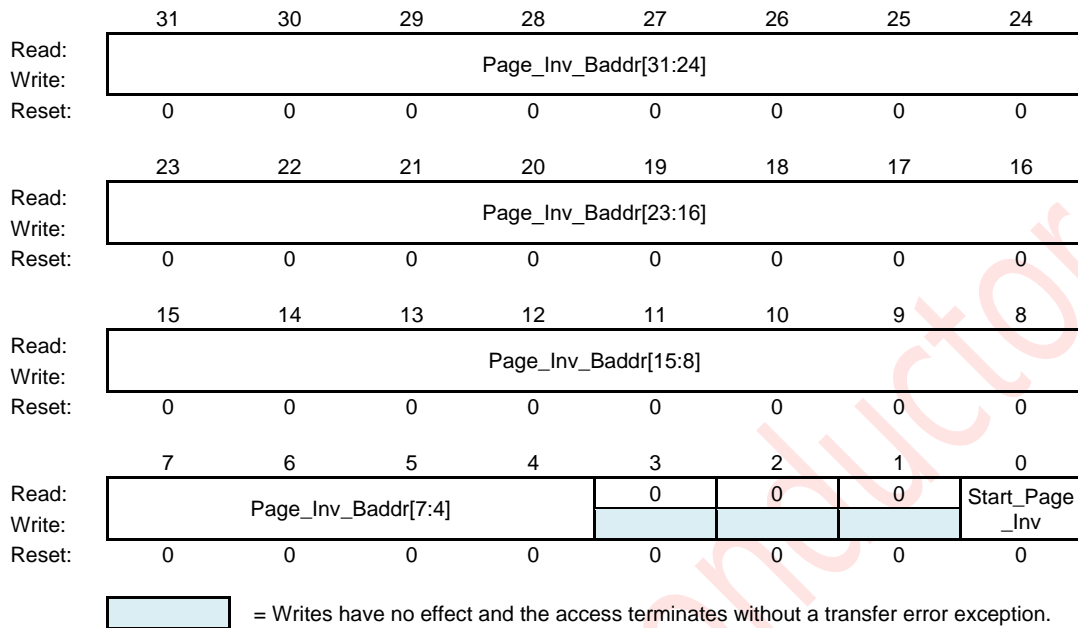


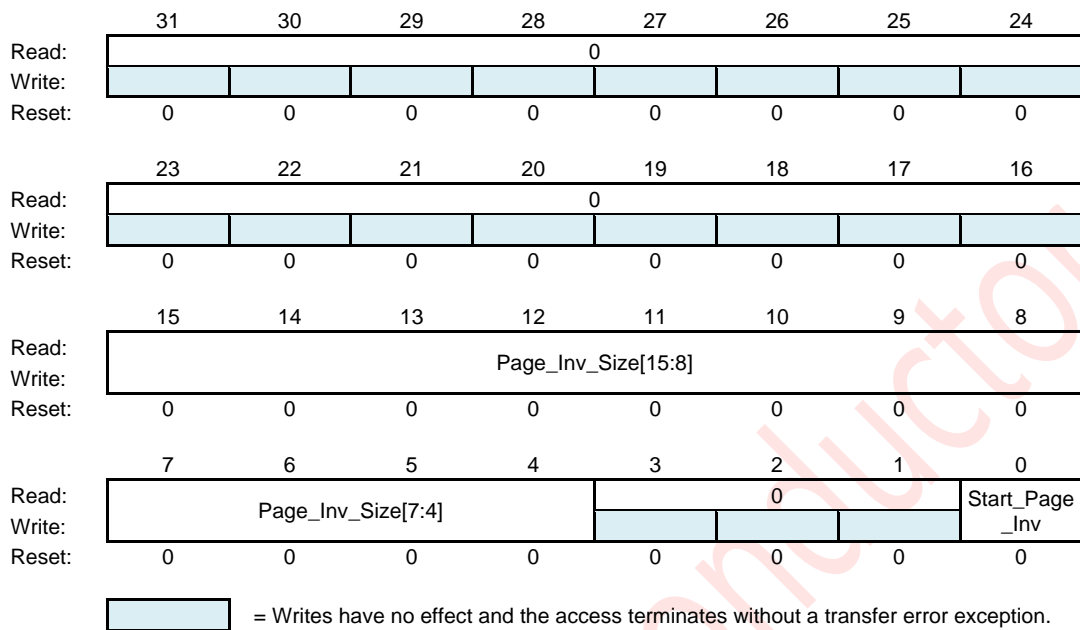
Figure 12-7: Cache Page Invalidate Base Address Register (LMEM\_PAGE\_INV\_BADDR)

**Page\_Inv\_Baddr[31:4]** — cache invalidate start address for system memory. Lower 4-bits 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.

**12.3.2.7. Cache Page Invalidation Base Size Register (LMEM\_PAGE\_INV\_SIZE)**

**Address: CACHE\_BASEADDR+0x0000\_0184**



**Figure 12-8: Cache Page Invalidate Size Register (LMEM\_PAGE\_INV\_SIZE)**

**Page\_Inv\_Size[15:4]** — cache invalidate size for system memory. Lower 4-bits is ignored because of line alignment.

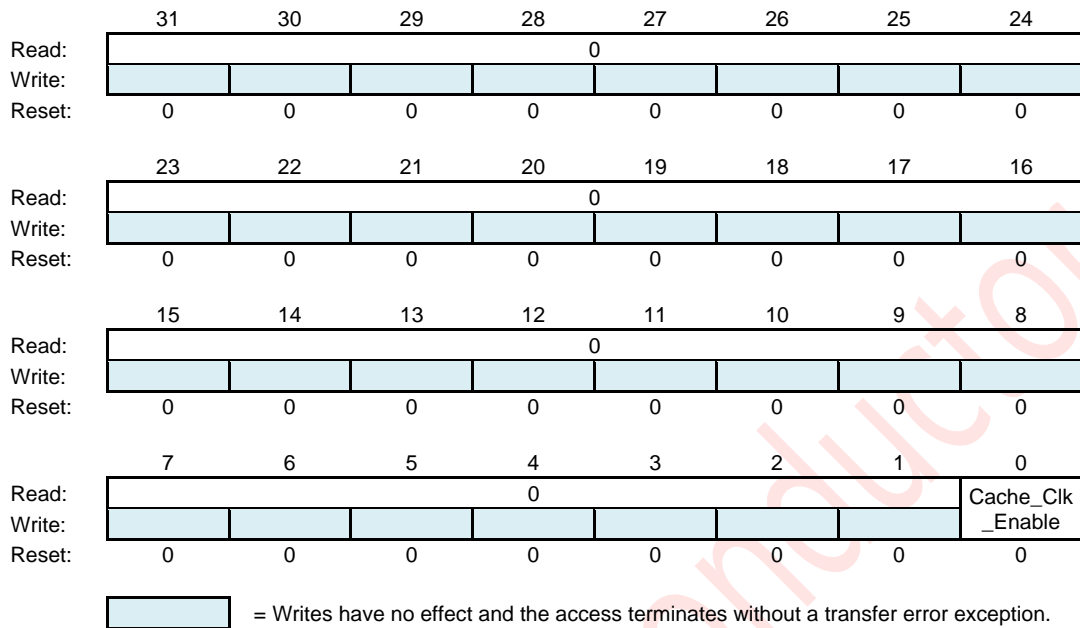
**Start\_Page\_Inv** — start to page invalidate. Cleared if the invalidation is completed.

This bit is shared in bit 0 of LMEM\_PAGE\_INV\_BADDR\_REG.



**12.3.2.8. Cache Clock Enable Register (LMEM\_CACHE\_CLK\_EN)**

**Address: CACHE\_BASEADDR+0x0000\_0188**



**Figure 12-9: Cache Clock Enable Register (LMEM\_CACHE\_CLK\_EN)**

**Cache\_Clk\_Enable** — cache clock is enabled. Must be '1' if cache is enabled.

### 12.4. 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 32K Bytes. The caches have 32-bits address and data paths, and a 16-byte line size. The cache tags and data storage use external single-port, synchronous RAMs.

For these 32K Bytes caches, each cache TAG function uses two 1024 x 20-bits RAM arrays and the cache DATA function uses two 4096 x 32-bits RAM arrays. The cache TAG entries store 18-bits 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:

$$\text{CACHE - 32K Bytes size} = (1024 \text{ sets}) \times (16\text{-byte lines}) \times (2\text{-way set-associative})$$

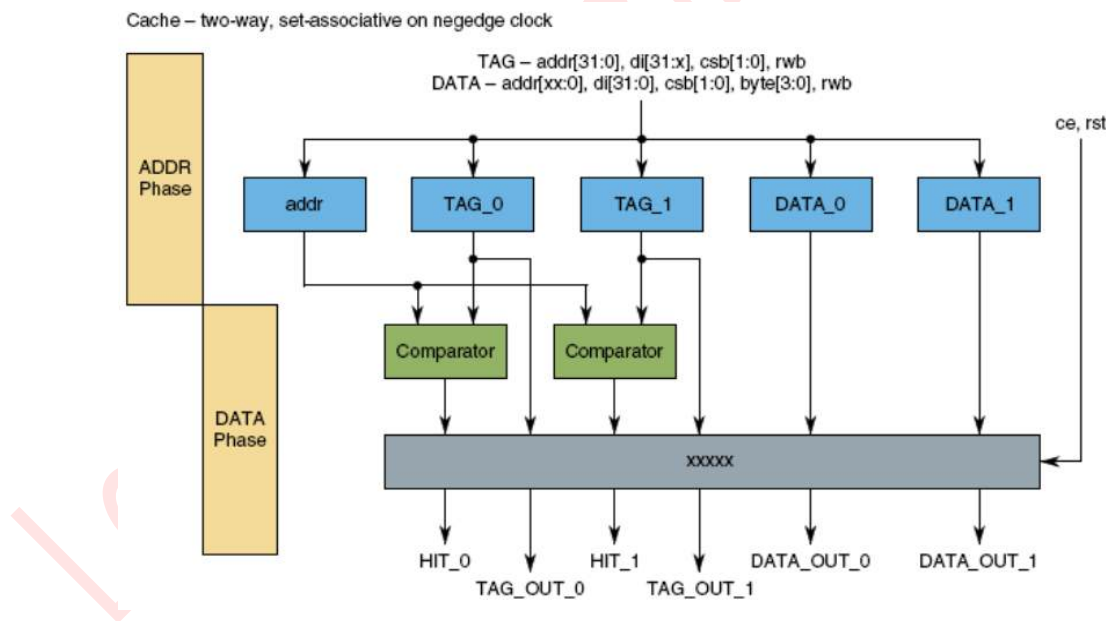
**TAG:**

- 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 32-bits words within a set
- address[1:0] used to select the byte within the 32-bits 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 12-10: 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.

## 12.5. 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.

### 12.5.1. 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 **Table 12-2**. Set commands work as follows:

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

**Table 12-2: Cache Set Commands**

CCR[27:24]				Command
PUSH W1	INW1	PUSH W0	INW0	
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

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

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

**12.5.2. 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]). When using a cache address, the command can be completely specified by the CLCR register. When 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 as 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:

**Table 12-3: Cache Line Commands**

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

**12.5.2.1. 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 the following 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]).

**12.5.2.2. Executing a Series of Line Commands Using Physical Addresses**

Perform a series of line commands with incremental physical addresses by 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 the following 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.

**12.5.2.3. 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.

**Table 12-4: Line Command Results**

CLCR[22:20]			For cache address commands	For physical address commands
LCWY	LCIMB	LCIVB		
0	0	0	Way 0 line 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.	

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.

Levetop Semiconductor

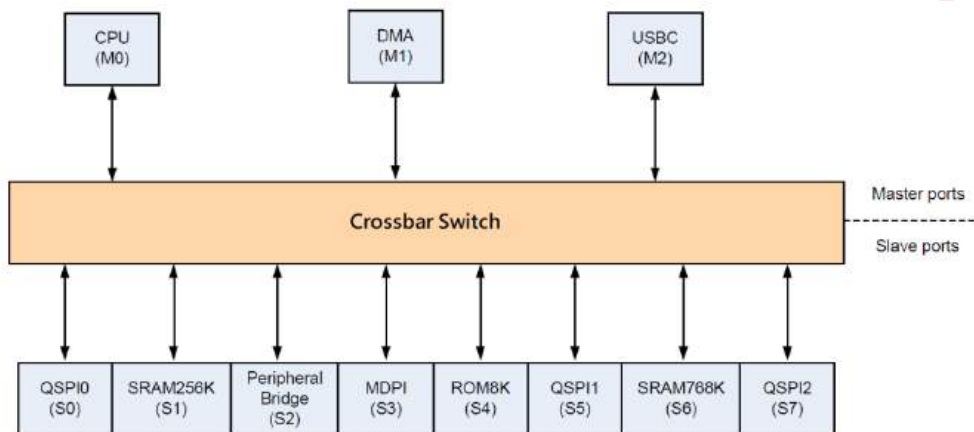
## 13. Crossbar Switch (XBAR)

### 13.1. Introduction

#### 13.1.1. 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 13-1** shows a block diagram of the XBAR.



**Figure 13-1: XBAR Device-Specific Block Diagram**

#### 13.1.2. Features

The crossbar switch includes these distinctive features:

- 3 Master Ports:
  - CPU
  - DMA
  - USBC
- 8 Slave Ports
  - ROM memory controller
  - Two Internal SRAM memory controller
  - Three 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
- 32-bits Address, 32-bits Data Paths
- Fixed Priority Scheme And Fixed Parking Strategy
- Support 8-, 16-, and 32-Bit Single Transfers
- Support Low-Power Park Mode

## 13.2. Memory Map and Registers

This section provides information of XBAR registers whose base address is 0x4017\_0000.

### 13.2.1. Memory Map

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 32-bits 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 13-1**.

**Table 13-1: XBAR Register Configuration Summary**

Address Offset	Bits[31:0]	Access <sup>(1)</sup>
0x0000	Master Priority Register for Slave Port 0(MPR0)	S/U
0x0010	General Purpose Control Register for Slave Port 0(SGPCR0)	S/U
0x0100	Master Priority Register for Slave Port 1(MPR1)	S/U
0x0110	General Purpose Control Register for Slave Port 1(SGPCR1)	S/U
0x0200	Master Priority Register for Slave Port 2(MPR2)	S/U
0x0210	General Purpose Control Register for Slave Port 2(SGPCR2)	S/U
0x0300	Master Priority Register for Slave Port 3(MPR3)	S/U
0x0310	General Purpose Control Register for Slave Port 3(SGPCR3)	S/U
0x0400	Master Priority Register for Slave Port 4(MPR4)	S/U
0x0410	General Purpose Control Register for Slave Port 4(SGPCR4)	S/U
0x0500	Master Priority Register for Slave Port 5(MPR5)	S/U
0x0510	General Purpose Control Register for Slave Port 5(SGPCR5)	S/U
0x0600	Master Priority Register for Slave Port 6(MPR6)	S/U
0x0610	General Purpose Control Register for Slave Port 6(SGPCR6)	S/U
0x0700	Master Priority Register for Slave Port 7(MPR7)	S/U
0x0710	General Purpose Control Register for Slave Port 7(SGPCR7)	S/U

**Note (1) :** S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. Accessing supervisor only addresses in user mode has no effect and result in a cycle termination transfer error.



13.2.2. Register Descriptions

13.2.2.1. 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 32-bits 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.

MPRn: Offset Address: 0x0000 + n\*0x100

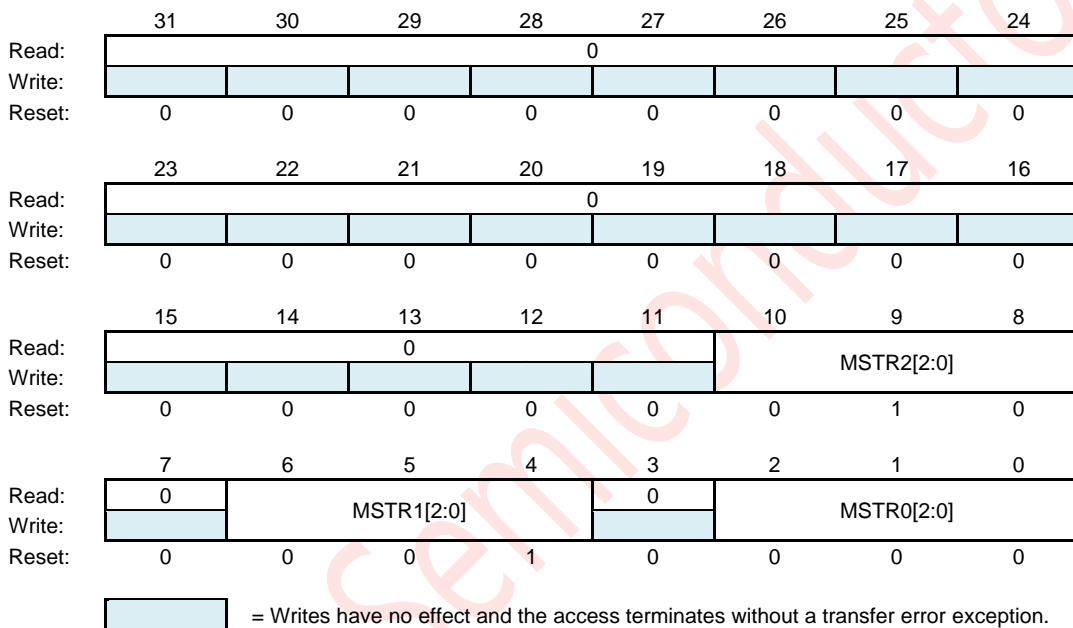


Figure 13-2: Master Priority Register (XBAR\_MPRn)

MSTR2[2:0] — 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.

## MSTR1[2:0] — 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.

## MSTR0[2:0] — 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.

### 13.2.2.2. Slave General Purpose Control Register (XBAR\_SGPCRn)

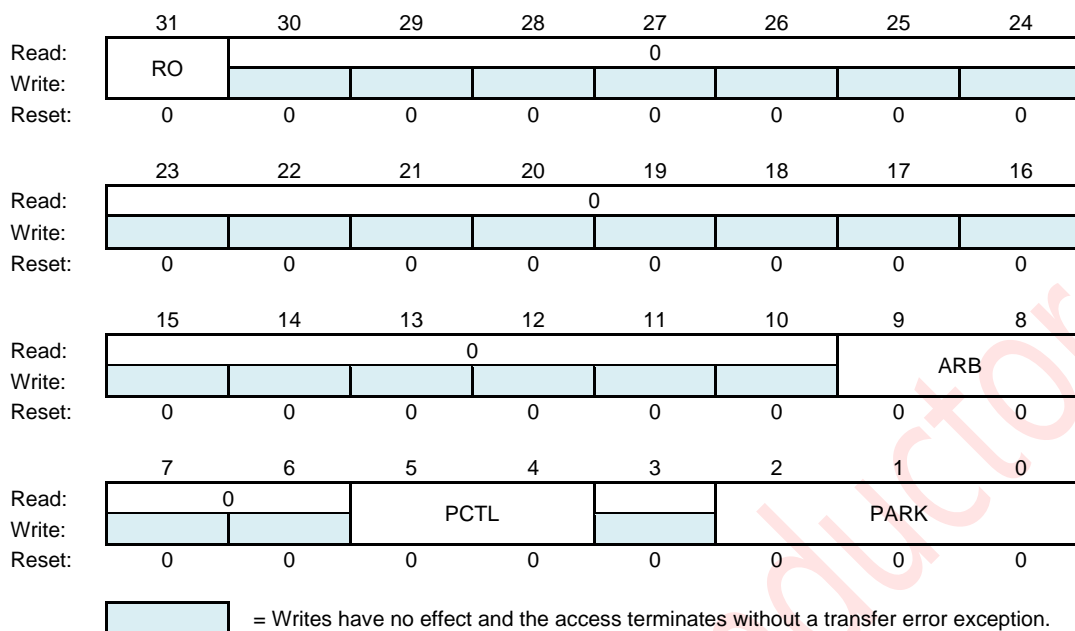
The Slave General Purpose Control Register (SGPCR) controls several features of each slave port. The Read Only (RO) bit will prevent any registers associated with this slave port from being written to once set. This bit may be written with '0' as many times as the user desires, but once it is written 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 a 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 users program 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 32-bits 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.

**SGPCRn: Offset Address: 0x0010 + n\*0x100**



**Figure 13-3: Slave General Purpose Control Register (XBAR\_SGPCRn)**

**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).

**ARB[1:0]** — 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

**PCTL[1:0]** — 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

## PARK[1:0]

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:** Select only the 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

## 13.3. Function

### 13.3.1. 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 the priority level of each master and the access time of the responding slave.

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].

### 13.3.2. 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.

### 13.3.3. 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.

#### 13.3.3.1. 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

#### 13.3.3.2. 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 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.

#### **13.3.4. Priority Assignment**

Each master port needs to be assigned a unique 3-bits 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.

#### **13.4. 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.

Levetop Semiconductor

## 14. Direct Memory Access Controller (DMAC)

### 14.1. Information Specific of DMA Controller

This chapter presents the LT168's Direct Memory Access Controller.

#### 14.1.1. DMAC Features

- 6 programmable channels to support independent 8, 16, or 32-bits 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.

### 14.1.2. Channel Assignments

The assignments between the DMA requests from the blocks to the channels of the DMA are shown in **Table 14-1**.

**Table 14-1: 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
QSPI1 DMA Receive Request	3	QSPI1 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
QSPI1 DMA Transmit Request	4	QSPI1 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
QSPI2 DMA Receive Request	5	QSPI2 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
QSPI2 DMA Transmit Request	6	QSPI2 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
PIT1 DMA Request	7	pit0_dma_req signal is generated when the PIT0 counter reaches 0x0000 and PCSR[PDMAE] = 1
PIT2 DMA Request	8	pit0_dma_req signal is generated when the PIT0 counter reaches 0x0000 and PCSR[PDMAE] = 1
PIT3 DMA Request	9	pit0_dma_req signal is generated when the PIT0 counter reaches 0x0000 and PCSR[PDMAE] = 1
SCI0 TX DMA Request	10	sci0_tx_req signal is generated when the number of datawords in the transmit FIFOs 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 FIFOs 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]
SCI2 TX DMA Request	14	sci2_tx_req signal is generated when the number of datawords in the transmit FIFOs equal to or less than the number indicated by SCI_WATER[TXWATER])
SCI2 RX DMA Request	15	sci2_rx_req signal is generated when the number of datawords in the receive buffer is greater than the number indicated by SCI_WATER[RXWATER]



### 14.2. 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 14-1 is a block diagram of the DMA module.

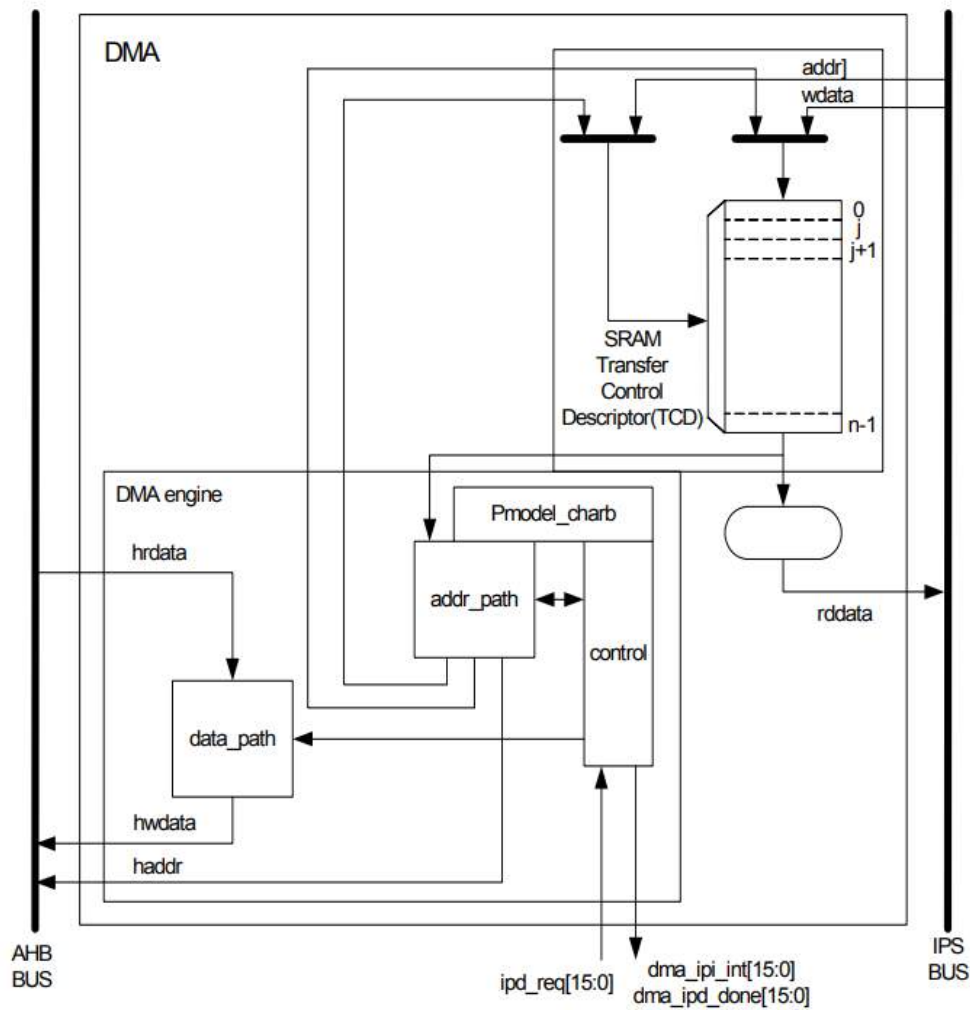


Figure 14-1: DMA Block Diagram

### 14.2.1. 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 is defined by a “minor” byte transfer count
  - An outer data transfer loop is 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 is 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 scatter/gather DMA processing
- Support complex data structures
- Support cancelling transfers via software or hardware

## 15. Efuse Control Module (EFM)

### 15.1. Introduction

Efuse Module is a parallel-in/parallel-out Electrical Fuse Macro IP which has 512-bits internal nonvolatile one-time programmable EFUSE storage. Through a serial interface, 1-bit can be programmed each time in program mode and 8-bits can be read at one time in read mode. The Efuse array are used for storing the trimming and configuration bits for the chip.

### 15.2. Features

Features of the EFM include:

- One-time programmable nonvolatile EFUSE storage cells are organized as 64 x 8-bits
- The EFUSE Macro has three modes of operation:
  - Program Mode
  - Read Mode
  - Inactive Mode

### 15.3. Memory Map and Registers

#### 15.3.1. Memory Map

The Efuse module also contains a set of control registers. The memory map for these registers is shown in **Table 15-1**.

**Table 15-1: Efuse Register Memory Map**

Address Offset	Bits[31:0]	Access <sup>1,2</sup>
0x0000	EFMCR	S/U
0x0080~0x00BF	EFMDR <sup>3</sup>	S/U

1. S = CPU supervisor mode access only.
2. Accessing to supervisor-only address locations in user mode has no effect and result in a cycle termination transfer error.
3. EFMDR can only be read by 8-bits access. Both 16-bits and 32-bits read will return wrong read data.

15.3.2. Register Descriptions

The Efuse Module register map is shown as following. The Efuse base address is 0x4012\_0000.

15.3.2.1. Efuse Configuration Register(EFMCR)

The Efuse Configuration Register (EFMCR) is unbanked and is used to configure and control the operation of the Efuse array.

Address: EFM\_BASEADDR+0x0000\_0000

	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:	AEN_PULSE[7:0]							
Write:								
Reset:	0	0	0	1	0	0	0	0
	15	14	13	12	11	10	9	8
Read:	READ_HOLD[7:0]							
Write:								
Reset:	0	0	0	1	0	0	0	0
	7	6	5	4	3	2	1	0
Read:	READ_SETUP[7:0]							
Write:								
Reset:	0	0	0	1	0	0	0	0

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

Figure 15-1: Efuse Module Configuration Register(EFMCR)

**AEN\_PULSE[7:0]** — Efuse AEN pulse width configuration

These bits determine the width of AEN of Efuse. Note that T<sub>AEN</sub> should not be less than 100ns.

$$T_{AEN} = (AEN\_PULSE[7:0] + 1) * T_{IPS\_CLK}$$

**READ\_HOLD[7:0]** — Efuse AEN to Address and RDEN hold time configuration.

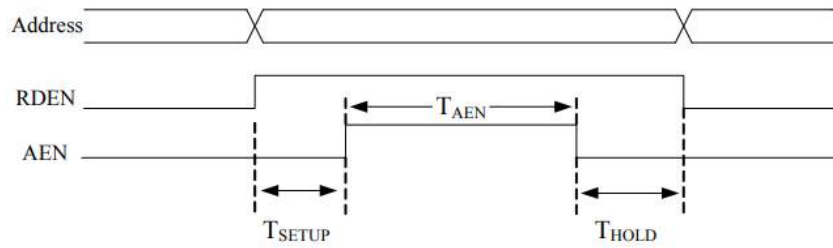
These bits determine the hold time of AEN to Address and RDEN. Note that T<sub>HOLD</sub> should not be less than 100ns.

$$T_{HOLD} = (READ\_HOLD[7:0] + 1) * T_{IPS\_CLK}$$

**READ\_SETUP[7:0]** — Efuse Address and RDEN to AEN setup time configuration.

These bits determine the setup time of Address and RDEN to AEN. Note that T<sub>SETUP</sub> should not be less than 100ns.

$$T_{SETUP} = (READ\_SETUP[7:0] + 1) * T_{IPS\_CLK}$$

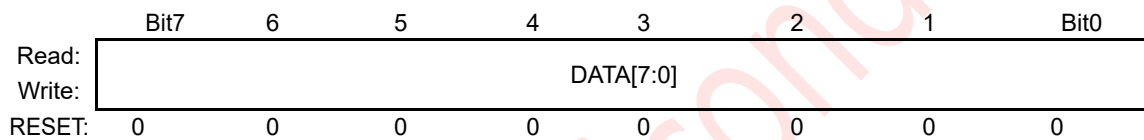


**Figure 15-2: Efuse Read Timing**

**15.3.2.2. Efuse Read Data Register(EFMDR)**

The Efuse Read Data Register (EFMDR) is used to store the data read from Efuse array and EFMDR can only be accessed by 8-bits read instruction.

**Address: EFM\_BASEADDR+0x0000\_0080~00BF**



**Figure 15-3: Efuse Read Data Register(EFMDR)**

**DATA[7:0]** — The EFMDR stores the data read from Efuse array.

## 16. Option Byte (OPB)

### 16.1. Memory Map and Registers

#### 16.1.1. Memory Map

The Option Byte memory map is shown in **Table 16-1**, and its base address is 0x4012\_0000 which is the same as the Efuse Module.

**Table 16-1: Register Memory Map**

Address Offset	Bits[31:24]	Bits[23:16]	Bits[15:8]	Bits[7:0]	Access <sup>1,2</sup>
0x001C	CCR		PVDC		S
0x0020	PLLOCKCR		EOSCST		S
0x0024	PVDFEVR		RFEVR		S
0x002C	FCR		Reserved3		S
0x0030	Reserved				S
0x0034	LDOTCR	VREFTCR	ADCCDISR		S
0x0038	RTCTCR		Reserved		S

**Notes:**


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

#### 16.1.2. Register Descriptions

##### 16.1.2.1. Programmable Voltage Detector Configuration Register (PVDC)

Address: EFM\_BASEADDR+0x0000\_001C

	15	14	13	12	11	10	9	8
Read:	PVDF	PVDPORR	PVDTEST[1:0]		PVDOE	PVDRE	PVDIE	PVDE
Write:		E						
RESET:	0	Note1	0	0	1	Note1	0	1
	7	6	5	4	3	2	1	0
Read:	0						PVDC[1:0]	
Write:								
RESET:	0	0	0	0	0	0	Note1	Note1

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

**Note1:** Determined by the value of FUSE Area

**Figure 16-1: Programmable Voltage Detector Configuration Register (PVDC)**

**PVDF** — Programmable Voltage Detector Flag

The PVDF indicates if 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

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

The PVDPORRE shows whether Program Voltage Detector Power-On reset is enable or not. When enabled, RSR[POR] flag will be set after PVD reset. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is loaded from the corresponding bit of FUSE area after power-on. This bit can be set by software after power on reset.

- 1 = PVD will generate Power-On reset when VDD33 is lower than PVD threshold
- 0 = PVD will NOT generate Power-On reset when VDD33 is lower than PVD threshold

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

The PVDC register can not be changed, unless the correct sequence is written. The right sequence is: 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the PVDC register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

**PVDOE** — Programmable Voltage Detector Output Enable

- 1 = PVD Output Enable
- 0 = PVD Output Disable

**PVDRE** — Program Voltage Detector (PVD) Reset Enable

The PVDRE shows whether Program Voltage Detector reset is enable. The default value is loaded from the FUSE area. When enabled and PVDPORRE is disabled, RSR[PVDF] flag will be set after PVD reset. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is loaded from the corresponding bit of FUSE area after power-on. This bit can be changed by software after power on reset.

- 1 = PVD reset is enabled
- 0 = PVD reset is disabled

**PVDIE** — Programmable Voltage Detector Interrupt Enable

The PVDIE is used to check if VCC is lower than PVD threshold

- 1 = If VCC is lower than PVD threshold, an interrupt will be generated
- 0 = If VCC is not lower than PVD threshold, no interrupt will be generated.

**PVDE** — Programmable Voltage Detector Enable

The PVDE determines whether PVD is enabled or not. POR can clear it.

- 1 = Enable PVD
- 0 = Disable PVD

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

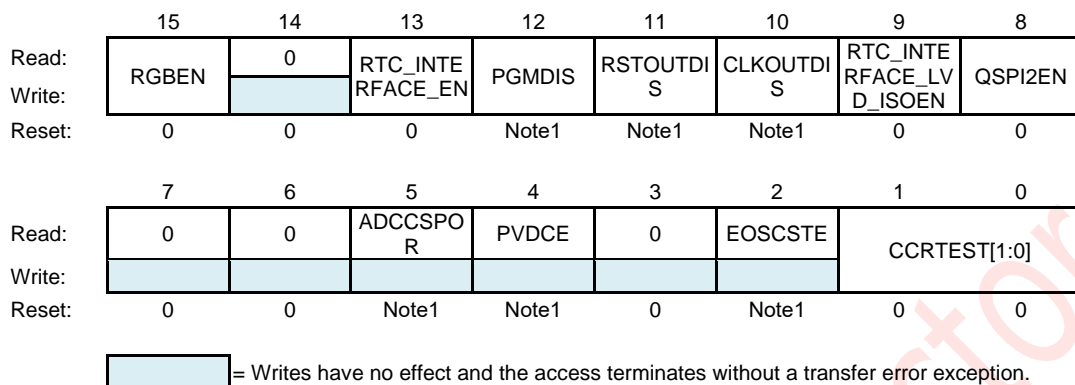
**Table 16-2: Programmable Voltage Detector**

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

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, PVDC[1:0] will be cleared after power on reset, otherwise if the content in corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. These bits can be changed by software after power on reset.

**16.1.2.2. Customer Configuration Register (CCR)**

**Address: EFM\_BASEADDR+0x0000\_001E**



**Note1:** Determined by the value of FUSE Areas

**Figure 16-2: Customer Configuration Register (CCR)**

**RGBEN** — RGB Interface Enable Control Bit

The RGBEN determines whether the RGB Interface is enabled or not.

- 1 = RGB Interface is enabled
- 0 = RGB Interface is disabled

**Table 16-3: RGB Interface Enable Control**

RGB Disabled	RGB Enabled
EBI_D[15:0]	TFT_RGB[15:0]
EBI_WR#	TFT_VS
EBI_RD#	TFT_HS
EBI_RS	TFT_DE
EBI_CS#	TFT_PCLK

**RTC\_INTERFACE\_EN** — This bit determines 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 can not be reconfigured.

**PGMDIS** — Programming Debug Interface Disable Bit

The PGMDIS shows whether the Programming Debug Interface is disabled or not. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit will be set after power-on. This bit can be changed by software after power on reset.

- 1 = Programming debug function is disabled
- 0 = Programming debug function is enabled



**Table 16-4: Programming Debug Interface Control**

Programming Enabled	Programming Disabled
PGMCK	RXD2
PGMIO	TXD2

**RSTOUTDIS** — RSTOUT Disable Bit

The RSTOUTDIS shows whether the RSTOUT pin function is disabled and INT1[6] is enabled. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit will be set after power-on. This bit can be changed by software after power on reset.

- 1 = RSTOUT Pin function is disabled
- 0 = RSTOUT Pin function is enabled

**Table 16-5: RSTOUT Disable Control**

RSTOUT Enabled	RSTOUT Disabled
RSTOUT	INT1[6]

**CLKOUTDIS** — CLKOUT Disable Bit

The CLKOUTDIS shows whether the CLKOUT pin function is disabled and INT1[0] is enabled. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit will be set after power-on. This bit can be changed by software after power on reset.

- 1 = CLKOUT Pin function is disabled
- 0 = CLKOUT Pin function is enabled

**Table 16-6: CLKOUT Disable Control**

CLKOUT Enabled	CLKOUT Disabled
CLKOUT	INT1[0]

**RTC\_INTERFACE\_LVD\_ISOEN** — This bit determines whether the RTC analog interface will be isolated or not when Program Voltage Detector event happens.

- 1 = RTC analog interface will not be isolated when Low voltage happens.
- 0 = RTC analog interface will be isolated when Low voltage happens.

**QSPI2EN** — QSPI2 Interface Enable Control Bit

The QSPI2EN determines whether the QSPI2 Interface is enabled or not.

- 1 = QSPI2 Interface is enabled
- 0 = QSPI2 Interface is disabled

**Table 16-7: QSPI2 Interface Enable Control**

QSPI2 Disabled	QSPI2 Enabled
PWM0[0]	QSCS2#
PWM0[1]	QSCK2
PWM0[2]	QSIO2[0]
PWM0[3]	QSIO2[1]

QSPI2 Disabled	QSPI2 Enabled
PWM1[0]	QSIO2[2]
PWM1[1]	QSIO2[3]

**ADCCSPOR** — ADC Channel Setting Bit after power-on

The ADCCSPOR shows whether ADCCDISR[ADCCDIS] is loaded from FUSE area or not after power on. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

- 1 = ADCCDISR[ADCCDIS] is loaded from FUSE area after Power on
- 0 = ADCCDISR[ADCCDIS] is not loaded from FUSE area after Power on

**PVDCE** — Program Voltage Detector (PVD) Configuration Enable

The PVDCE shows whether Program Voltage Detector configuration is loaded from FUSE area or not after power on. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

- 1 = PVD configuration is loaded from FUSE Area after Power on
- 0 = PVD configuration is not loaded from FUSE Area after Power on

**EOSCSTE** — External High Speed Oscillator Stable Time Configuration Enable

The EOSCSTE shows whether External High Speed Oscillator Stable Time configuration is loaded from FUSE Area after power on. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

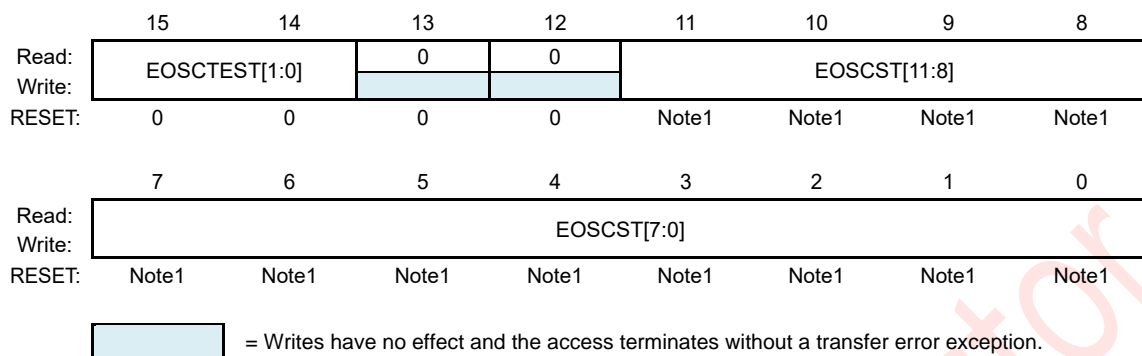
- 1 = External High Speed Oscillator(FXOSC) Stable Time configuration is loaded from FUSE Area after Power on
- 0 = External High Speed Oscillator(FXOSC) Stable Time configuration is not loaded from FUSE Area after Power on

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

The writable bit of CCR register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of CCR register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

16.1.2.3. External High Speed Oscillator Stable Time Configuration Register (EOSCST)

Address: EFM\_BASEADDR+0x0000\_0020



Note1: Determined by the value of FUSE Area

Figure 16-3: External High Speed Oscillator Stable Time Configuration Register (EOSCST)

**EOSCTEST[1:0]** — Write Access Enable Sequence Input

The writable bit of EOSCST register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of EOSCST register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

**EOSCST[11:0]** — External High Speed Oscillator Stable Time Value.

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the value of EOSCST[11:0] is 12'h3ff after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. Software can change it after power on reset. After External Oscillator is enabled, it will wait EOSCST[11:0] cycles of 128KHz oscillator and then the gate of the clock will be turn-on.

16.1.2.4. PLL Lock Time Configuration Register (PLLOCKCR)

Address: EFM\_BASEADDR+0x0000\_0022

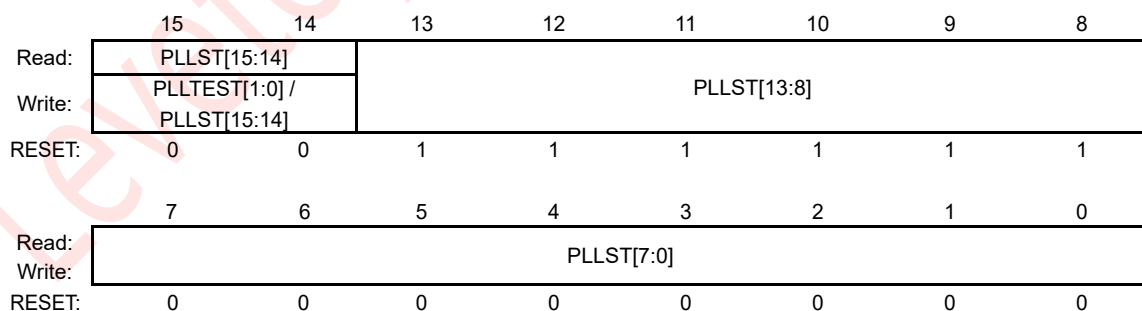


Figure 16-4: PLL Lock Time Configuration Register (PLLOCKCR)

**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.

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

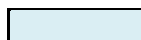
The PLLST register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the PLLST register can be changed at will. Any writes can clear these two bits when the value equals to 2'b11.

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

**16.1.2.5. RESET Pin Filter Enable and Value Register (RFEVR)**

**Address: EFM\_BASEADDR+0x0000\_0024**

	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:	Note1	Note1	Note1	Note1	Note1	Note1	Note1	Note1

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

**Note1:** Determined by the value of FUSE Area

**Figure 16-5: RESET Pin Filter Enable and Value Register (RFEVR)**

**RFEVRTEST [1:0]** — Write Access Enable Sequence Input

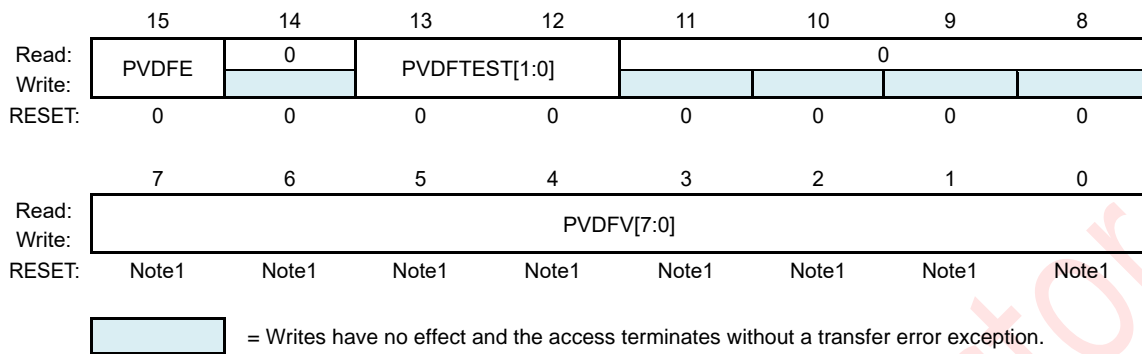
The writable bit of RFEVR register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of RFEVR register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

**RFV[7:0]** — RESET Pin Filter Value

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the value of RFV[7:0] is 8'h0d after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. These bits can be changed by software after power on reset.

**16.1.2.6. Programmable Voltage Detector Filter Enable and Value Register (PVDFEVR)**

**Address: EFM\_BASEADDR+0x0000\_0026**



**Note1:** Determined by the value of FUSE Area

**Figure 16-6: Programmable Voltage Detector Filter Enable and Value Register (PVDFEVR)**

**PVDFE** — Programmable Voltage Detector Filter Enable.

The PVDFE shows whether Program Voltage Detector Filter is enabled. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the PVDFE is cleared after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. This bit can be changed by software after power on reset.

- 1 = PVD filter is enabled
- 0 = PVD filter is disabled

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

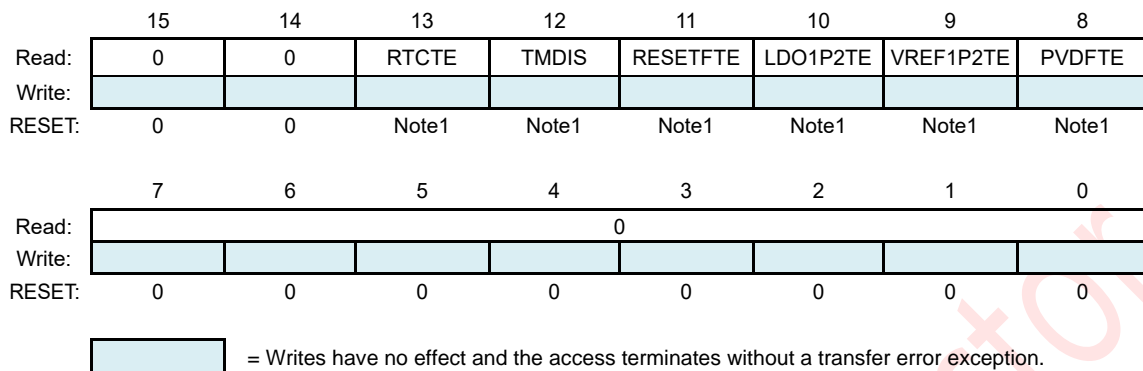
The writable bit of PVDFEVR register can not be changed, unless the correct sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of PVDFEVR register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

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

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the value of PVDFV[7:0] is 8'h0 after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. These bits can be changed after power on reset.

**16.1.2.7. Factory Configuration Register (FCR)**

**Address: EFM\_BASEADDR+0x0000\_002E**



**Note1:** Determined by the value of FUSE Area

**Figure 16-7: Factory Configuration Register (FCR)**

**RTCTE** — RTC Bias/Cap rimming Enable

The RTCTE shows whether RTC Bias or Capacitance is trimmed or not. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

- 1 = RTC Bias or Capacitance is trimmed
- 0 = RTC Bias or Capacitance is not trimmed

**TMDIS** — Test Mode Disable Bit

The TMDIS shows whether test mode (except STB) is disabled or not. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

- 1 = Test mode is disabled
- 0 = Test mode is enabled

**RESETFTE** — Reset Pin Filter Trimming Enable

The RESETFTE shows whether Reset Pin Filter is trimmed or not. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

- 1 = Reset Pin Filter is trimmed
- 0 = Reset Pin Filter is not trimmed

**LDO1P2TE** — LDO1P2 Trimming Enable

The LDO1P2TE shows whether LDO1P2 is trimmed or not. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

- 1 = LDO1P2 is trimmed
- 0 = LDO1P2 is not trimmed

**VREF1P2TE** — VREF1P2 Module Trimming Configuration Register

The VREF1P2TE shows whether VREF1P2 Module is trimmed or not. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

- 1 = Internal VREF1P2 Module is trimmed
- 0 = Internal VREF1P2 Module is not trimmed

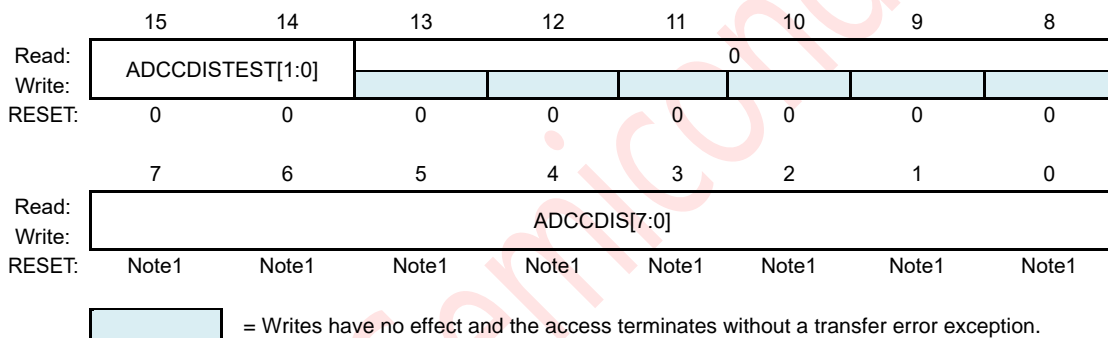
**PVDFTE** — PVD Filter Trimming Enable

The PVDFTE shows whether PVD Filter is trimmed or not. The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, this bit is cleared, otherwise if the content in the corresponding FUSE area is matched, this bit is set after power-on.

- 1 = PVD Filter is trimmed
- 0 = PVD Filter is not trimmed

**16.1.2.8. Channel Disable Configuration Register (ADCCDISR)**

**Address: EFM\_BASEADDR+0x0000\_0034**



**Note1:** Determined by the value of FUSE Area

**Figure 16-8: ADC Channel Disable Configuration Register (ADCCDISR)**

**ADCCDISTEST[1:0]** — Write Access Enable Sequence Input

The writable bits ADCCDIS[7:0] can not be changed until the correct sequences are written into ADCCDISTEST. The right sequence are : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of ADCCDISTEST register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

**ADCCDIS[7:0]** — ADC Channel Disable Configuration

ADCCDIS[7:0] determine whether AIN[7:0] are used as ADC channel or GPIO. If the corresponding FUSE area is in an erased status, these bits are cleared and ADC channel function is valid, otherwise if the content in the corresponding FUSE area is matched, ADCCDIS[7:0] are loaded from the FUSE area. These bits can be changed after power on reset.

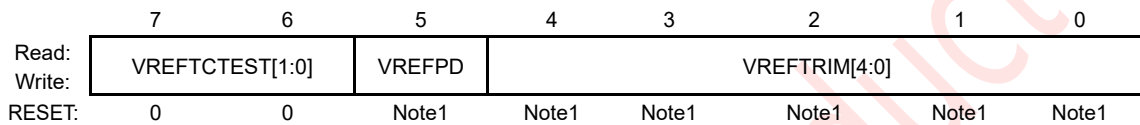
**Table 16-8: ADC Channel Disable Control**

ADC Channel Disabled	ADC Channel Enabled
INT0[0]	AIN[0]
INT0[1]	AIN[1]
INT0[2]	AIN[2]

ADC Channel Disabled	ADC Channel Enabled
INT0[3]	AIN[3]
INT0[4]	AIN[4]
INT0[5]	AIN[5]
INT0[6]	AIN[6]
INT0[7]	AIN[7]

**16.1.2.9. VREF1P2 Trimming Configuration Register (VREFTCR)**

**Address: EFM\_BASEADDR+0x0000\_0036**



**Note1:** Determined by the value of FUSE Area

**Figure 16-9: VREF Trimming Configuration Register (VREFTCR)**

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

The writable bit of VREFTCTEST register can not be changed, unless the correct data sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of VREFTCTEST register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values 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.

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the VREFPD is set after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. This bit can be changed by software after power on reset.

- 1 = Internal VREF1P2 Module Power Down
- 0 = Internal VREF1P2 Module Power On

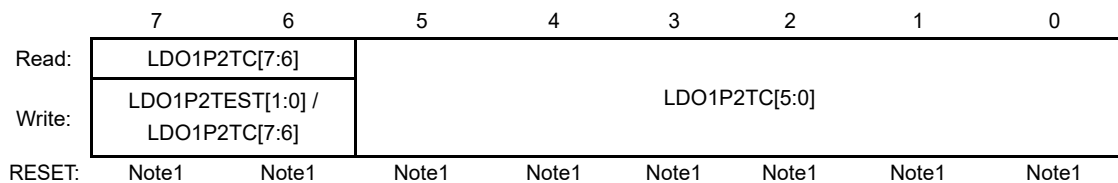
**VREFTRIM[4:0]** — VREF1P2 Module Trimming Value

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the VREFTRIM[4:0] is 5'h18 after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. These bits can be changed by software after power on reset.



**16.1.2.10. LDO1P2 Trimming Configuration Register (LDOTCR)**

**Address: EFM\_BASEADDR+0x0000\_0037**



**Note1:** Determined by the value of FUSE Area

**Figure 16-10: LDO1P2 Trimming Configuration Register (LDOTCR)**

**LDO1P2TC[7]** — LDO over-current limit function control bit

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the LDO1P2TC[7] is 1'b0 after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. This bit can be changed after power on reset.

- 1 = LDO over-current limit function will be turned off.
- 0 = LDO over-current limit function will be turned on.

**LDO1P2TC[6]** — LDO1P2 power down control bit

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the LDO1P2TC[6] is 1'b0 after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. This bit can be changed after power on reset.

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

**LDO1P2TC[5:0]** — LDO1P2 Trimming Value

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the LDO1P2TC[5:0] is 6'h00 after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. These bits can be changed by software after power on reset.

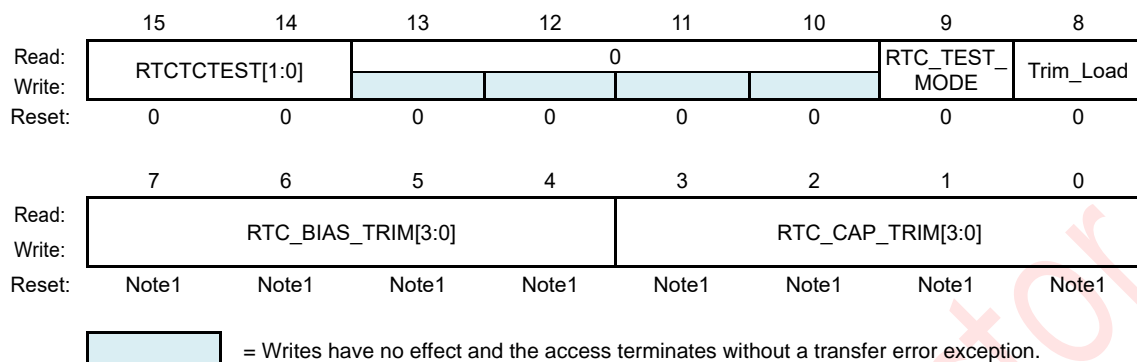
**LDO1P2TEST[1:0]** — LDO1P2TC Write Access Sequence In

The writable bit of LDO1P2TC register can not be changed, unless the correct data sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by 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.

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

16.1.2.11. RTC Trimming Configuration Register (RTCTCR)

Address: EFM\_BASEADDR+0x0000\_003A



**Note1:** Determined by the value of FUSE Area

**Figure 16-11: RTC Trimming Configuration Register (RTCTCR)**

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

The writable bit of RTCTC register can not be changed, unless the correct data sequence is written. The right sequence is : 2'b01 → 2'b10 → 2'b11. After writing these two bits by this sequence, these two bits' value == 2'b11, then the writable bit of RTCTC register can be changed at will. Writing 2'b00 can clear these two bits when the value equals to 2'b11. Writing other values has no effect and returns 2'b11.

**RTC\_TEST\_MODE** — RTC Test Mode enable.

- 1 = RTC Test Mode is enabled.
- 0 = RTC Test Mode is disabled.

**Trim\_Load** — Low to high edge will latch RTC\_BIAS\_TRIM and RTC\_CAP\_TRIM value into RTC analog module.

**RTC\_BIAS\_TRIM[3:0]** — RTC Oscillator Bias Trimming configuration bits.

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the RTC\_BIAS\_TRIM[3:0] is 4'b0111 after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. These bits can be changed by software after power on reset.

**RTC\_CAP\_TRIM[3:0]** — RTC Oscillator Load Capacitance C1 and C2 Trimming configuration bits.

The default value is loaded from the FUSE area. If the corresponding FUSE area is in an erased status, the RTC\_BIAS\_TRIM[3:0] is 4'b1100 after power on reset, otherwise if the content in the corresponding FUSE area is matched, these bits are loaded from the FUSE area after power-on. These bits can be changed by software after power on reset.

## 17. RGB Controller Module (RGBC)

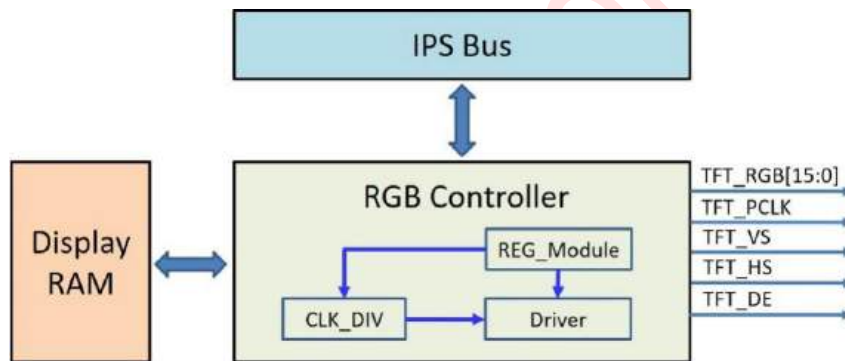
### 17.1. Introduction

LT168 has an embedded RGB Controller that can generate all the necessary signals required to drive the TFT LCD displays. These signals include 16-bits RGB data bus, Pixel Clock, Data Enable, Horizontal-Sync, and Vertical-Sync.

The RGB Controller features the following:

- Data Frame Format: Red:5, Green:6, Blue:5
- Supported Resolution Ratio:
  - 480 x 480
  - 480 x 272 or Less
- Timing of the interface signals is configurable
- Polarity and pixel clock polarity is configurable

### 17.2. Block Diagram



**Figure 17-1: RGB Controller Block Diagram**

### 17.3. RGB Interface Definition

The display operation via the RGB interface is synchronized with the VSYNC, HSYNC, and TFT\_PCLK signals. The data can be written only within the specified area. The back porch and front porch are used to set the RGB interface timing.

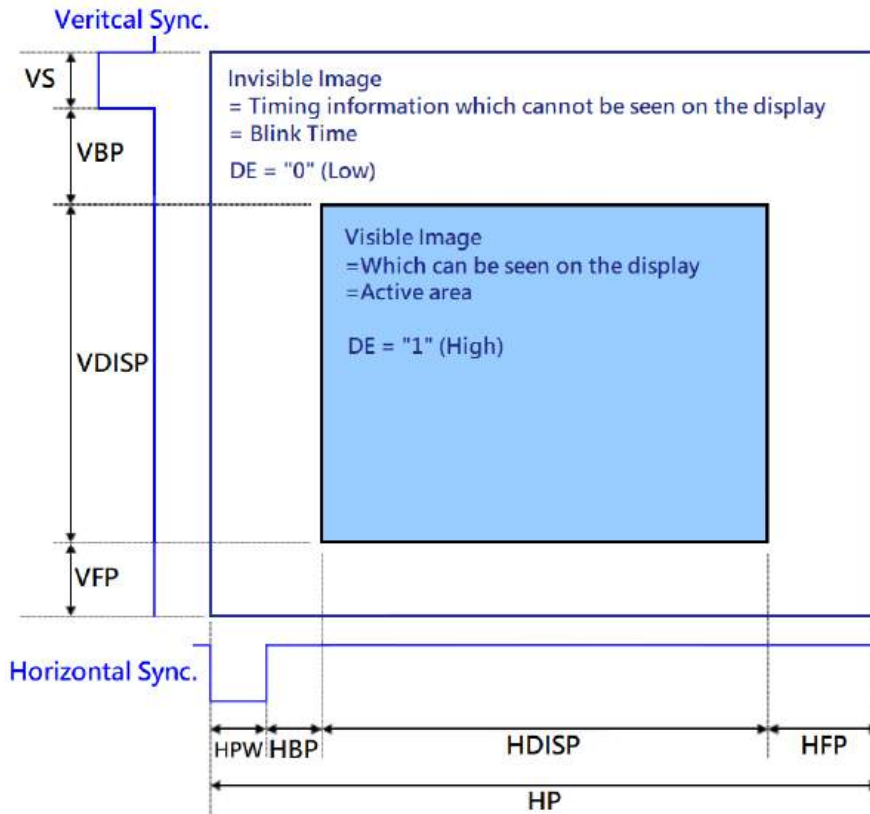


Figure 17-2: Display RAM Access Area by RGB Interface

### 17.4. RGB Interface Timing

The timing chart of RGB interface is shown as follows.

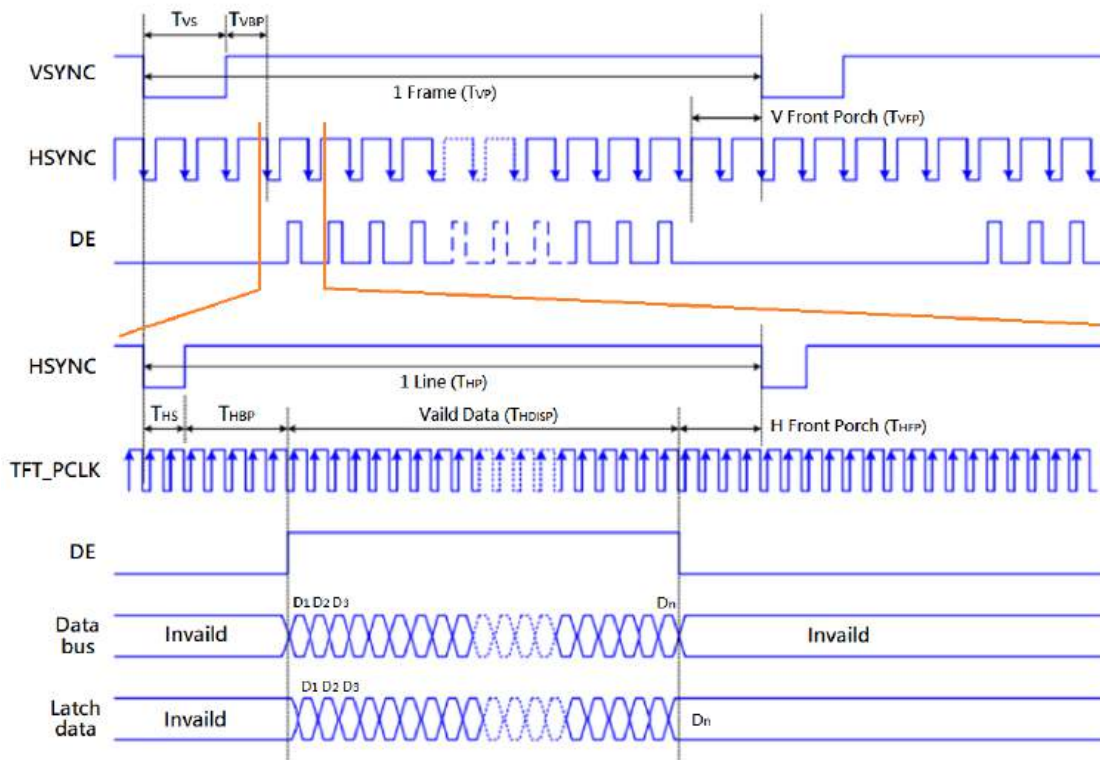


Figure 17-3: Timing Chart of Signals in RGB Interface

## 17.5. Memory Map and Registers

The RGB Controller memory map is shown in **Table 17-1**, and its base address is **0x401A\_0000**. This subsection describes the memory map and registers for the RGB controller.

### 17.5.1. Memory Map

Refer **Table 17-1** to for an overview of the RGB controller memory map.

**Table 17-1: RGB Controller Module Memory Map**

Offset Address	Bits[31:0]	Access
0x0000	Horizontal pulse width and back porch Register (THPB) <sup>1</sup>	R/W
0x0004	Horizontal period and front porch Register (THDF) <sup>1</sup>	R/W
0x0008	Vertical pulse width and back porch Register (TVPB) <sup>1</sup>	R/W
0x000C	Vertical period and front porch Register (TVDF) <sup>1</sup>	R/W
0x0010	Memory start address Register (MSADDR)	R/W
0x0014	Memory end address Register (MEADDR)	R/W
0x0018	Number of Frames Transferred Register (NFTR)	R
0x001C	Number of Frames Transferred Threshold Configuration Register(NFTTCR)	R/W
0x0020	RGB Control and Status Register(RCSR)	R/W

**Note:** These registers can only be changed when RCSR[RGBE] = 1'b0.

**17.5.2. Register Descriptions**

**17.5.2.1. Horizontal Pulse Width And Back Porch Register (THPB)**

**Address: RGBC\_BASEADDR+0x0000\_0000**

	31	30	29	28	27	26	25	24
Read:	HPW[15:8]							
Write:								
Reset:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
Read:	HPW[7:0]							
Write:								
Reset:	0	0	1	0	1	0	0	1
	15	14	13	12	11	10	9	8
Read:	HBP[15:8]							
Write:								
Reset:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
Read:	HBP[7:0]							
Write:								
Reset:	0	0	0	0	0	0	1	0

**Figure 17-4: Horizontal Pulse Width and Back Porch Register (THPB)**

**HPW[15:0]** — Horizontal synchronizing pulse width (in pixel clock cycles).

$$T_{HPW} = HPW[15:0] * T_{pixel\_clock}$$

**HBP[15:0]** — HSYNC back-porch pulse width (in pixel clock cycles).

$$T_{HBP} = HBP[15:0] * T_{pixel\_clock}$$

**17.5.2.2. Horizontal Period And Front Porch Register (THDF)**

**Address: RGBC\_BASEADDR+0x0000\_0004**

	31	30	29	28	27	26	25	24
Read:	HDISP[15;8]							
Write:	HDISP[15;8]							
Reset:	0	0	0	0	0	0	0	1
	23	22	21	20	19	18	17	16
Read:	HDISP[7:0]							
Write:	HDISP[7:0]							
Reset:	1	1	1	0	0	0	0	0
	15	14	13	12	11	10	9	8
Read:	HFP[15;8]							
Write:	HFP[15;8]							
Reset:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
Read:	HFP[7:0]							
Write:	HFP[7:0]							
Reset:	0	0	0	0	0	0	1	0

**Figure 17-5: Horizontal Period and Front Porch Register (THDF)**

**HDISP[15:0]** — HSYNC active pulse width (in pixel clock cycles).

$$T_{HDISP} = HDISP[15:0] * T_{pixel\_clock}$$

**HFP[15:0]** — HSYNC front-porch pulse width (in pixel clock cycles).

$$T_{HFP} = HFP[15:0] * T_{pixel\_clock}$$



**17.5.2.3. Vertical Pulse Width And Back Porch Register (TVPB)**

**Address: RGBC\_BASEADDR+0x0000\_0008**

	31	30	29	28	27	26	25	24
Read:	VS[15:8]							
Write:	VS[15:8]							
Reset:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
Read:	VS[7:0]							
Write:	VS[7:0]							
Reset:	0	0	0	0	1	0	1	0
	15	14	13	12	11	10	9	8
Read:	VBP[15:8]							
Write:	VBP[15:8]							
Reset:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
Read:	VBP[7:0]							
Write:	VBP[7:0]							
Reset:	0	0	0	0	0	0	1	0

**Figure 17-6: Vertical Pulse Width and Back Porch Register (TVPB)**

**VS[15:0]** — VSYNC active pulse width (in horizontal line cycles).

$$T_{VS} = VS[15:0] * T_{Horizontal\_line}$$

**VBP[15:0]** — VSYNC back-porch pulse width (in horizontal line cycles).

$$T_{VBP} = VBP[15:0] * T_{Horizontal\_line}$$

**17.5.2.4. Vertical Period And Front Porch Register (TVDF)**

**Address: RGBC\_BASEADDR+0x0000\_000C**

	31	30	29	28	27	26	25	24
Read:	VDISP[15:8]							
Write:	VDISP[15:8]							
Reset:	0	0	0	0	0	0	0	1
	23	22	21	20	19	18	17	16
Read:	VDISP[7:0]							
Write:	VDISP[7:0]							
Reset:	0	0	0	1	0	0	0	0
	15	14	13	12	11	10	9	8
Read:	VFP[15:8]							
Write:	VFP[15:8]							
Reset:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
Read:	VFP[7:0]							
Write:	VFP[7:0]							
Reset:	0	0	0	0	0	0	1	0

**Figure 17-7: Vertical Period and Front Porch Register (TVDF)**

**VDISP[15:0]** — Vertical period pulse width (in horizontal line cycles).

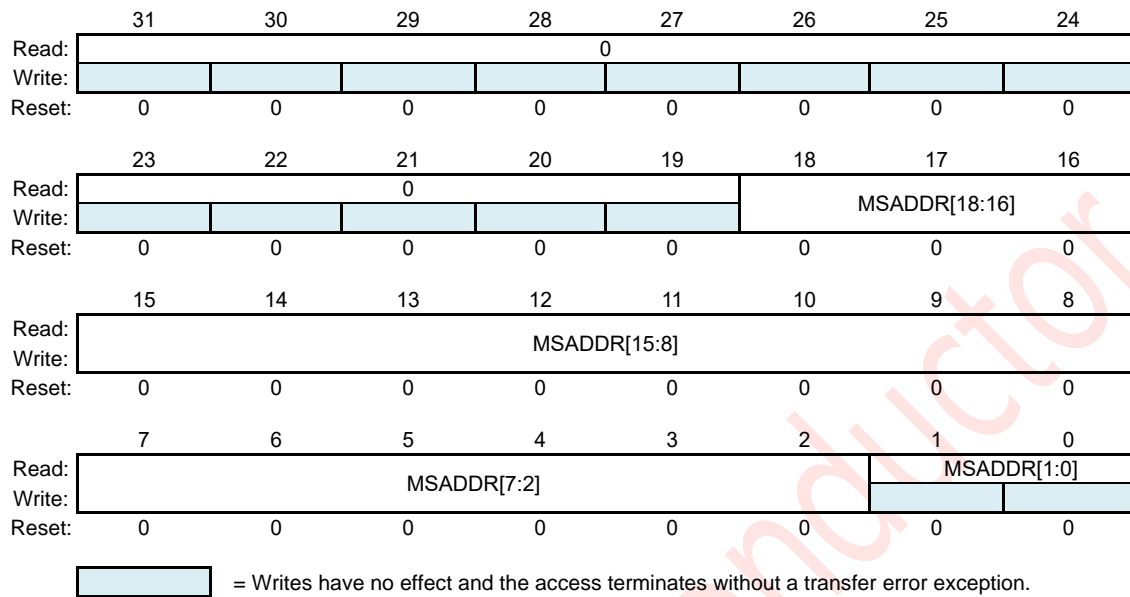
$$T_{VDISP} = VDISP[15:0] * T_{Horizontal\_line}$$

**VFP[15:0]** — VSYNC front-porch pulse width (in horizontal line cycles).

$$T_{VFP} = VFP[15:0] * T_{Horizontal\_line}$$

**17.5.2.5. Memory Start Address Register (MSADDR)**

**Address: RGBC\_BASEADDR+0x0000\_0010**



**Figure 17-8: Memory Start Address Register (MSADDR)**

**MSADDR[18:0]** — The first pixel point offset address at Display RAM, and MSADDR[1:0] are fixed to 2'b00. Once a new start address and end address are configured, the data of the previous frame can be completely transmitted.

**Note:** The MSADDR and MEADDR must be written by word access, otherwise the content of MSADDR and MEADDR will not be changed. When the MSADDR or MEADDR needs to be reconfigured, both addresses must be written, even if the other address is not changed, otherwise the reconfigured address will not take effect.

17.5.2.6. Memory End Address Register (MEADDR)

Address: RGBC\_BASEADDR+0x0000\_0014

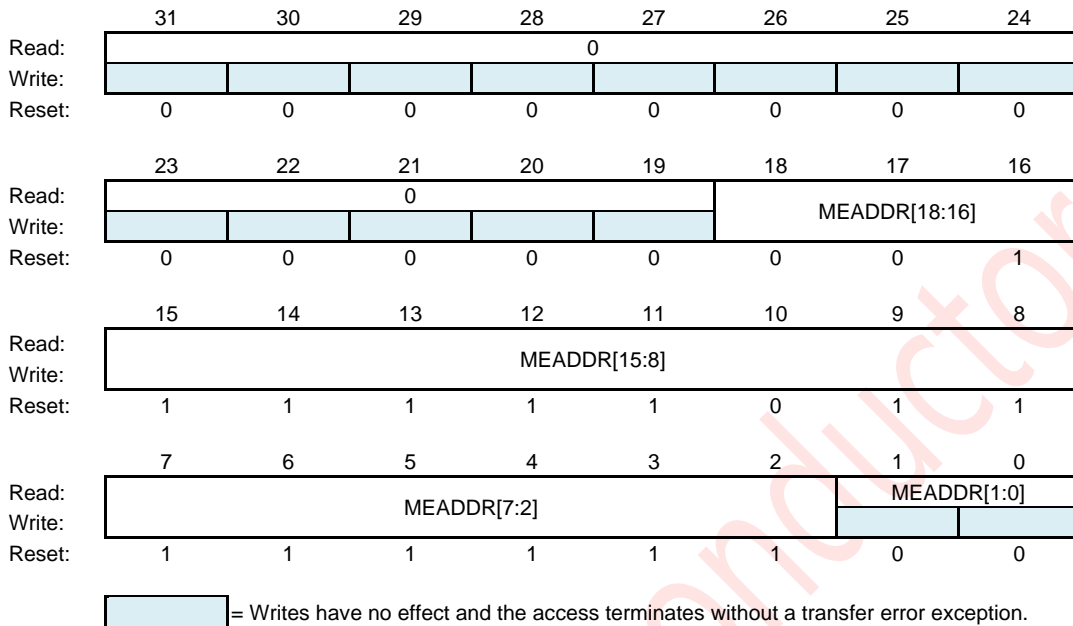


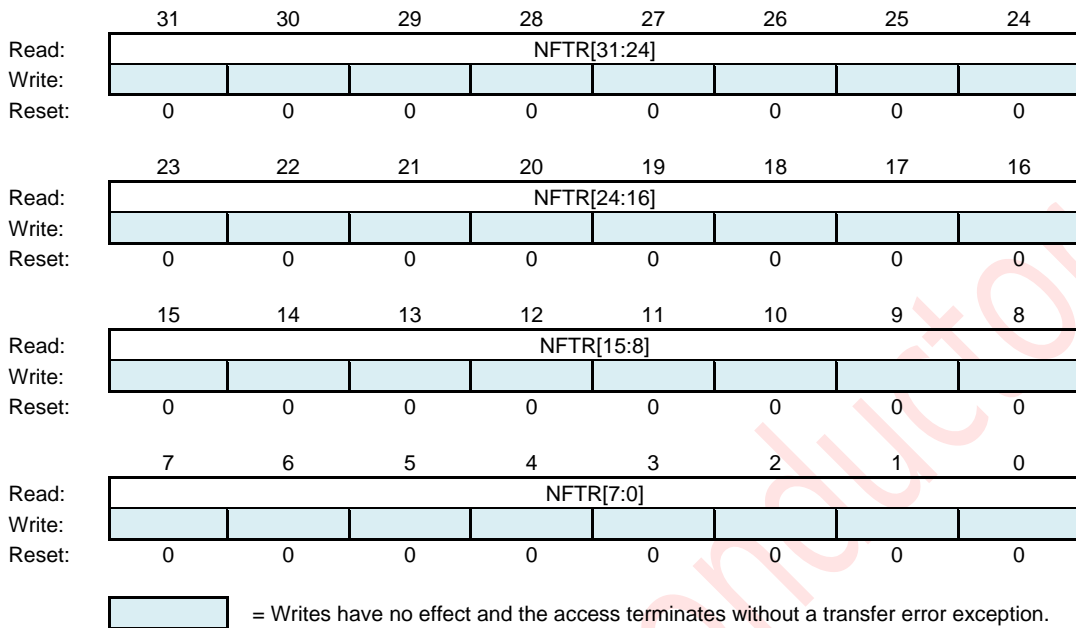
Figure 17-9: Memory End Address Register (MEADDR)

MEADDR[18:0] — The last pixel point offset address at Display RAM, and MEADDR[1:0] are fixed to 2'b00.

**Note:** The MSADDR and MEADDR must be written by word access, otherwise the content of MSADDR and MEADDR will not be changed. When the MSADDR or MEADDR needs to be reconfigured, both addresses must be written, even if the other address is not changed, otherwise the reconfigured address will not take effect.

**17.5.2.7. Number of Frames Transferred (NFTR)**

**Address: RGBC\_BASEADDR+0x0000\_0018**

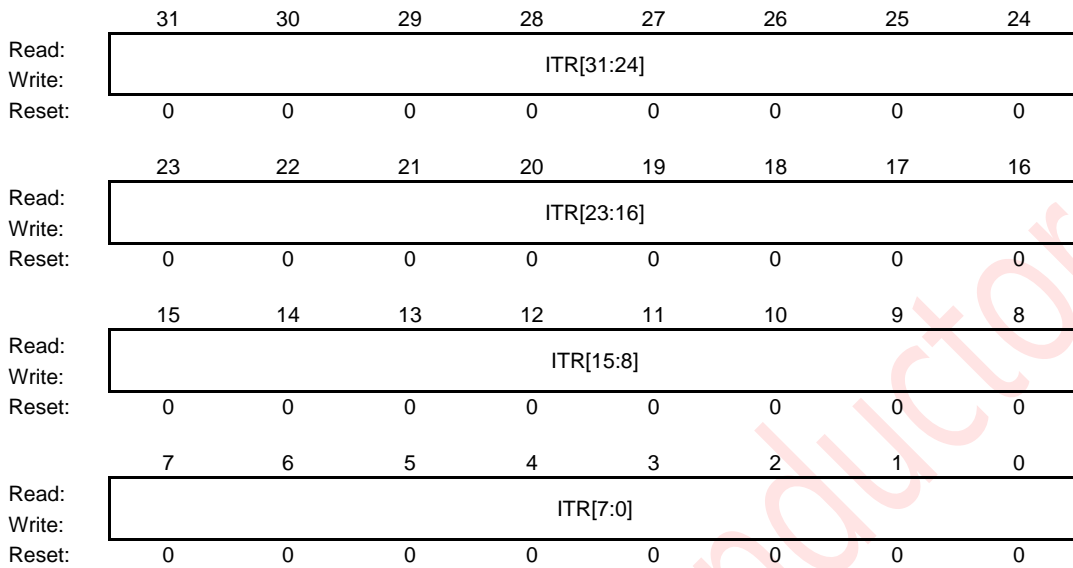


**Figure 17-10: Number of Frames Transferred Register (NFTR)**

**NFTR[31:0]** — The amount of frames have been transmitted after RGB enabled by setting RGBE. NFT can be cleared by set RGBE to 1'b0 or system reset.

**17.5.2.8. Number of Frames Transferred Threshold Configuration Register (NFTTCR)**

**Address: RGBC\_BASEADDR+0x0000\_001C**



**Figure 17-11: Number of Frames Transferred Threshold Configuration Register(NFTTCR)**

**ITR[31:0]** — Users can configure this register to determine how many frames of data transmission are completed and then RCSR[INTN] bit will be set.

17.5.2.9. RGB Control And Status Register (RCSR)

Address: RGBC\_BASEADDR+0x0000\_00020

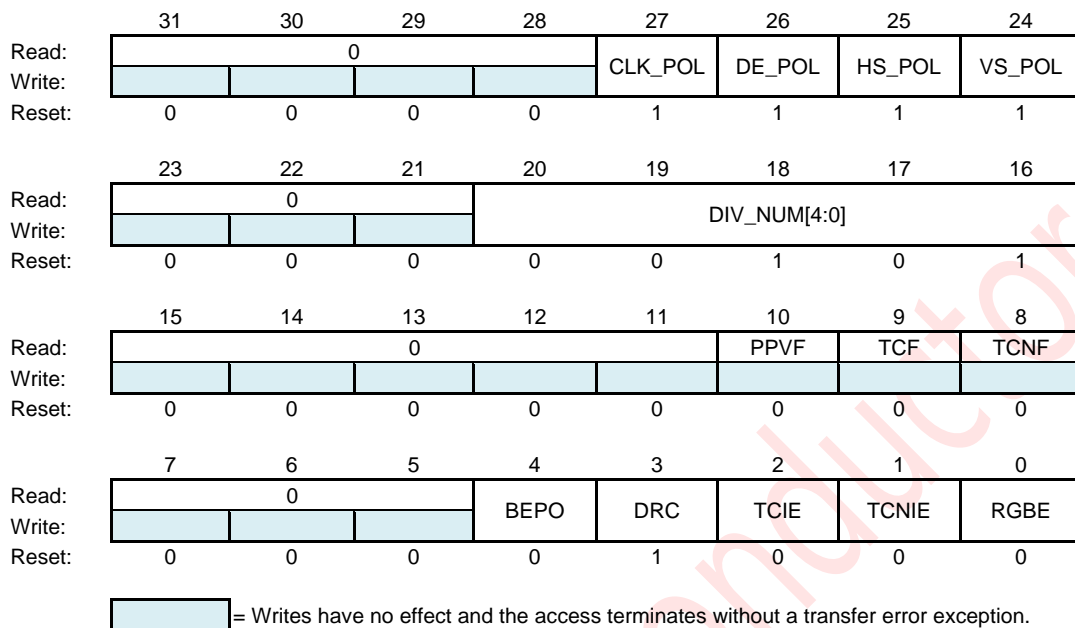


Figure 17-12: RGB Control and Status Register(RCSR)

**CLK\_POL** — This bit determines which edge of pixel clock will be used to drive the TFT LCD display signals.  
 1 = All the necessary signals required to drive the TFT LCD displays will be driven by the falling edge of pixel clock  
 0 = All the necessary signals required to drive the TFT LCD displays will be driven by the rising edge of pixel clock

**DE\_POL** — Polarity change of Data Enable.  
 1 = None-invert Data Enable signal, active HIGH  
 0 = Invert Data Enable signal, active LOW

**HS\_POL** — Polarity change of HSYNC.  
 1 = None-invert HSYNC signal, active HIGH  
 0 = Invert HSYNC signal, active LOW

**VS\_POL** — Polarity change of VSYNC.  
 1 = None-invert VSYNC signal, active HIGH  
 0 = Invert VSYNC signal, active LOW

**DIV\_NUM[4:0]** — The DIV\_NUM bits control the the divided value of the pixel clock relative to IPS clock.

**Table 17-2: Pixel Clock Divider**

<b>DIV_NUM[4:0]</b>	<b>Divide Value</b>
5'b00000	Reserved
5'b00001	Divide-by-2
5'b00010	Divide-by-4
5'b00011	Divide-by-6
5'b00100	Divide-by-8
5'b00101	Divide-by-10
5'b00110	Divide-by-12
5'b00111	Divide-by-14
.....	.....
.....	.....
5'b11101	Divide-by-58
5'b11110	Divide-by-60
5'b11111	Divide-by-62

**PPVF** — Pixel point valid flag. The change of low to high of this flag means the RGB data is going to transfer the first valid pixel data of a frame. The change of high to low means the last valid pixel data has been finished.

- 1 = RGB data transmission in progress, Display RAM is busy.
- 0 = No RGB data communication, Display RAM is free and can be updated.

**TCF** — One frame transmission completion flag. TCF flag can be cleared by writing one to this bit.

- 1 = One frame transmission is completed.
- 0 = One frame transmission is not completed.

**TCNF** — The specified number of frame transmission completion flag. TCNF flag can be cleared by writing one to this bit. This flag will be set period according to NFTTCR[ITR].

- 1 = The specified number of frame transmission is completed.
- 0 = The specified number of frame transmission is not completed.

**BEPO** — Big-endian pixel ordering.

- 1 = Big-endian pixel ordering, the first pixel data is stored in the upper half-word
- 0 = Little-endian pixel ordering, the first pixel data is stored in the lower half-word

**DRC** — Data frame format reverse control.

- 1 = data frame format: R5, G6, B5
- 0 = data frame format: B5, G6, R5

**TCIE** — TCF Interrupt Enable.

- 1 = TCF interrupt is enabled.
- 0 = TCF interrupt is disabled.

**TCNIE** — TCNF Interrupt Enable.

- 1 = TCNF interrupt is enabled.
- 0 = TCNF interrupt is disabled.



**RGBE** — The RGB Controller Enable Bit. The readable/writeable bit enables RGB Controller operation. When the RGB Controller is disabled, RGB data are in unknown state.

1 = RGB Controller is enabled.

0 = RGB Controller is disabled.

**Note:** To configure the polarity of the output signal, the polarity bit must be configured before configuring RGBE.

## 17.6. Function Description

The RGB controller is a slave on the IPS bus; it fetches graphic source information directly from memory and dynamically performs RGB interface timing before delivering data to a TFT LCD panel.

## 18. Blender Controller (BLDC)

### 18.1. Introduction

LT168's Blender is a slave on the internal IPS bus; it fetches graphic source information directly from memory and dynamically performs blending and bit-blitting operations before delivering data to a TFT LCD panel. The Blender features the following:

- Alpha Blending with 16-bits Resolution
- Inner Loop Offset
- The blender timing of SRAM is fetch twice and send once

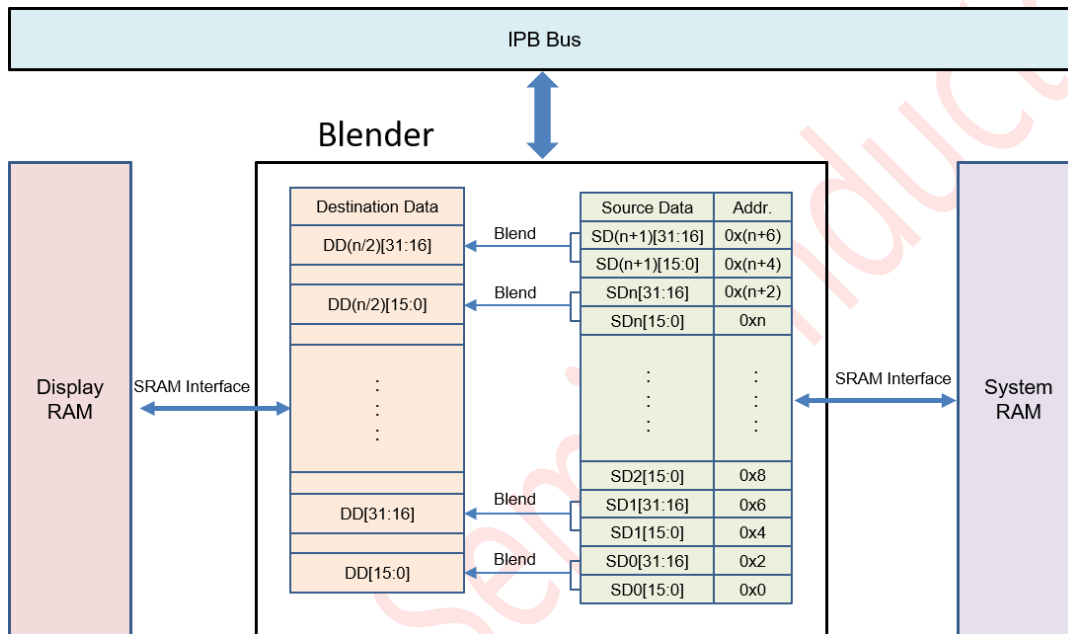


Figure 18-1: Blender Block Diagram

## 18.2. Blending Algorithm

**Table 18-1: Source Data Format**

bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
R0					G0						B0				
bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
α				R1				G1				B1			

**Table 18-2: Destination Data Format**

bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
R2					G2						B2				

The value of the blending process is derived from the following equation:

$$R2 = (R1 \ll 1) * \alpha + R0 * (0xF - \alpha);$$

$$G2 = (G1 \ll 2) * \alpha + G0 * (0xF - \alpha);$$

$$B2 = (B1 \ll 1) * \alpha + B0 * (0xF - \alpha);$$

## 18.3. Memory Map and Registers

This subsection describes the memory map and registers for the Blender. The Blender Module memory map is shown in **Table 18-3**, and its base address is **0x401B\_0000**.

### 18.3.1. Memory Map

Refer to **Table 18-3** for an overview of the blender memory map.

**Table 18-3: Blender Module Memory Map**

Offset Address	Bits[31:0]	Access
0x0000	Blender Control and Status Register(BCSR)	R/W
0x0004	Source Address Register(SADDR)	R/W
0x0008	Transfer Total Address Length Register (TTALR)	R/W
0x000C	Destination Address Register(DADDR)	R/W
0x0010	Transfer Destination Minor Length and Offset Register(TDMLOR)	R/W

18.3.2. Register Descriptions

18.3.2.1. Blender Control and Status Register (BCSR)

Address: Blender\_BASEADDR+0x0000\_0000

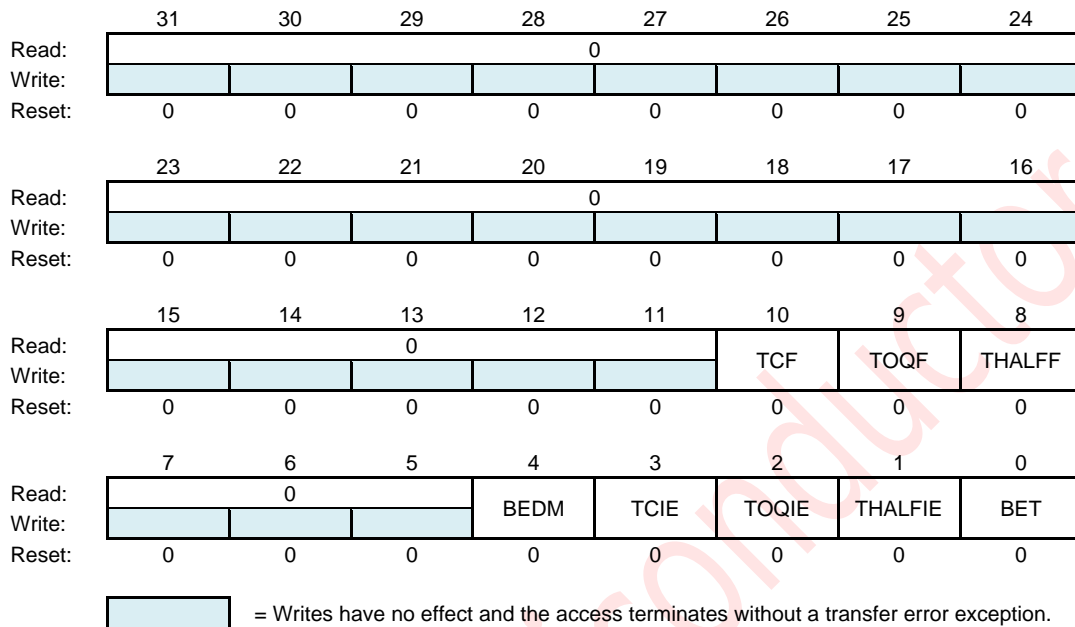


Figure 18-2: Blender Control and Status Register(BCSR)

**TCF** — Indicates the transfer of total source data for blending is completed. This flag will be cleared by writing one to it.

- 1 = The transfer of total source data for blending is completed
- 0 = The transfer of source data for blending is not completed

**TOQF** — Indicates the transfer of one quarter source data for blending is completed. This flag will be cleared by writing one to it

- 1 = The transfer of one quarter source data for blending is completed
- 0 = The transfer of one quarter source data for blending is completed

**THALFF** — Indicates the transfer of one half source data for blending is completed. This flag will be cleared by writing one to it

- 1 = The transfer of one half source data for blending is completed
- 0 = The transfer of one half source data for blending is completed

**BEDM** — Big-endian data mode for destination address.

- 1 = big-endian data mode, the first pixel data of blending is stored in the upper half-word
- 0 = little-endian data mode, the first pixel data of blending is stored in the lower half-word.

**TCIE** — TCF Interrupt Enable

- 1 = TCF interrupt is enabled
- 0 = TCF interrupt is disabled

**TOQIE** — TOQF Interrupt Enable

- 1 = TOQF interrupt is enabled
- 0 = TOQF interrupt is disabled

**THALFIE** — THALFF Interrupt Enable

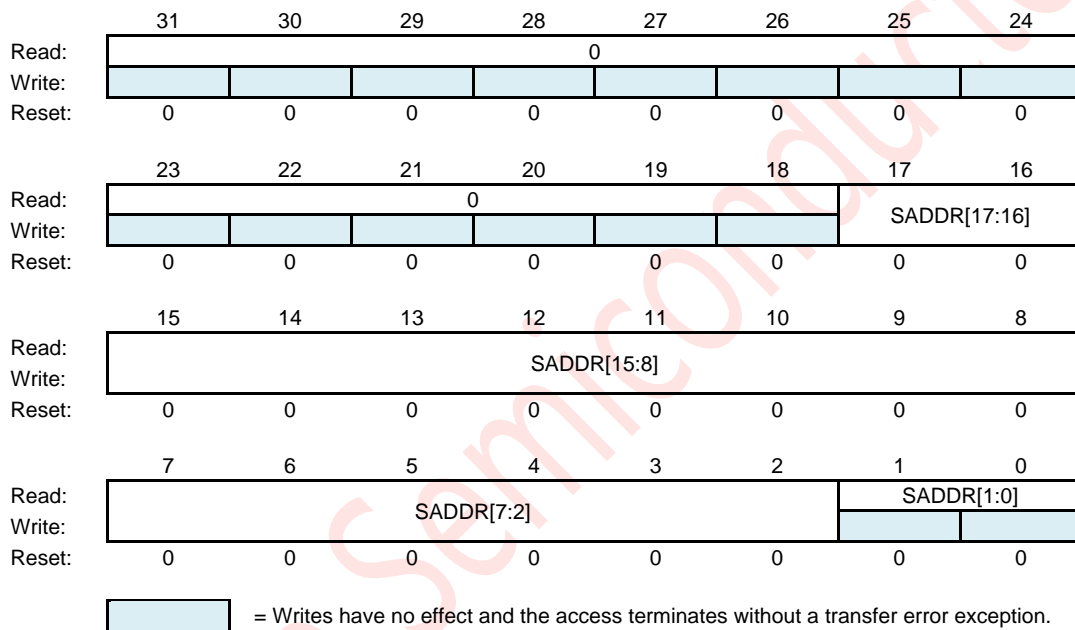
- 1 = THALFF interrupt is enabled
- 0 = THALFF interrupt is disabled

**BET** — The blender is enabled with a transfer state

- 1 = This bit is set by software to enter transfer state, the blender can start a new data transmission when resetting this bit
- 0 = This bit is cleared by hardware when the current transmission is completed,

**18.3.2.2. Source Address Register (SADDR)**

**Address: Blender\_BASEADDR+0x0000\_0004**

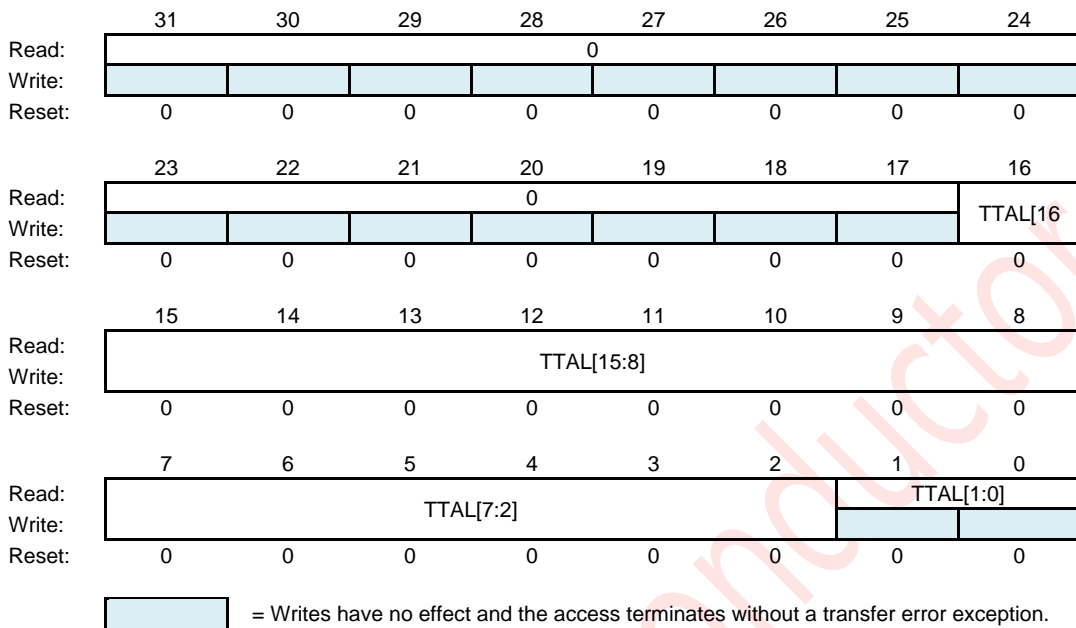


**Figure 18-3: Source Address Register(SADDR)**

**SADDR[17:0]** — Source offset address in 256K Bytes System RAM. SADDR[1:0] are fixed to 2'b00.

**18.3.2.3. Transfer Total Address Length Register (TTALR)**

**Address: Blender\_BASEADDR+0x0000\_0008**

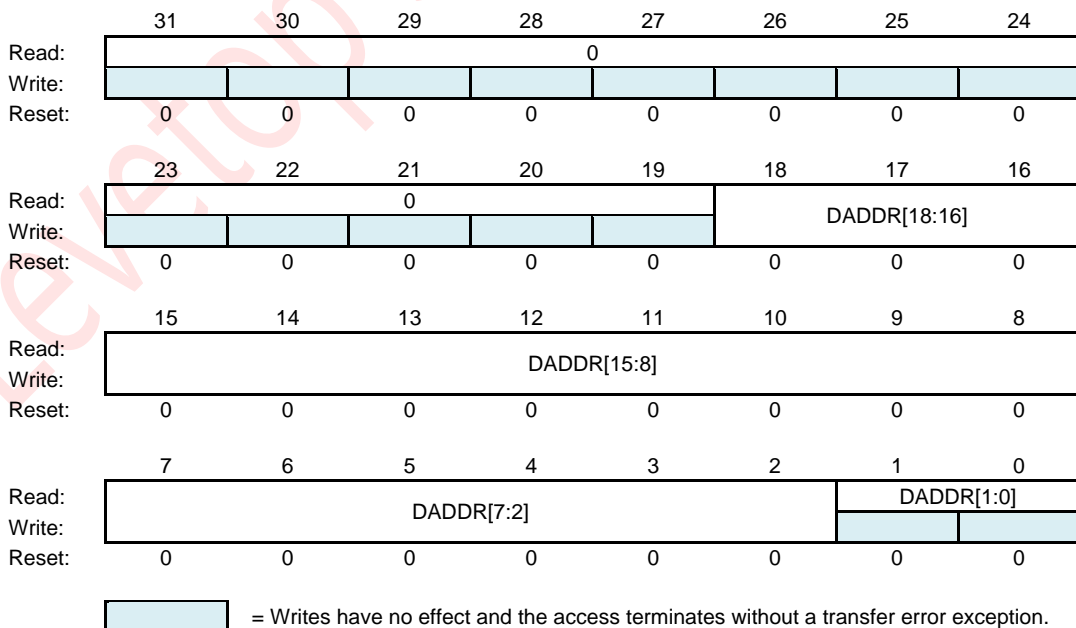


**Figure 18-4: Transfer Total Address Length Register (TTALR)**

**TTAL[16:0]** — Source Data Total Transfer Length in word. TTAL[1:0] are fixed to 2'b00.

**18.3.2.4. Destination Address Register (DADDR)**

**Address: Blender\_BASEADDR+0x0000\_000C**

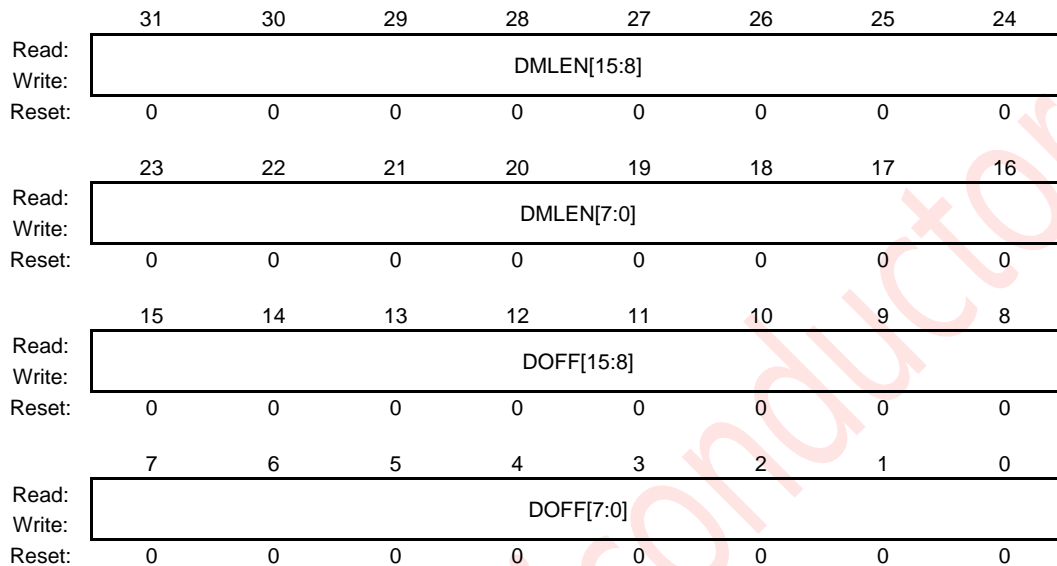


**Figure 18-5: Destination Address Register(DADDR)**

**DADDR[18:0]** — Destination offset address in 512K Bytes Display RAM. DADDR[1:0] are fixed to 2'b00.

**18.3.2.5. Transfer Destination Minor Length and Offset Register (TDMLOR)**

**Address: Blender\_BASEADDR+0x0000\_0010**



**Figure 18-6: Transfer Destination Minor Length and Offset Register (TDMLOR)**

**DMLEN[15:0]** — Destination minor loop length in word

**DOFF[15:0]** — Destination address signed offset in word. Sign-extended offset is applied to the current destination address to form the next-state value as each destination write is completed.

New destination offset address (when minor loop is completed) = DADDR[19:0] + n \* (DMLEN[15:0] + DOFF[15:0]), where n stands for minor loop counter.

## 19. External Bus Interface (EBI)

### 19.1. Introduction

LT168 has an embedded External Bus Interface (EBI) for controlling the transfer of the information between the internal bus and the external 8080 MCU Display Panel. One asynchronous chip select channel is available.

### 19.2. Features

Features of the External Bus 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

### 19.3. Signal Description

**Table 19-1:** Provides an overview of the signals described below.

**Table 19-1: Signals Properties**

Name	Function	Pullup
EBI_D[15:0]	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

### 19.4. Memory Map and Registers

#### 19.4.1. Memory Map

The EBI Module memory map is shown in **Table 19-2**, and its base address is **0x4018\_0000**.

**Table 19-2: Shows The EBI Register Memory Map.**

Address Offset	Bits[31:16]	Bits[15:0]	Access <sup>1,2</sup>
0x0000	CSCR0 — EBI Control Register		S
0x0004	Reserved		S
0x0008	Reserved		S
0x000C	IOPCR — IO Pull Control Register		S
0x0010	IOCR — IO Control Register	Reserved	S
0x0014	GPIODO — GPIO Data Output Register		S
0x0018	GPIODIR — GPIO Direction Register		S



Address Offset	Bits[31:16]	Bits[15:0]	Access <sup>1, 2</sup>
0x001C	GPIODIN — GPIO Data Input Register		S

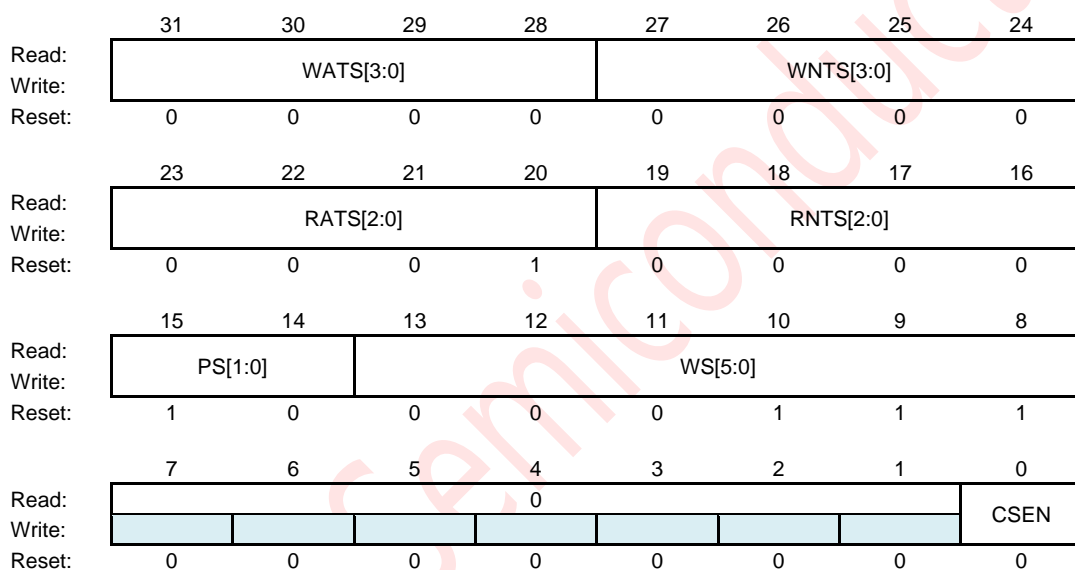
**Notes:**

1. S = CPU supervisor mode access only.
2. Accessing to supervisor-only address locations in user mode has no effect and result in a cycle termination transfer error.

**19.4.2. Register Descriptions**

**19.4.2.1. EBI Control Registers (EBICR)**

**Address: EBI\_BASEADDR+0x0000\_0000**



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

**Figure 19-1: EBI Control Register (EBICR)**

**WATS[3:0]** — Write Signal Assert Timing Select

These bits select the assertion timing of EBI\_WR#, when the operation is write. See functional description for details.

**WNTS[3:0]** — Write Signal Negate Timing Select

These bits select the negation timing of EBI\_WR#, when the operation is write. See functional description for details.

**RATS[3:0]** — Read Assert Timing Select

These bits select the assertion timing of EBI\_RD#, when the operation is read. See functional description for details.

**RNTS[3:0]** — Read Assert Timing Select

These bits select the negation timing of EBI\_RD#, when the operation is read. See functional description for details.

**PS[1:0]** — Port Size Bits

The PS bit defines the width of the external data port supported by the chip select as either 8-bits or 16-bits.

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

**WS[5:0]** — 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 64 possible wait states, even slow devices can be interfaced with the MCU.

**Table 19-3: EBI\_CS# Chip Select Wait States Encoding**

WS[5:0]	Number of EBI_CS# Cycles	
	Read Access	Write Access
000000	1	1
000001	2	2
000010	3	3
000011	4	4
000100	5	5
000101	6	6
000110	7	7
000111	8	8
001000	9	9
001001	10	10
001010	11	11
001011	12	12
001100	13	13
001101	14	14
.....	.....	.....
111111	64	64

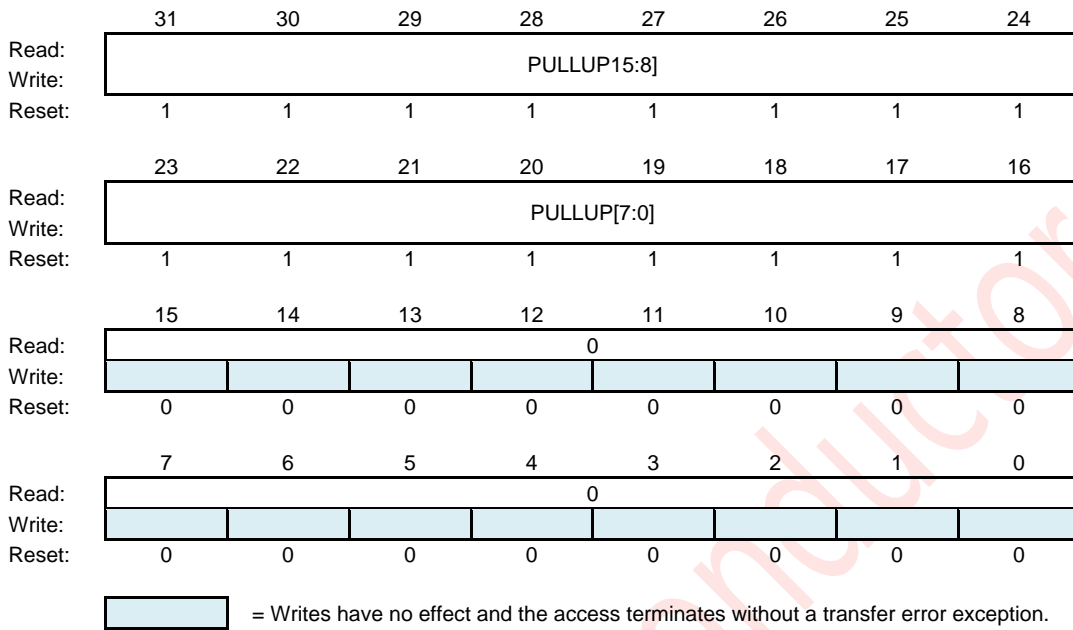
**CSEN** — Chip Select Enable Bit

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 is enabled.
- 0 = Chip select function is disabled.

**19.4.2.2. IO Pull Control Register (IOPCR)**

**Address: EBI\_BASEADDR+0x0000\_000C**



**Figure 19-2: IO Pull Control Register(IOPCR)**

**PULLUP[15:0]** — EBI\_D[15:0] Pull up enable control bits when used as GPIO input mode.

1 = EBI\_D[15:0] port pull up is enabled

0 = EBI\_D[15:0] port pull up is disabled

19.4.2.3. IO Control Register (IOCR)

Address: EBI\_BASEADDR+0x0000\_0010

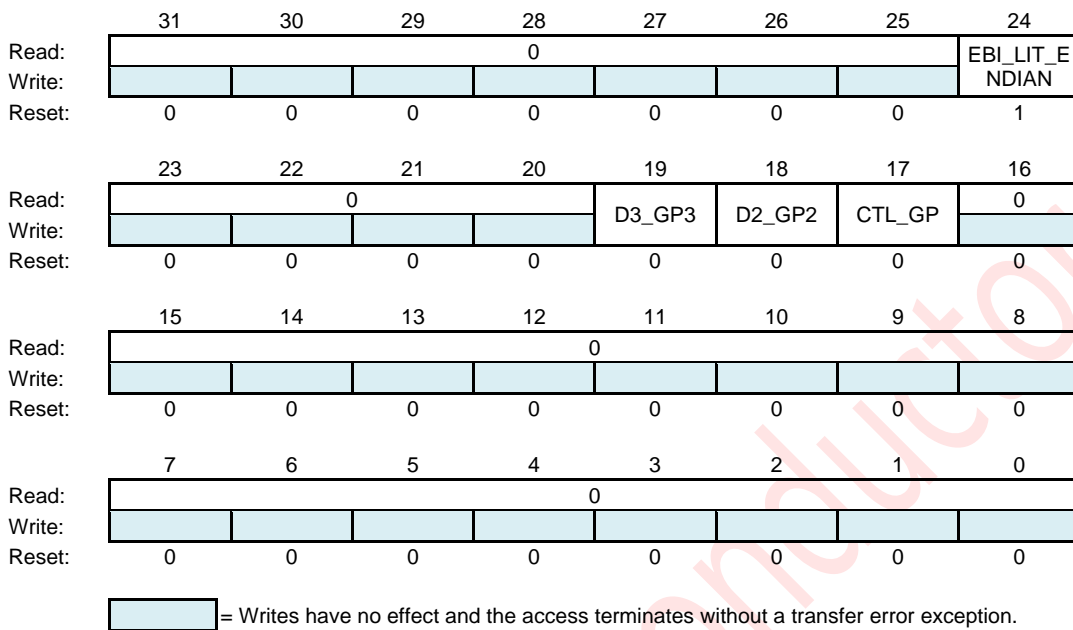


Figure 19-3: IO Control Register(IOCR)

**EBI\_LIT\_ENDIAN** — EBI ports is little endian

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

**D3\_GP3** — D[15:8] port used as GPIO

Writing this bit determines the use of the D[15:8] port.

- 1 = D[15:8] port is used as GPIO
- 0 = D[15:8] port is used as data bus for display panel

**D2\_GP2** — EBI\_D[7:0] port used as GPIO

Writing this bit determines the use of the EBI\_D[7:0] port.

- 1 = EBI\_D[7:0] port is used as GPIO
- 0 = EBI\_D[7:0] port is used as data bus for display panel

**CTL\_GP** — EBI\_WR#, EBI\_RD#, EBI\_CS#, EBI\_RS pin as EPORT1 function

Table 19-4: EBI Control Pin as EPORT1

CTL_GP = 1'b0	CTL_GP = 1'b1
EBI_CS#	INT1[5]
EBI_RD#	INT1[4]
EBI_WR#	INT1[2]
EBI_RS	INT1[1]

**19.4.2.4. GPIO Data Output Register (GPIODO)**

**Address: EBI\_BASEADDR+0x0000\_0014**



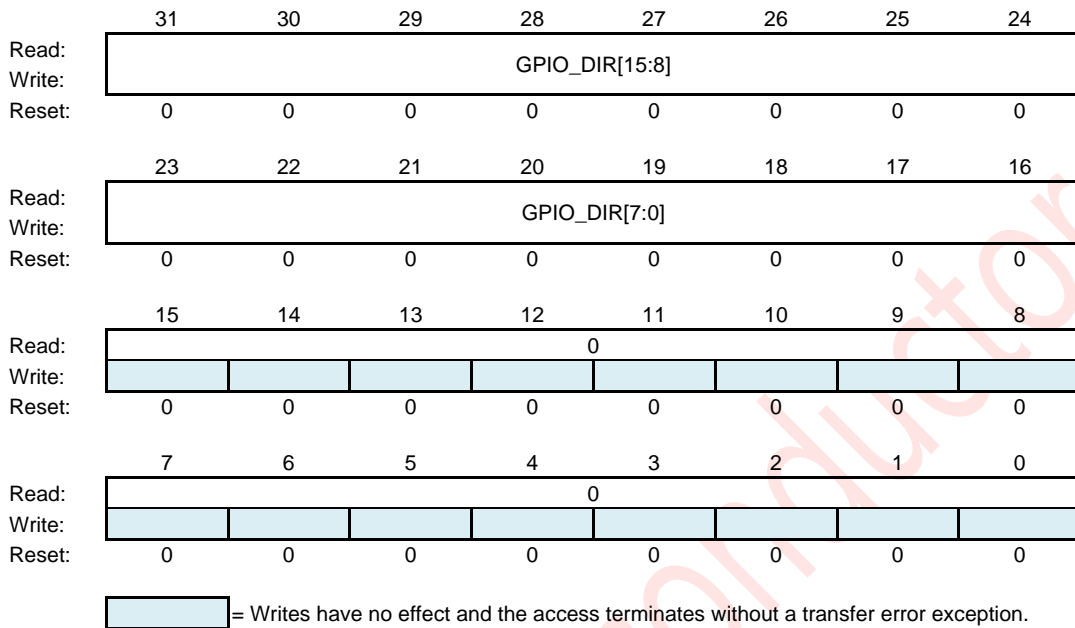
**Figure 19-4: GPIO Data Output Register(GPIODO)**

**DO[15:0]** — Output data of EBI\_D[15:0] pin

Writing these bits determines the output data of EBI\_D[15:0] pin when used as GPIO.

**19.4.2.5. GPIO Direction Register (GPIODIR)**

**Address: EBI\_BASEADDR+0x0000\_0018**



**Figure 19-5: GPIO Direction Register(GPIODIR)**

**GPIO\_DIR[15:0]** — GPIO Pin's Direction

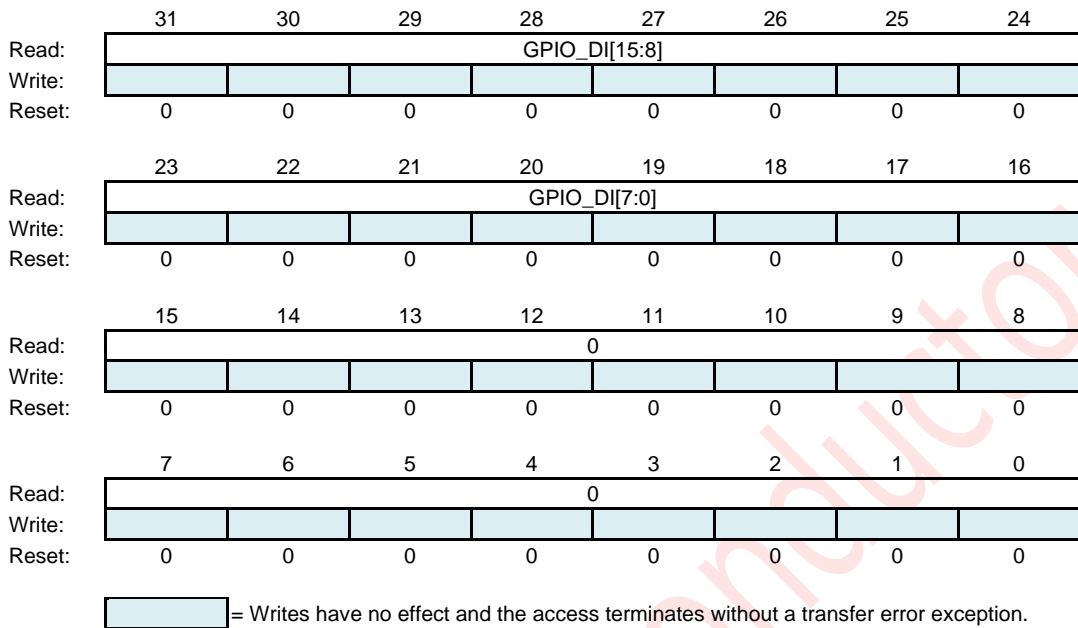
Writing these bits determines the direction of GPIO pins.

1 = GPIO is output

0 = GPIO is input

**19.4.2.6. GPIO Data Input Register (GPIODI)**

**Address: EBI\_BASEADDR+0x0000\_001C**



**Figure 19-6: GPIO Data Input Register(GPIODI)**

**GPIO\_DI[15:0]** — GPIO's Input Data

Reading these bits reflects the status of EBI\_D[15:0] pins when used as GPIO.

**19.5. Functional Description**

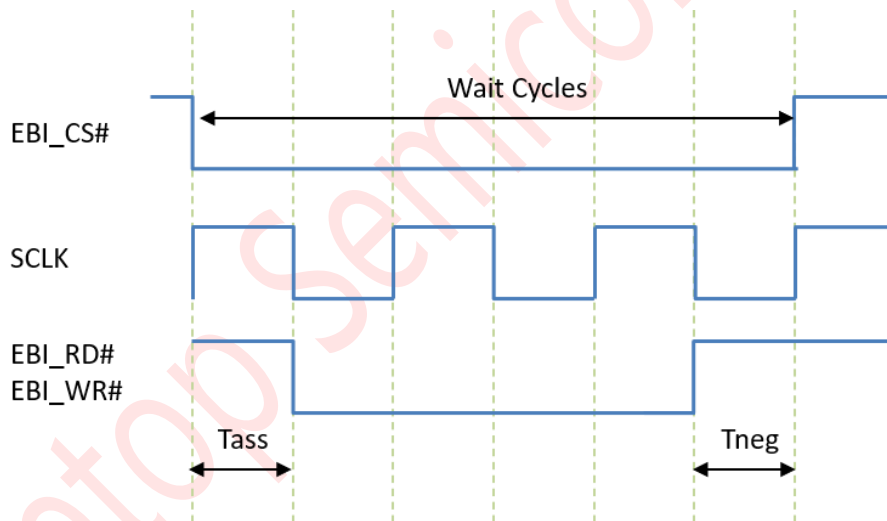
**19.5.1. EBI\_WR#, EBI\_RD# Signal Timing**

For write and read operation, EBI\_WR# and EBI\_RD# assert and negate timing can be configured.

**Table 19-5: EBI\_WR# and EBI\_RD# Assert and Negate Timing**

RATS[3:0]/WATS[3:0]	Assert Time (T <sub>ass</sub> )	RNTS[3:0]/WNTS[3:0]	Negate Time (T <sub>neg</sub> )
0000	1/2	0000	1/2
0001	2/2	0001	2/2
0010	3/2	0010	3/2
0011	4/2	0011	4/2
0100	5/2	0100	5/2
.....	.....	.....	.....
1110	15/2	1110	15/2
1111	16/2	1111	16/2

Unit: system cycle



**Figure 19-7: EBI\_WR#, EBI\_RD# Assert/Negate Timing**



**19.5.2. EBI Functional Description**

**19.5.2.1. EBI\_CS# Chip Select**

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, is determined by the wait state (WS) field, and the port size (PS) field is used for the access.

**Table 19-6: Chip Select Address Range Encoding**

Chip Select	Block Size	Normal Mode Address Range
EBI_CS#	1M Bytes	0x2000_0000-0x2FFF_FFFF

**19.5.2.2. 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 8-bits or 16-bits 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 8-bits port is assigned to pins D[31:24], and a 16-bits port is assigned to pins D[31:16].

**Table 19-7** to **Table 19-8** shows each possible transfer size, alignment, and port width of EBI\_CS#. The data bytes shown in the table represent external data pins. These data are 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 drive them with undefined data on write transfers.

**Table 19-7: 8-bits 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 19-8: 16-bits 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 <sup>1</sup>	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 16-bits ports. The table shows the data placement for both bus cycles.

## 20. Programmable Interrupt Timer (PIT)

### 20.1. Introduction

LT168's Programmable Interrupt Timer (PIT) is a 16-bits 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.

### 20.2. Block Diagram

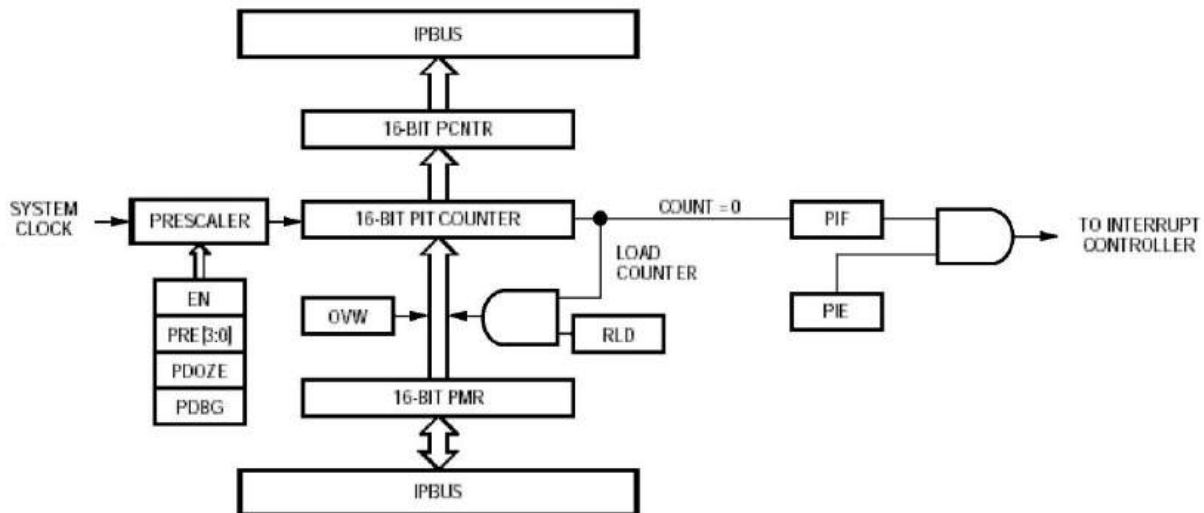


Figure 20-1: PIT Block Diagram

### 20.3. Modes of Operation

This subsection describes the three low-power modes and the debug mode.

#### 20.3.1. 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.

#### 20.3.2. 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.

#### 20.3.3. Stop Mode

In stop mode, the system clock is absent, and PIT module operation stops.

#### 20.3.4. Debug Mode

In debug mode with the PDBG bit set in PCSR, PIT module operation stops. In debug mode with the PDBG bit cleared, 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.

## 20.4. Signals

The PIT module has no off-chip signals.

## 20.5. Memory Map and Registers

This subsection describes the memory map and register structure for PIT. The PIT Module memory map is shown in **Table 20-1**. The PIT0 base address is **0x4004\_0000**, PIT1 is **0x4005\_0000**, PIT2 is **0x4006\_0000**, and PIT3 is **0x4007\_0000**.

### 20.5.1. Memory Map

Refer to **Table 20-1** for a description of the memory map.

This device has four programmable interrupt timers.

**Table 20-1: Programmable Interrupt Timer Module Memory Map**

PITn Address	Bits[15:0]	Access <sup>(1)</sup>
0x0	PIT Modulus Register (PMR)	S
0x2	PIT Control and Status Register (PCSR)	S
0x4	Unimplemented <sup>(2)</sup>	—
0x6	PIT Count Register (PCNTR)	S/U

**Notes:**

- (1) S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. Accessing to supervisor only addresses in user mode has 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.

### 20.5.2. Register Descriptions

The PIT programming model consists of below registers:

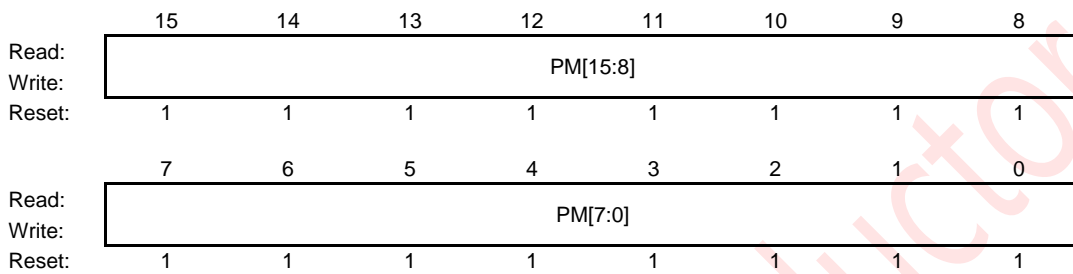
- The PIT Control and Status Register (PCSR) configures the timer's operation. See **Section 20.5.2.2**.
- The PIT Modulus Register (PMR) determines the timer modulus reload value. See **Section 20.5.2.1**.
- The PIT Count Register (PCNTR) provides visibility to the counter value. See **Section 20.5.2.3**.

**20.5.2.1. PIT Modulus Register (PMR)**

The 16-bits 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.

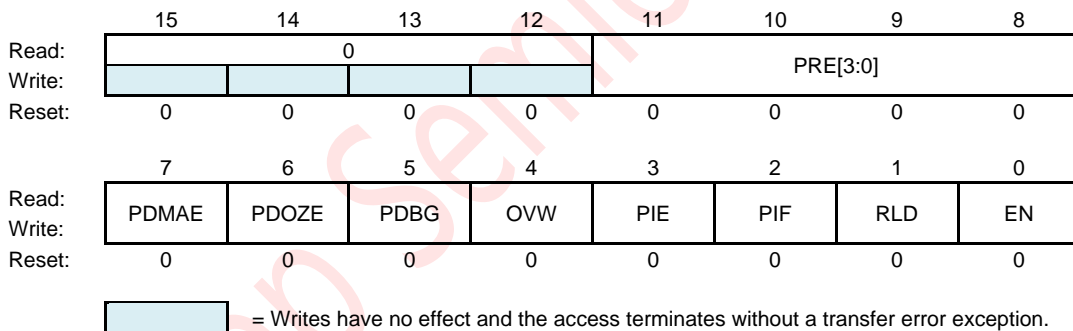
**Address: PITn\_BASEADDR+0x0000\_0000**



**Figure 20-2: PIT Modulus Register (PMR)**

**20.5.2.2. PIT Control and Status Register (PCSR)**

**Address: PITn\_BASEADDR+0x0000\_0002**



**Figure 20-3: 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 20-2** shows.

To accurately predict the timing of the next count, change the PRE[3:0] bits only when the enable bit (EN) is cleared. 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 20-2: Prescaler Select Encoding**

PRE[3:0]	IPS Clock Divisor
0000	1
0001	2
0010	4
0011	8

PRE[3:0]	IPS Clock Divisor
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 Control Bit

The read/write PDMAE bit controls 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 will not 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 is stopped in doze mode
- 0 = PIT function is 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 is stopped in debug mode
- 0 = PIT function is 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 the 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 is enabled
- 0 = PIF interrupt requests is 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 is reloaded from PMR on count of 0x0000
- 0 = Counter rolls over to 0xFFFF on count of 0x0000

**EN** — PIT Enable Bit

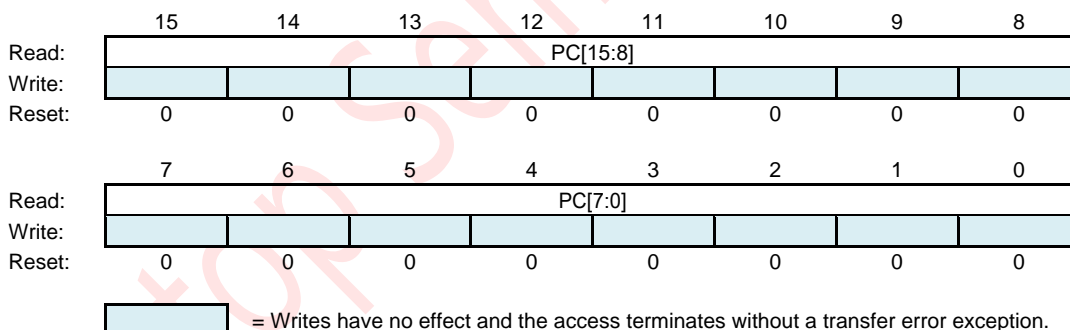
The read/write EN bit enables PIT operation. When the PIT is disabled, the counter and prescaler are held in a stopped state.

- 1 = PIT is enabled
- 0 = PIT is disabled

**20.5.2.3. PIT Count Register (PCNTR)**

The 16-bits, read-only PIT Control Register (PCNTR) contains the counter value. Reading the 16-bits counter with two 8-bits reads is not guaranteed to be coherent. Writing to PCNTR has no effect, and write cycles are terminated normally.

**Address: PITn\_BASEADDR+0x0000\_0006**



**Figure 20-4: PIT Count Register (PCNTR)**

## 20.6. Functional Descripiton

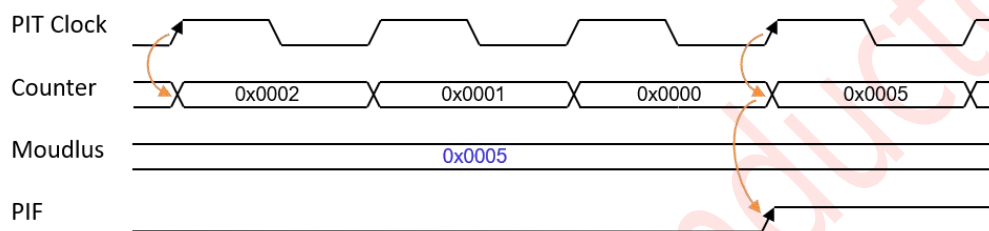
This subsection describes the PIT functional operation.

### 20.6.1. 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.



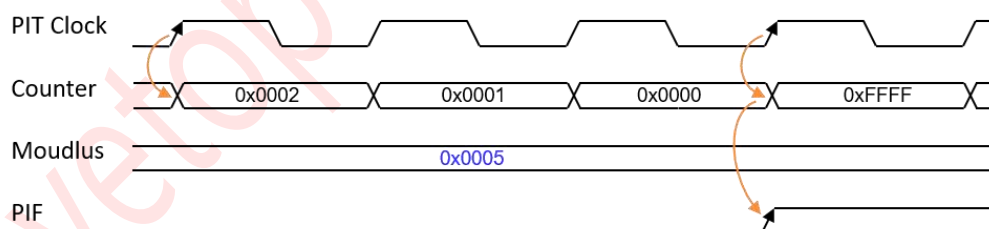
**Figure 20-5: Counter Reloading from the Modulus Latch**

### 20.6.2. Free-Running Timer Operation

This mode of operation is selected when the RLD bit in PCSR is cleared. 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 20-6: Counter in Free-Running Mode**

### 20.6.3. Timeout Specifications

The 16-bits 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.

$$\text{timeout period} = \text{PRE}[3:0] \times (\text{PM}[15:0] + 1) \text{ clocks}$$



## 20.7. Interrupt Operation

Table 20-3 lists the interrupt requests generated by the PIT.

Table 20-3: PIT Interrupt Requests

Interrupt Request	Flag	Enable Bit
Timeout	PIF	PIE

Levetop Semiconductor

## **21. Watchdog Timer (WDT)**

### **21.1. Introduction**

LT168's Watchdog Timer is a 16-bits 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.

### **21.2. Modes of Operation**

This subsection describes the operation of the watchdog timer in low-power modes and debug mode of operation.

#### **21.2.1. 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 cleared, the watchdog timer continues to operate normally.

#### **21.2.2. Doze Mode**

In doze mode with the DOZE bit set in WCR, watchdog timer module operation stops. In doze mode with the DOZE bit cleared, the watchdog timer continues to operate normally.

#### **21.2.3. 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 cleared, the watchdog timer continues to operate normally.

#### **21.2.4. Debug Mode**

In debug mode with the DBG bit set in WCR, watchdog timer module operation stops. In debug mode with the DBG bit cleared, 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.

### 21.3. Block Diagram

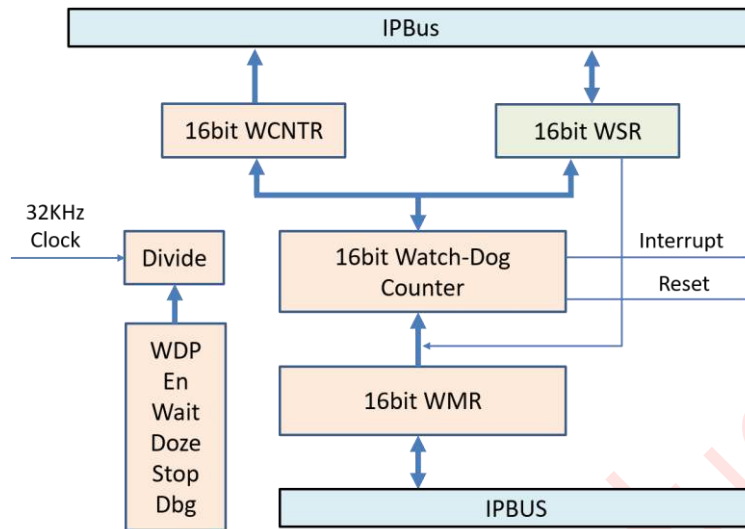


Figure 21-1: Watchdog Timer Block Diagram

### 21.4. Signals

The watchdog timer module has no off-chip signals.

### 21.5. Memory Map and Registers

This subsection describes the memory map and registers for the Watchdog Timer. The WDT Module base address is 0x4013\_0000.

#### 21.5.1. Memory Map

Refer to Table 21-1 for an overview of the watchdog memory map.

Table 21-1: Watchdog Timer Module Memory Map

Offset Address	Bits[15:0]	Access <sup>(1)</sup>
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 (1)** : S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. Accessing to supervisor only addresses in user mode has no effect and result in a cycle termination transfer error.

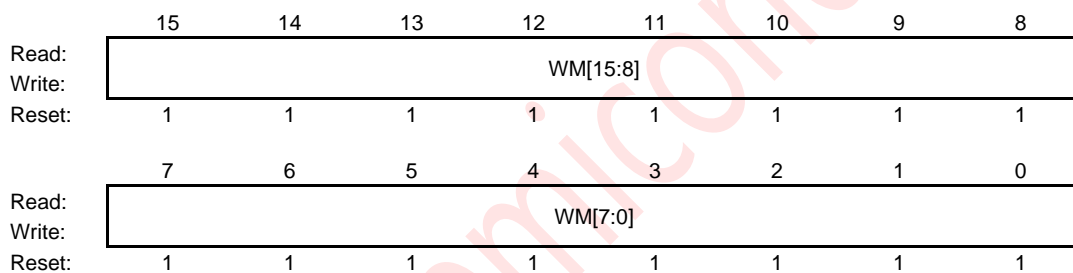
**21.5.2. Register Description**

The watchdog timer programming model consists of these registers:

- The Watchdog Control Register (WCR) configures watchdog timer operation. See **Section 21.5.2.2**.
- The Watchdog Modulus Register (WMR) determines the timer modulus reload value. See **Section 21.5.2.1**.
- The Watchdog Count Register (WCNTR) provides visibility to the watchdog counter value. See **Section 21.5.2.4**.
- The Watchdog Service Register (WSR) requires a service sequence to prevent reset. The read-only WC[15:0] field reflects the current value in the watchdog counter. Reading the 16-bits WCNTR with two 8-bits 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 may not be stable, please read it time after time continuously..

**21.5.2.1. Watchdog Modulus Register (WMR)**

**Address: WDT\_BASEADDR+0x0000\_0000**



**Figure 21-2: 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 0xFFFF.

21.5.2.2. Watchdog Control Register (WCR)

The 16-bits read/write Watchdog Control Register (WCR) configures watchdog timer operation.

Address: WDT\_BASEADDR+0x0000\_0002

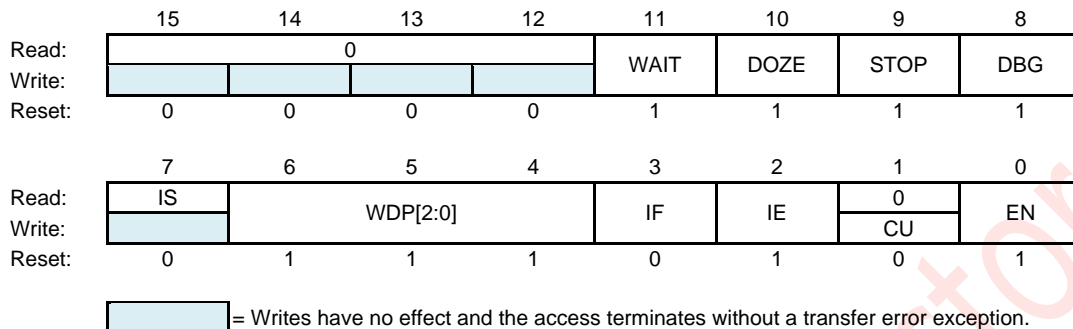


Figure 21-3: Watchdog Control Register (WCR)

**WAIT** — Wait Mode Bit

WAIT bit controls the function of the watchdog timer in wait mode. Reset sets WAIT.

- 1 = Watchdog timer is stopped in wait mode
- 0 = Watchdog timer is 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 is stopped in doze mode
- 0 = Watchdog timer is 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 is stopped in stop mode
- 0 = Watchdog timer is 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 is stopped in debug mode
- 0 = Watchdog timer is 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.

**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 wants to sleep or stop, it will be wakeup again. Therefore, before CPU wants to sleep or stop, 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 21-2** Watchdog Timer Prescaler.

**Table 21-2: 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

**IF** — Watchdog Interrupt Flag Bit

Write One to this bit will clear the flag.

**IE** — Watchdog Interrupt Enable Bit

IE bit enables the watchdog timer interrupt mode. Once interrupt is generated and the EN bit is 1, this bit will be auto cleared.

1 = Watchdog timer interrupt mode is enabled

0 = Watchdog timer interrupt mode is disabled

**CU** — Watchdog Change Update Bit

Writing One to CU bit will update the WDP[2:0] and WMR to the work latch.

**EN** — Watchdog Enable Bit

EN bit enables the watchdog timer.

1 = Watchdog timer is enabled

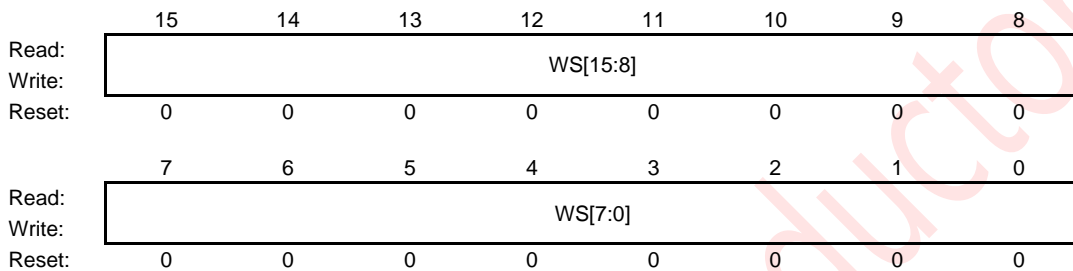
0 = Watchdog timer is disabled

**21.5.2.3. Watchdog Service Register (WSR)**

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, both values are required to be written to prevent the watchdog timer from causing a reset.

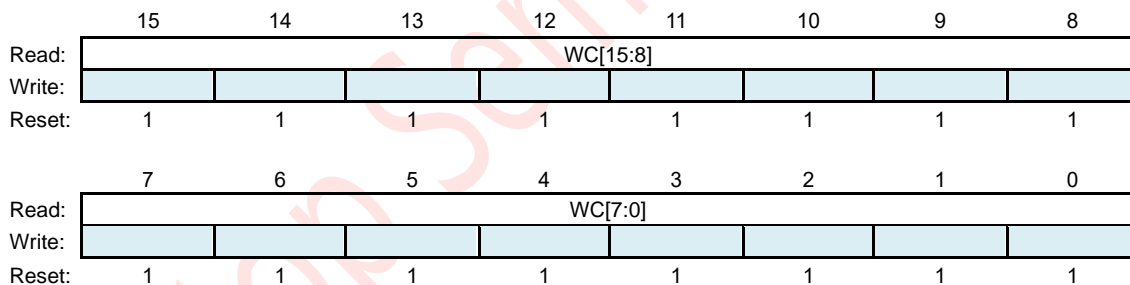
**Address: WDT\_BASEADDR+0x0000\_0004**



**Figure 21-4: Watchdog Service Register (WSR)**

**21.5.2.4. Watchdog Count Register (WCNTR)**

**Address: WDT\_BASEADDR+0x0000\_0006**



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

**Figure 21-5: 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 16-bits WCNTR with two 8-bits 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 may not be stable, please read it time after time continuously.

## 22. Real Time Controller (RTC)

### 22.1. Introduction

LT168's Real Time Controller controls a comprehensive PMU with RTC function. It can reconfigure RTC counter, set alarms, and generate time/alarm interrupts.

### 22.2. Features

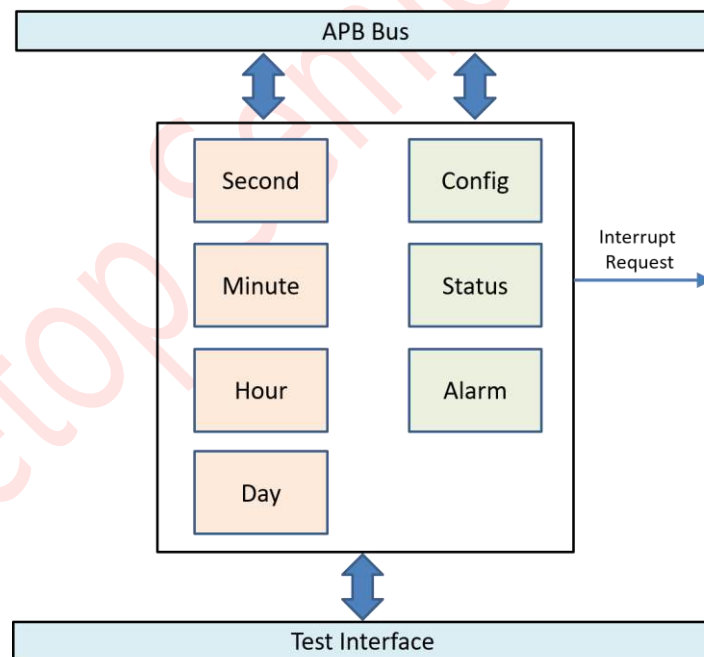
The main features of the module:

- 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 .

### 22.3. Test Mode

In test mode, RTC can be configured by CPU or by chip pins, and RTC status and clock signals can be checked by chip pins.

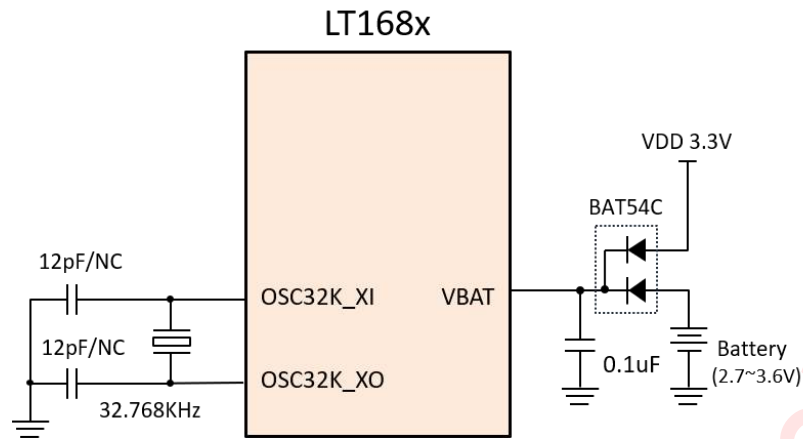
### 22.4. Block Diagram



**Figure 22-1: RTC Block Diagram**



**22.5. Application Circuit**

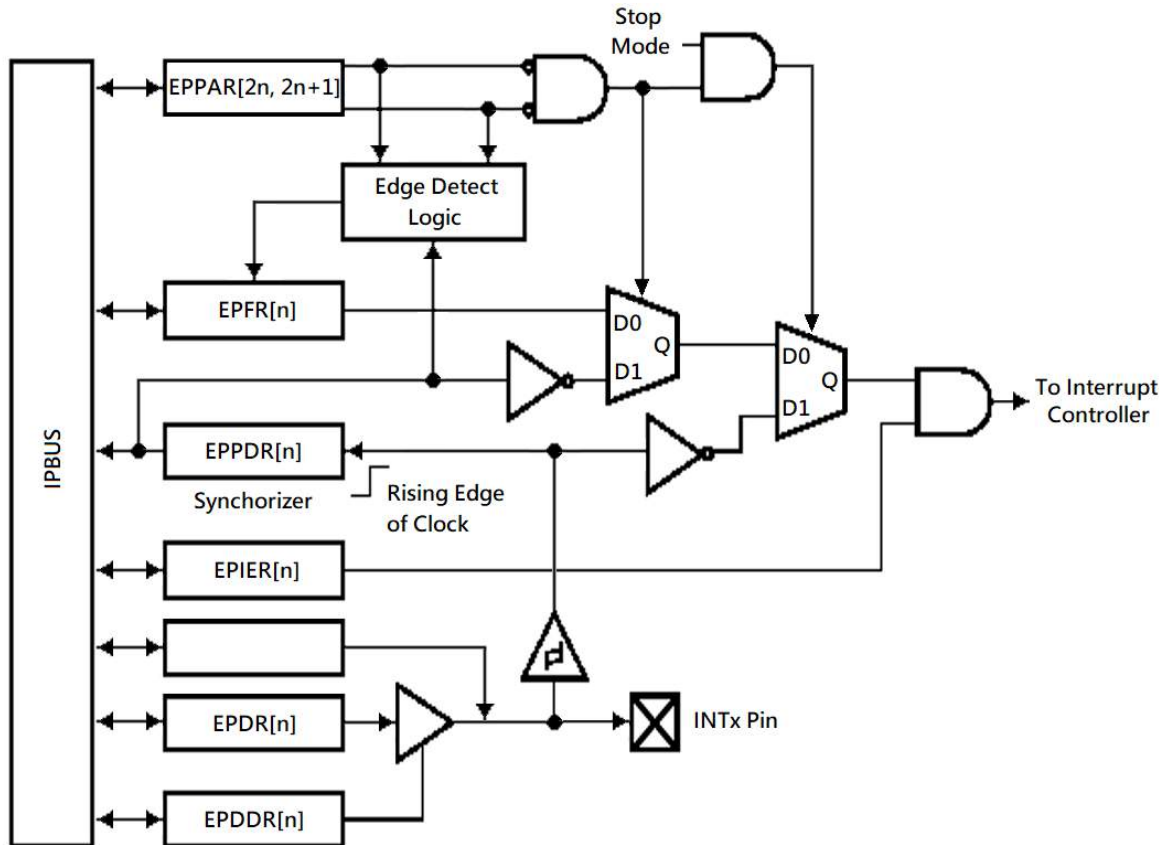


**Figure 22-2: RTC Application Circuit**

## 23. Edge Port Module (EPORT)

### 23.1. Introduction

LT168 has three Edge Port Modules (EPORT). And each EPORT has eight corresponding 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 23-1**.



**Figure 23-1: EPORT Block Diagram**

### 23.2. Low-Power Mode Operation

This subsection describes the operation of the EPORT module in low-power modes.

#### 23.2.1. 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.

#### 23.2.2. Stop Mode

In stop mode, there is 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 is available.

### 23.3. 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 reading 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.

### 23.4. Memory Map And Registers

The EPORT Module memory map is shown in **Table 23-1**. The EPORT0 base address is **0x400F\_0000**, EPORT1 is **0x4010\_0000**, and EPORT2 is **0x401D\_0000**. This subsection describes the memory map and register structure.

#### 23.4.1. Memory Map

Refer to **Table 23-1** for a description of the EPORT memory map.

**Table 23-1: Module Memory Map**

Address Offset	Bits[15:8]	Bits[7:0]	Access <sup>(1)</sup>
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	Reserved		S
0x000E	EPORT Bit Clear Register (EPBCR)	Reserved	S

**Note (1)** : S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. Accessing to supervisor only addresses in user mode has no effect and result in a cycle termination transfer error.

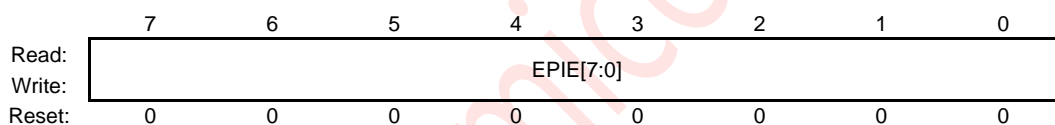
### 23.4.2. Register Description

The EPORT programming model consists of below 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) enables the filter and controls the width of input pulse that will be filtered.

#### 23.4.2.1. Edge Port Interrupt Enable Register (EPIER)

**Address: EPORTn\_BASEADDR+0x0000\_0000**



**Figure 23-2: 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

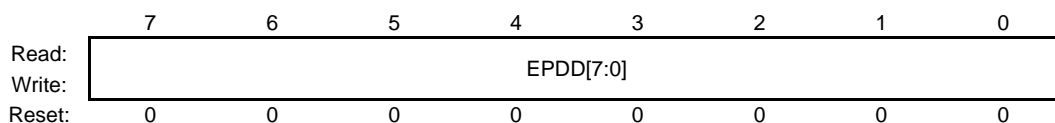
Clearing a bit in EPIER negates any interrupt request from the corresponding EPORT pin. Reset clears EPIE[7:0].

1 = Interrupt requests from corresponding EPORT pin is enabled

0 = Interrupt requests from corresponding EPORT pin is disabled

#### 23.4.2.2. Eport Data Direction Register (EPDDR)

**Address: EPORTn\_BASEADDR+0x0000\_0001**



**Figure 23-3: 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 is configured as output
- 0 = Corresponding EPORT pin is configured as input

**23.4.2.3. Eport Pin Assignment Register (EPPAR)**

**Address: EPORTn\_BASEADDR+0x0000\_0002**

	15	14	13	12	11	10	9	8
Read:	EPPA7[1:0]		EPPA6[1:0]		EPPA5[1:0]		EPPA4[1:0]	
Write:								
Reset:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
Read:	EPPA3[1:0]		EPPA2[1:0]		EPPA1[1:0]		EPPA0[1:0]	
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 23-4: EPORT Pin Assignment Register (EPPAR)**

**EPPAx[1:0]** — EPORT Pin Assignment Select Fields

The read/write EPPAx fields configure EPORT pins for level detection and rising and/or falling edge detection as shown in Table 23-2.

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 no matter it is configured as input or output.

**Table 23-2: EPPAx Field Settings**

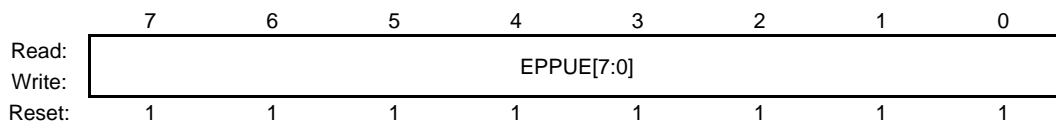
<b>EPPAx</b>	<b>Pin Configuration</b>
00	Pin INTx level-sensitive
01	Pin INTx rising edge triggered
10	Pin INTx falling edge triggered
11	Pin INTx both falling edge and rising edge triggered

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.

**23.4.2.4. Eport Pin Pull-up Enable Register (EPPUE)**

**Address: EPOR<sub>Tn</sub>\_BASEADDR+0x0000\_0004**



**Figure 23-5: EPOR<sub>T</sub> Pin Pull-up enable Register (EPPUE)**

**EPPUE[7:0]** — Edge Port Pin Pull-up enable Bits

Setting any bit in the EPPUE configures the corresponding pin to enable pull-up. The setting is available only when EPOR<sub>T</sub> are in input mode. Clearing any bit in EPPUE configures the corresponding pin to disable pull-up.

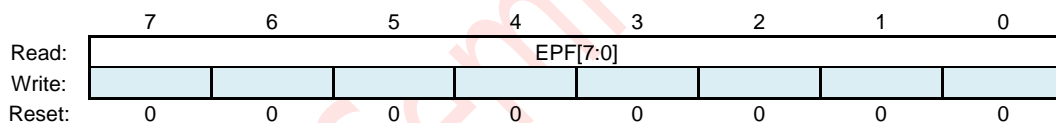
Reset sets EPPUE[7:0].

1 = Corresponding EPOR<sub>T</sub> pin configured to enable pull-up

0 = Corresponding EPOR<sub>T</sub> pin configured to disable pull-up

**23.4.2.5. Edge Port Flag Register (EPFR)**

**Address: EPOR<sub>Tn</sub>\_BASEADDR+0x0000\_0005**



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

**Figure 23-6: EPOR<sub>T</sub> Port Flag Register (EPFR)**

**EPF[7:0]** — Edge Port Flag Bits

When an EPOR<sub>T</sub> 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].

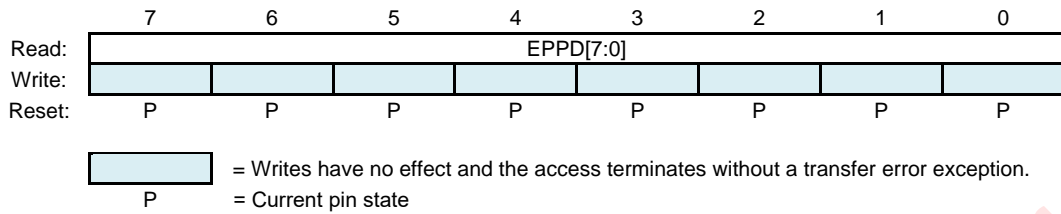
1 = Selected edge for INT<sub>x</sub> pin has been detected.

0 = Selected edge for INT<sub>x</sub> 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 (EPPAR<sub>x</sub> = 00), pin transitions do not affect this register.

**23.4.2.6. Edge Port Pin Data Register (EPPDR)**

**Address: EPOR<sub>Tn</sub>\_BASEADDR+0x0000\_0006**



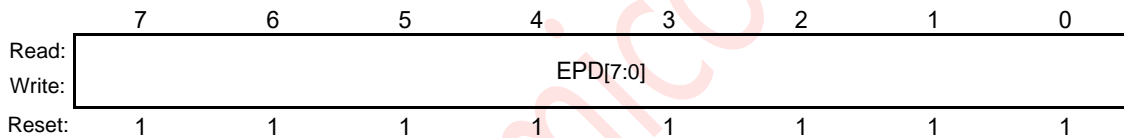
**Figure 23-7: EPORT Port Pin Data Register (EPPDR)**

**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.

**23.4.2.7. Edge Port Data Register (EPDR)**

**Address: EPOR<sub>Tn</sub>\_BASEADDR+0x0000\_0007**



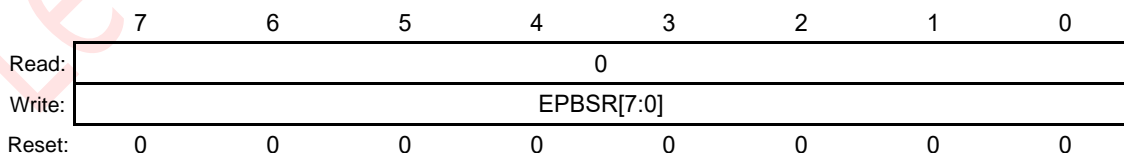
**Figure 23-8: EPORT Port Data Register (EPDR)**

**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 EPDR returns the data stored in the register. Reset sets EPD[7:0].

**23.4.2.8. Eport Port Bit Set Register (EPBSR)**

**Address: EPOR<sub>Tn</sub>\_BASEADDR+0x0000\_0008**



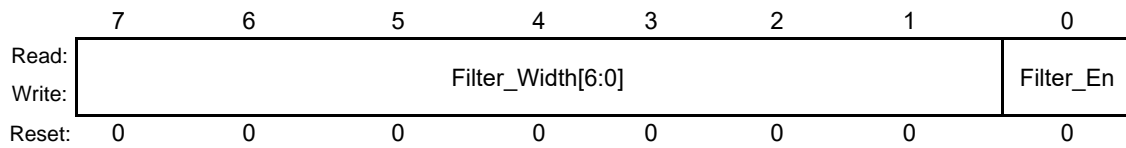
**Figure 23-9: EPORT Port Bit Set Register (EPBSR)**

**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 effected;

**23.4.2.9. Eport Digital Filter Control Register (EPFC)**

**Address: EPORn\_BASEADDR+0x0000\_0009**



**Figure 23-10: EPORn Digital Filter Control Register (EPFC)**

**Filter\_Width[6:0]** — EPORn Digital Filter Pulse Width Select Bit

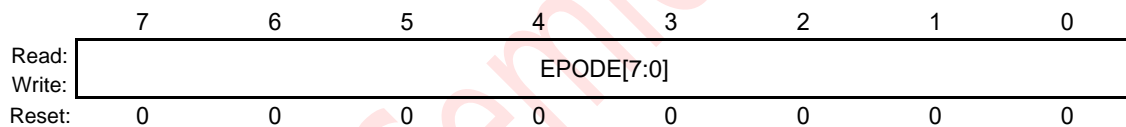
Filter\_Width[6:0] determine the width of the input pulse to be filtered. If the input pulse width is less than (Filter\_Width[6:0]+2), the pulse will be filtered.

**Filter\_En** — EPORn Digital Filter Enable Bit

- 1 = EPORn digital filter is enabled;
- 0 = EPORn digital filter is disabled;

**23.4.2.10. Eport Open Drain Enable Register (EPODE)**

**Address: EPORn\_BASEADDR+0x0000\_000A**



**Figure 23-11: EPORn Open Drain enable Register (EPODE)**

**EPODE[7:0]** — Edge Port Open Drain enable Bits

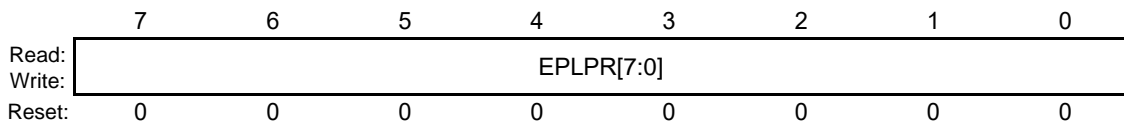
If EPORn are configured to output, setting any bit in the EPODE configures the corresponding pin to Open Drain output. Clearing any bit in EPODE configures the corresponding pin to CMOS output. Reset clears EPODE[7:0].

- 1 = Corresponding EPORn pin is configured to Open Drain output
- 0 = Corresponding EPORn pin is configured to CMOS output



**23.4.2.11. Eport Level Polarity Register (EPLPR)**

**Address: EPOR<sub>Tn</sub>\_BASEADDR+0x0000\_0000B**



**Figure 23-12: EPOR<sub>T</sub> Level Polarity Register (EPLPR)**

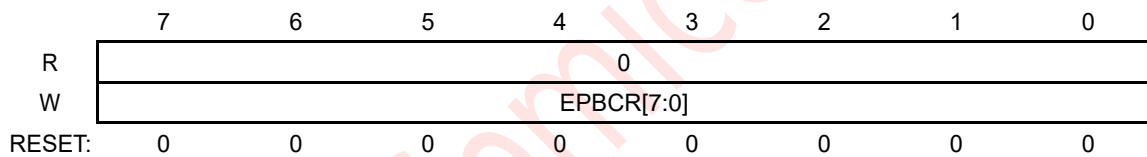
**EPLPR[7:0]** — Edge Port Level Polarity Bits

If EPOR<sub>T</sub> 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 EPOR<sub>T</sub> pin is configured to high level-sensitive
- 0 = Corresponding EPOR<sub>T</sub> pin is configured to low level-sensitive

**23.4.2.12. Eport Port Bit Clear Register (EPBCR)**

**Address: EPOR<sub>Tn</sub>\_BASEADDR+0x0000\_000F**



**Figure 23-13: EPOR<sub>T</sub> Port Bit Clear Register (EPBCR)**

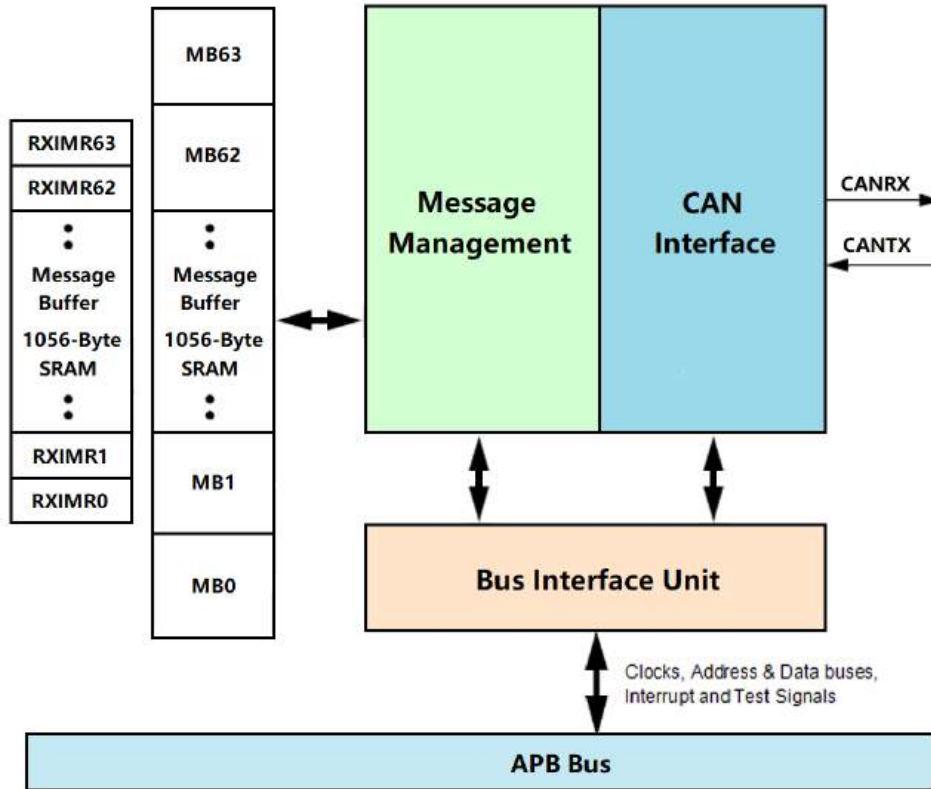
**EPBCR[7:0]** — EPOR<sub>T</sub> Port Bit Clear Register

- 1 = The corresponding bit of EPDR will be cleared;
- 0 = The corresponding bit of EPDR will not be effected;

## 24. CANBus Controller(CANBC)

### 24.1. Introduction

LT168’s CANBus module is a communication controller implementing the CAN protocol based on the CAN 2.0B protocol specification. A general block diagram is shown in **Figure 24-1**, 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. The functions of the submodules are described in subsequent sections.



**Figure 24-1: CANBus Block Diagram**

#### 24.1.1. 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.

Up to 64 Message Buffers are available. 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 CANBus, 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.

### 24.1.2. CANBus Module Features

The CANBus module includes below 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
- 64 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 1056 bytes (64 MBs) of SRAM used for MB storage
- Includes 256 bytes (64 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 (8-bits) IDs, with individual masking capability
- Programmable clock source to the CANBUS 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 16-bits 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

### 24.1.3. 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 CANBUS 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 CANBus is lost.

#### • 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 CANBUS 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.

**24.2. External Signal Description**

**24.2.1. Overview**

The CANBus module has two I/O signals connected to the external MCU pins. These signals are summarized in **Table 24-1** and described in more detail in the next subsections.

**Table 24-1: CANBus Signals**

Signal name	Direction	Description
CANRX	Input	CANBUS receive pin
CANTX	Output	CANBUS transmit pin

**24.2.2. Signal Descriptions**

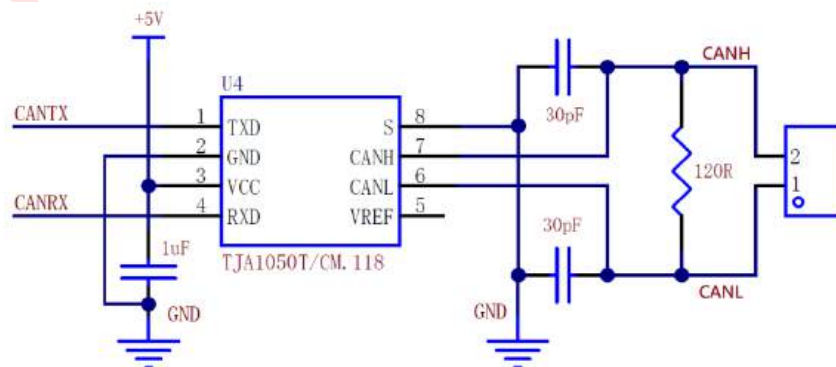
**24.2.2.1. CANRX**

This pin is the receive pin from the CANBus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.

**24.2.2.2. CANTX**

This pin is the transmit pin to the CANBus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.

The following is the example of Can bus application circuit.



**Figure 24-2: Canbus Circuit Example**

## 25. Serial Communication Interface (SCI)

### 25.1. Introduction

LT168's 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.

### 25.2. Features

Features of each SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13-bits 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 8-bits, 9-bits or 10-bits character length
- Programmable 1-bit or 2-bits 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 13-bits break character generation / 11-bits 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

### 25.3. Modes of Operation

- Stop mode
- Wait mode

### 25.4. Block Diagram

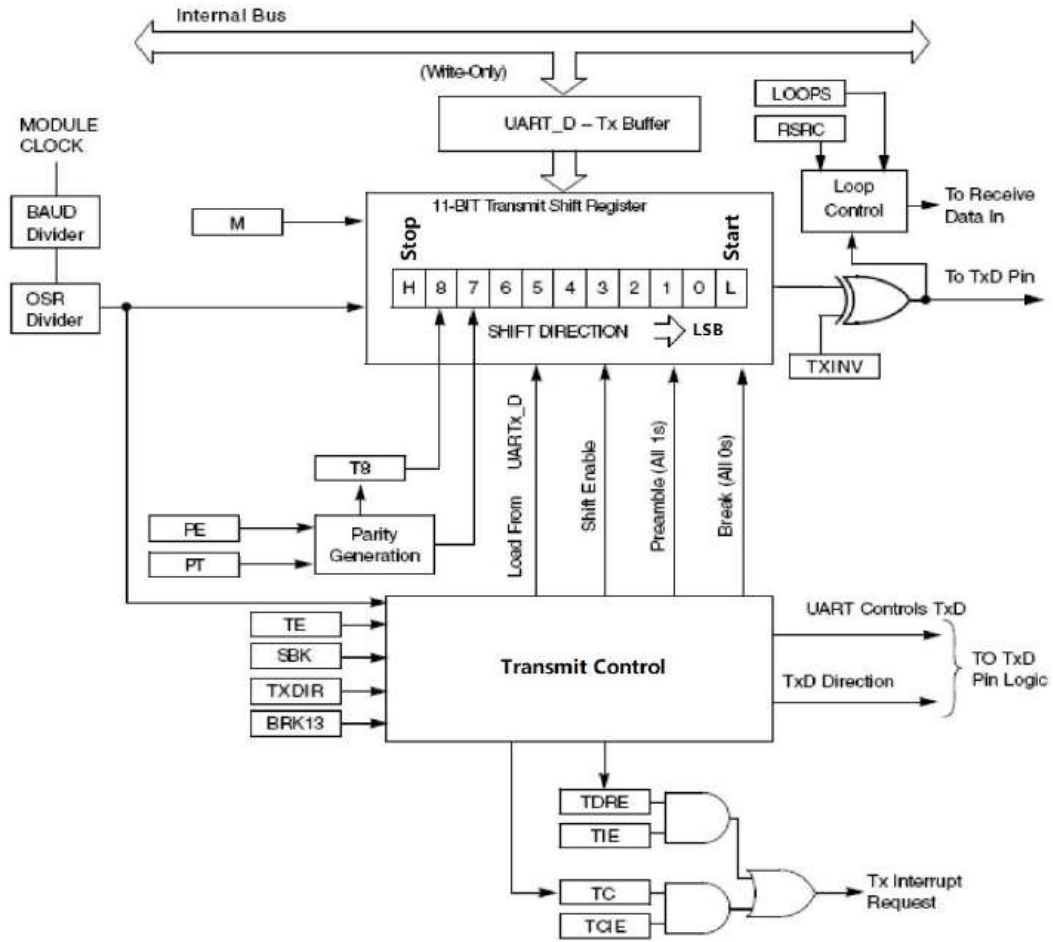


Figure 25-1: SCI Transmitter Block Diagram

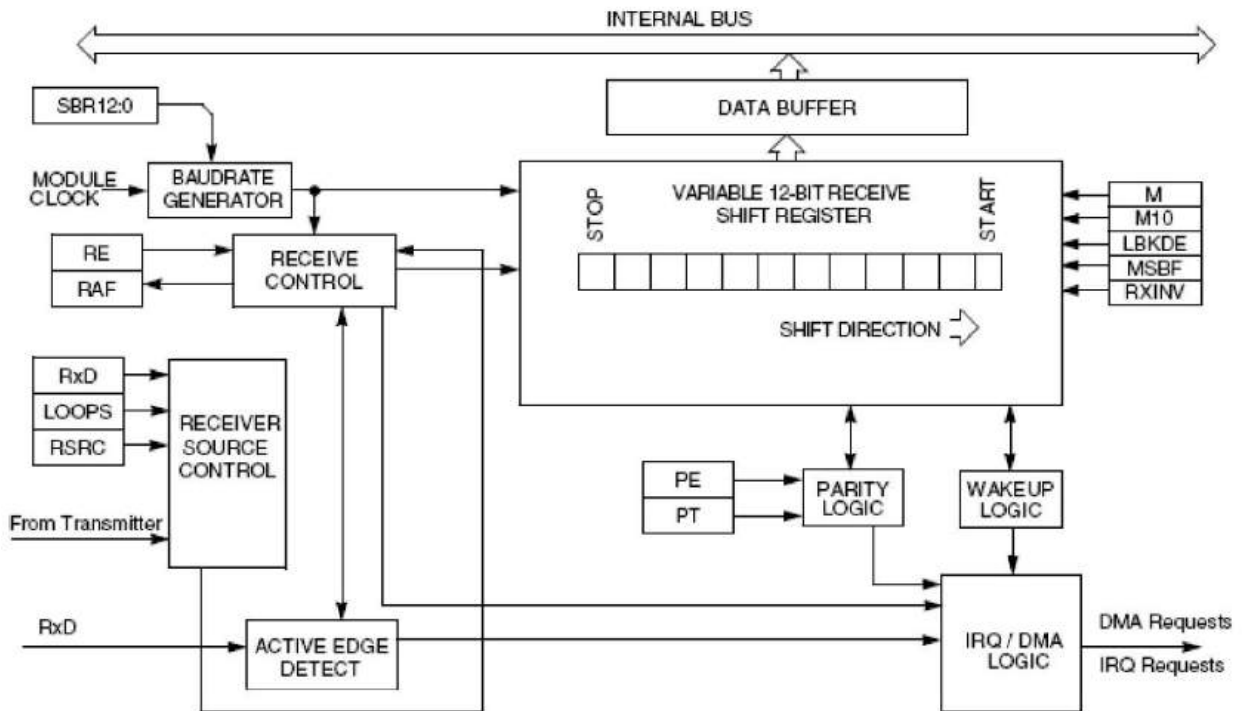


Figure 25-2: SCI Receiver Block Diagram

## 25.5. Modes of Operation

### 25.5.1. Stop Mode

The SCI will not be functional during Stop mode.

### 25.5.2. 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.

## 25.6. Signal Description

Table 25-1 gives an overview of the signals which are described here.

Table 25-1: 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

## 25.7. Memory Map and Registers

### 25.7.1. Memory Map

The SCI Module memory map is shown in **Table 25-2**. The SCI0 base address is 0x4009\_0000, SCI1 is 0x4008\_0000, and SCI2 is 0x400C\_0000.

**Table 25-2: SCI Module Memory Map**

Offset Address	Bits[31:0]
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:**

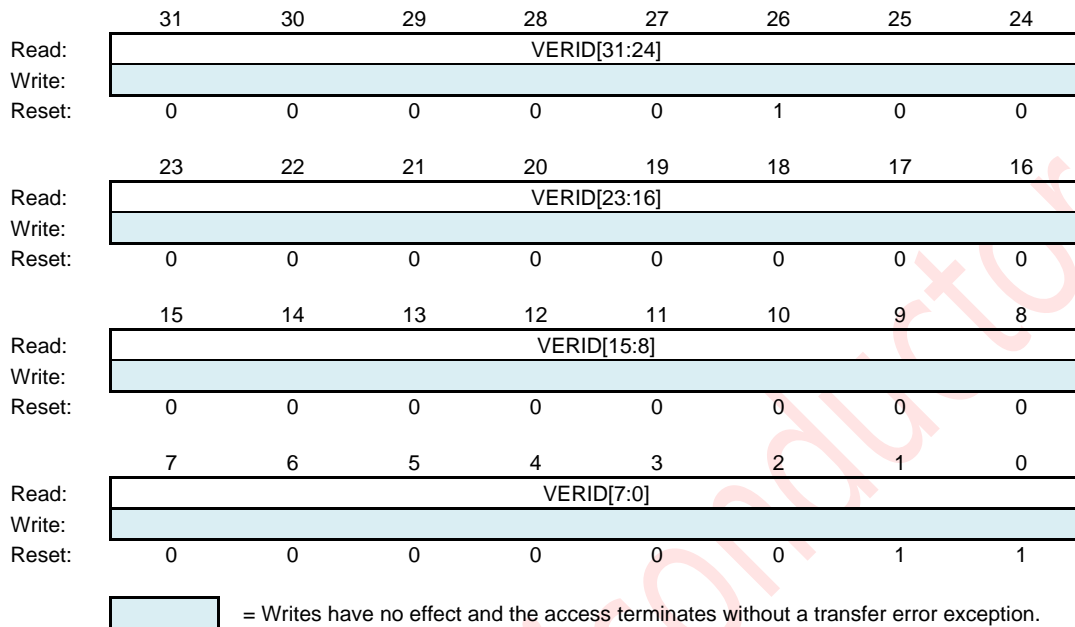
Each module is assigned 16K Bytes of address space, all of which may not be decoded. Accesses outside of the specified module memory map generate a bus error exception.



**25.7.2. Register Description**

**25.7.2.1. SCI Version ID Register (SCI\_VERID)**

**Address: SCIn\_BASEADDR+0x0000\_0000**



**Figure 25-3: SCI Version ID Register (SCI\_VERID)**

**VERID\_ID[31:0]** — SCI Version ID

**25.7.2.2. SCI Parameter Register (SCI\_PARAM)**

**Address: SCIn\_BASEADDR+0x0000\_0004**



**Figure 25-4: SCI Parameter Register (SCI\_PARAM)**

**RXFIFO\_SZ[3:0]** — The receive buffer/FIFO size

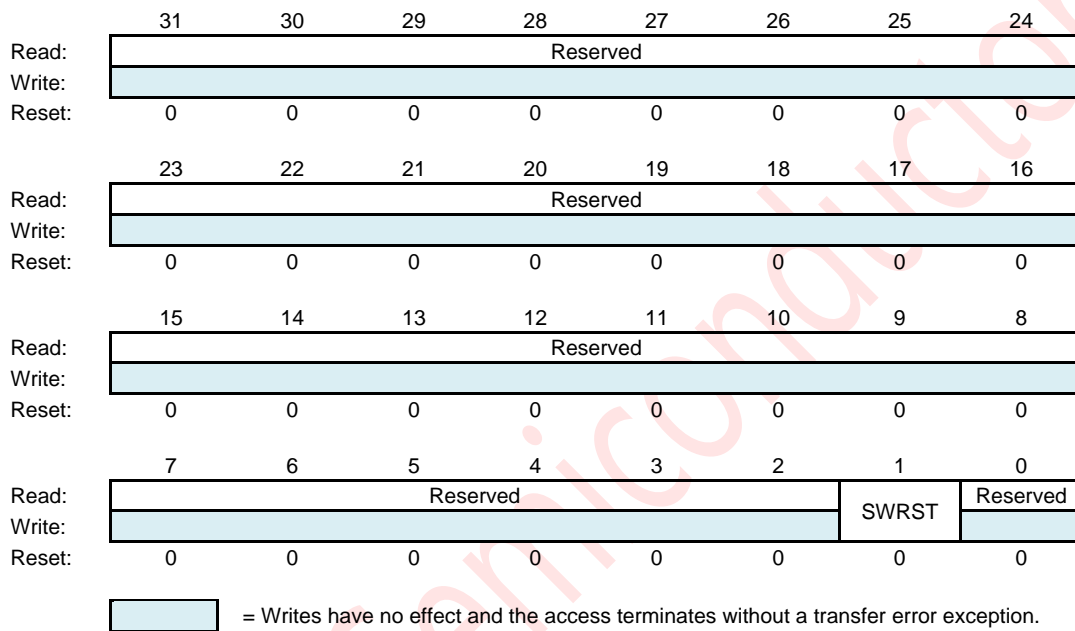
This read-only control bit indicates the the receive buffer/FIFO size.

**TXFIFO\_SZ[3:0]** — The transmit buffer/FIFO size

This read-only control bit indicates the the transmit buffer/FIFO size.

**25.7.2.3. SCI Reset Register (SCI\_RESET)**

**Address: SCIn\_BASEADDR+0x0000\_0008**



**Figure 25-5: SCI Reset Register (SCI\_RESET)**

**SWRST** — Software reset

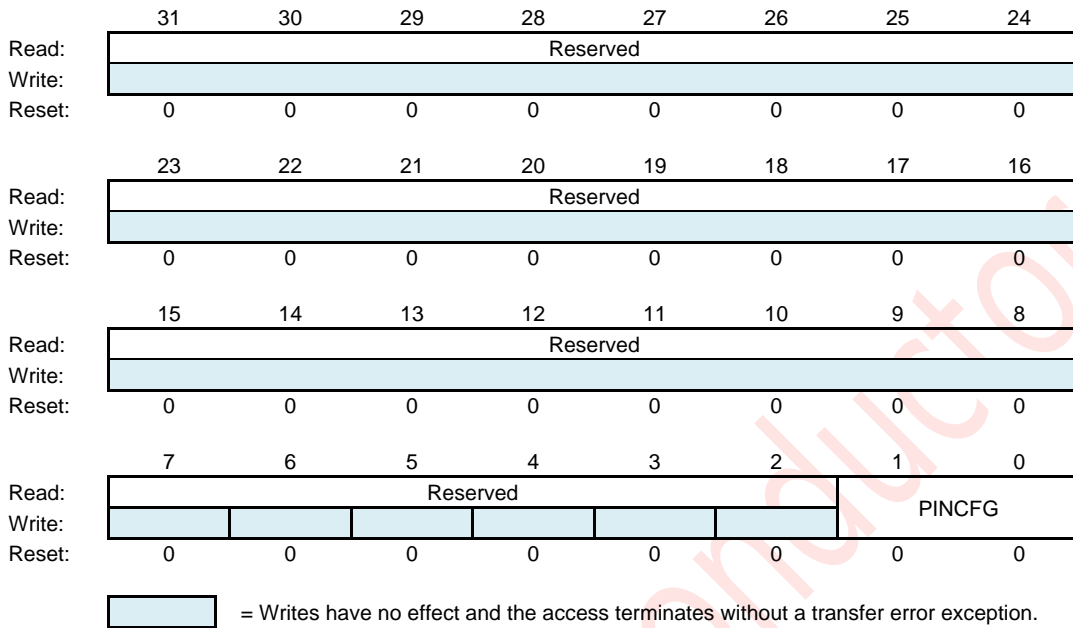
This read/write bit allows the software to reset SCI IP.

1 = Software reset is asserted

0 = No software reset is asserted

**25.7.2.4. SCI Pin Register (SCI\_PIN)**

**Address: SCIn\_BASEADDR+0x0000\_000C**

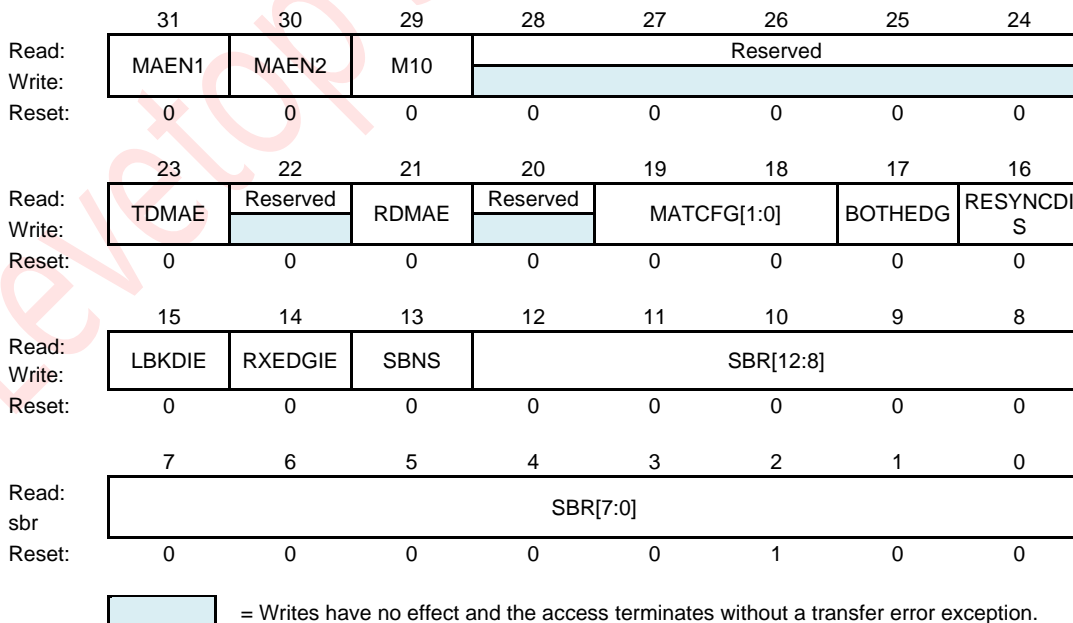


**Figure 25-6: SCI Pin Register (SCI\_PIN)**

**PINCFG** — Useless in this module

**25.7.2.5. SCI Baud Rate Register (SCI\_BAUD)**

**Address: SCIn\_BASEADDR+0x0000\_0010**



**Figure 25-7: SCI Baud Rate Register (SCI\_BAUD)**

**MAEN1** — Match Address Mode Enable 1

- 1 = Enable automatic address matching or data matching mode for MATCH[MA1]
- 0 = Normal operation

**MAEN2** — Match Address Mode Enable 1

- 1 = Enable automatic address matching or data matching mode for MATCH[MA2]
- 0 = Normal operation

**M10** — 10-bits 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 10-bits data characters
- 0 = Receiver and transmitter use 8-bits or 9-bits 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 is enabled
- 0 = DMA request is 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 is enabled
- 0 = DMA request is disabled

**MATCFG[1:0]** — Match Configuration

Configure the match addressing mode used:

- 00 -- Address Match Wakeup;
- 01 -- Idle Match Wakeup;
- 10 -- Match On and Match Off;
- 11 -- Enable RWU on Data Match and Match On/Off for transmitter CTS input

**BOTHEDGE** — Both Edge Sampling

Enable 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, it 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 interrupts are requested when SCI\_STAT[LBKDIF] flag is 1
- 0 = Hardware interrupts from SCI\_STAT[LBKDIF] are disabled (use polling)

**RXEDGIE** — RX Input Active Edge Interrupt Enable

Enable 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] is 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[12:0]** — 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 13-bits 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).

**25.7.2.6. SCI Status Register (SCI\_STAT)**

Address: SCIn\_BASEADDR+0x0000\_0014

	31	30	29	28	27	26	25	24
Read:	LBKDIF	RXEDGIF	MSBF	RXINV	RWUID	BRK13	LBKDE	RAF
Write:	w1c	w1c						
Reset:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
Read:	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write:				w1c	w1c	w1c	w1c	w1c
Reset:	1	1	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
Read:	MA1F	MA2F	Reserved					
Write:	w1c	w1c						
Reset:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
Read:	Reserved							
sbr								
Reset:	0	0	0	0	0	1	0	0

= Writes have no effect and the access terminates without a transfer error exception.  
 w1c = .Write 1 to the bit will clear it

**Figure 25-8: 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 are inverted
- 0 = Receive data are 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-bits 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-bits 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-bits 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 datawords 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 is empty

0 = Transmit data buffer is 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 is idle (transmission activity complete)

0 = Transmitter is 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 datawords in the receive buffer is greater than the number indicated by SCI\_WATER[RXWATER]. To clear RDRF, read SCI\_DATA until the number of datawords 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 is full

0 = Receive data buffer is 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-bits times, needed for the receiver to detect an idle line. When ILT is set, the receiver does not 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 LBDIF 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 data word 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

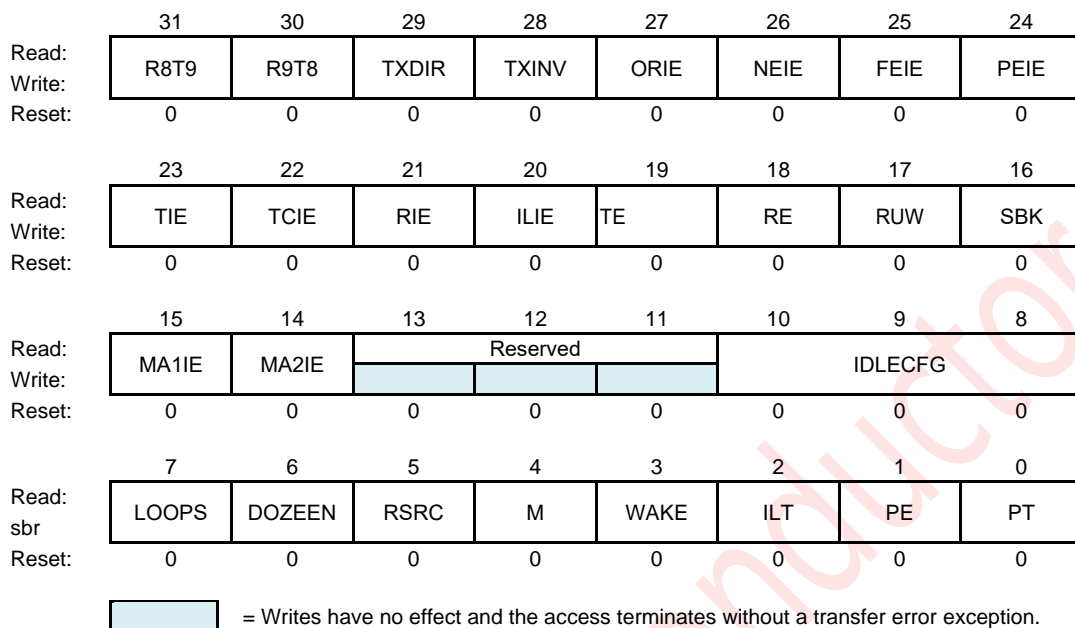
MA2F is set whenever the next character to be read from SCI\_DATA matches MA2. To clear MA2F, write a logic one to the MA2F.

- 1 = Received data is equal to MA2
- 0 = Received data is not equal to MA2



**25.7.2.7. SCI Control Register (SCI\_CTRL)**

**Address: SCIn\_BASEADDR+0x0000\_0018**



**Figure 25-9: 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 9-bits or 10-bits data formats. When reading 9-bits or 10-bits data, read R8 before reading SCI\_DATA.

T9 is the tenth data bit received when the SCI is configured for 10-bits data formats. When writing 10-bits 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, then 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 10-bits data formats. When reading 10-bits data, read R9 before reading SCI\_DATA

T8 is the ninth data bit received when the SCI is configured for 9-bits or 10-bits data formats. When writing 9-bits or 10-bits 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, then 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 is inverted
- 0 = Transmit data is not inverted

**ORIE** — Overrun Interrupt Enable

This bit enables the overrun flag (OR) to generate hardware interrupt requests.

- 1 = Hardware interrupt is requested when OR is set
- 0 = OR interrupts are disabled; use polling

**NEIE** — Noise Error Interrupt Enable

This bit enables the noise flag (NF) to generate hardware interrupt requests.

- 1 = Hardware interrupt is requested when NF is set
- 0 = NF interrupts are disabled; use polling

**FEIE** — Framing Error Interrupt Enable

This bit enables the framing error flag (FE) to generate hardware interrupt requests.

- 1 = Hardware interrupt is requested when FE is set
- 0 = FE interrupts are disabled; use polling

**PEIE** — Parity Error Interrupt Enable

This bit enables the parity error flag (PF) to generate hardware interrupt requests.

- 1 = Hardware interrupt is requested when PF is set
- 0 = PF interrupts are disabled; use polling

**TIE** — Transmit Interrupt Enable

This bit enables STAT[TDRE] to generate interrupt requests.

- 1 = Hardware interrupt is requested when TDRE flag is 1
- 0 = Hardware interrupts from TDRE are disabled; use polling

**TCIE** — Transmission Complete Interrupt Enable for

TCIE enables the transmission complete flag, TC, to generate interrupt requests.

- 1 = Hardware interrupt is requested when TC flag is 1
- 0 = Hardware interrupts from TC are disabled; use polling

**RIE** — Receiver Interrupt Enable

This bit enables STAT[RDRF] to generate interrupt requests.

- 1 = Hardware interrupt is requested when RDRF flag is 1
- 0 = Hardware interrupts from RDRF are disabled; use polling

**ILIE** — Idle Line Interrupt Enable

ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests.

- 1 = Hardware interrupt is requested when IDLE flag is 1
- 0 = Hardware interrupts from IDLE are disabled; use polling

**TE** — Transmitter Enable

This bit 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 is enabled
- 0 = Transmitter is disabled

**RE** — Receiver Enable

This bit 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 is enabled
- 0 = Receiver is 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 is enabled
- 0 = MA1F interrupt is disabled

**MA2IE** — Match 2 Interrupt Enable

- 1 = MA2F interrupt is enabled
- 0 = MA2F interrupt is disabled

**IDLECFG** — Idle Configuration

This bit 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** — 9-Bit or 8-Bit Mode Select

1 = Receiver and transmitter use 9-bits data characters

0 = Receiver and transmitter use 8-bits data characters **WAKE** — Receiver Wakeup Method Select

**WAKE** — Receiver Wakeup Method Select

This bit 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

This bit 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

This bit enables hardware parity generation and checking. When parity is enabled, the bit before the stop bit is immediately treated as the parity bit.

1 = Parity is enabled

0 = No hardware parity generation or checking

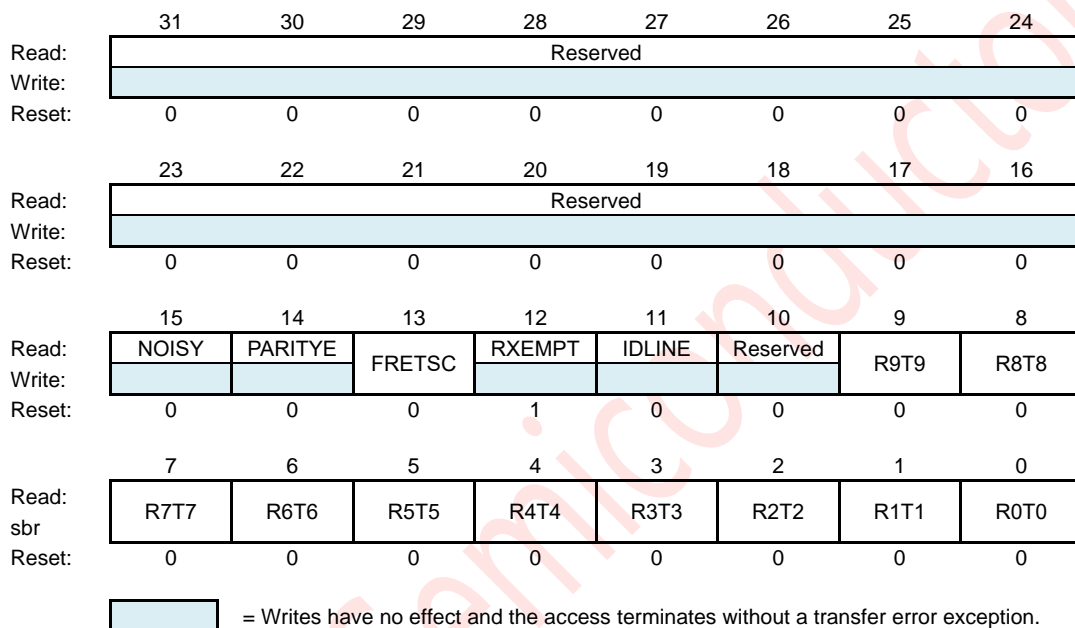
**PT** — Parity Type

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

**25.7.2.8. SCI Data Register (SCI\_DATA)**

**Address: SCIn\_BASEADDR+0x0000\_001C**



**Figure 25-10: 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, this bit indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, this bit 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, the 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

This bit asserts when there is no data in the receive buffer. This field does not take into account the 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

**IDLIN** — Idle Line

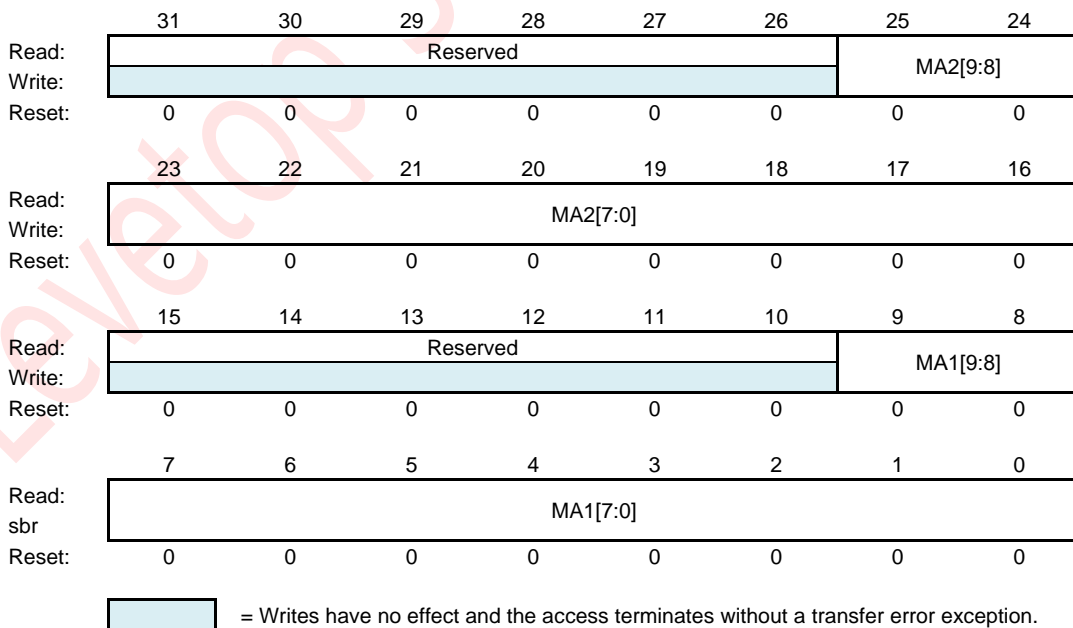
This bit indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled.

- 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

**25.7.2.9. SCI Match Address Register (SCI\_MATCH)**

**Address: SCIn\_BASEADDR+0x0000\_0020**



**Figure 25-11: SCI Match Address Register (SCI\_MATCH)**

**MA2[9:0]** — Match Address 2

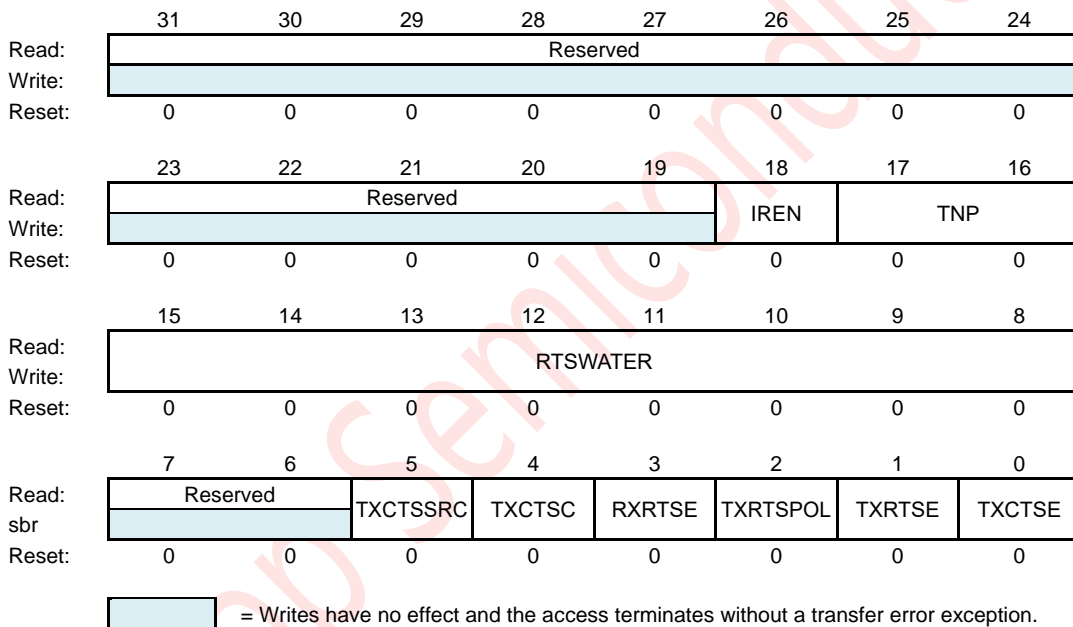
The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is cleared.

**MA1[9:0]** — 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 cleared.

**25.7.2.10. SCI Modem IrDA Register (SCI\_MODIR)**

**Address: SCIn\_BASEADDR+0x0000\_0024**



**Figure 25-12: SCI Modem IrDA Register (SCI\_MODIR)**

**IREN** — Infrared enable

This bit enables/disables the infrared modulation/demodulation.

- 1 = IR is enabled
- 0 = IR is disabled

**TNP** — Transmitter narrow pulse

This bit 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

This bit 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

This bit 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

This bit 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

This bit 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

This bit 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

This bit 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

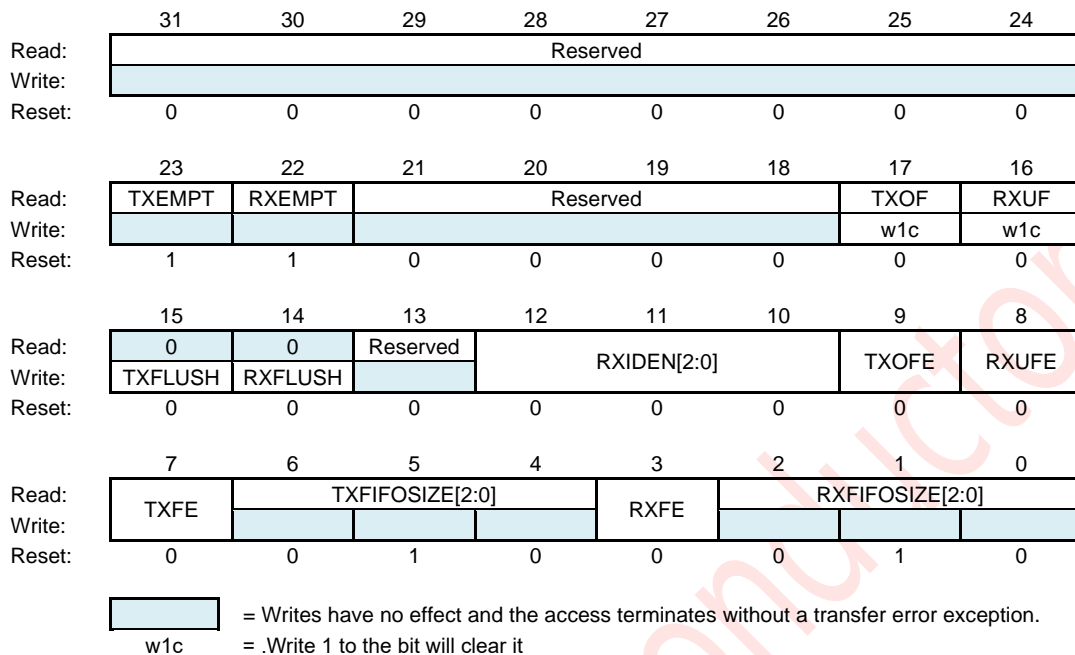
TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE.

- 1 = Clear-to-send operation is enabled. 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



**25.7.2.11. SCI FIFO Register (SCI\_FIFO)**

**Address: SCIn\_BASEADDR+0x0000\_0028**



**Figure 25-13: SCI FIFO Register (SCI\_FIFO)**

**TXEMPT** — Transmit Buffer/FIFO Empty

This bit asserts when there is no data in the Transmit FIFO/buffer. This field does not take into account the data that is in the transmit shift register.

- 1 = Transmit buffer is empty
- 0 = Transmit buffer is not empty

**RXEMPT** — Receive Buffer/FIFO Empty

This bit asserts when there is no data in the receive FIFO/Buffer. This field does not take into account the data that is in the receive shift register.

- 1 = Receive buffer is empty
- 0 = Receive buffer is not empty

**TXOF** — Transmitter Buffer Overflow Flag

This bit 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

This bit 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 the 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 the 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, it will enable 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 = RXUF 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 disabled. Buffer is depth 1. (Legacy support)

**TXFIFOSIZE[2:0]** — Transmit FIFO/Buffer Depth

This field indicates 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 the depth indicated by RXFIFOSIZE
- 0 = Receive FIFO is disabled. Buffer is depth 1. (Legacy support)

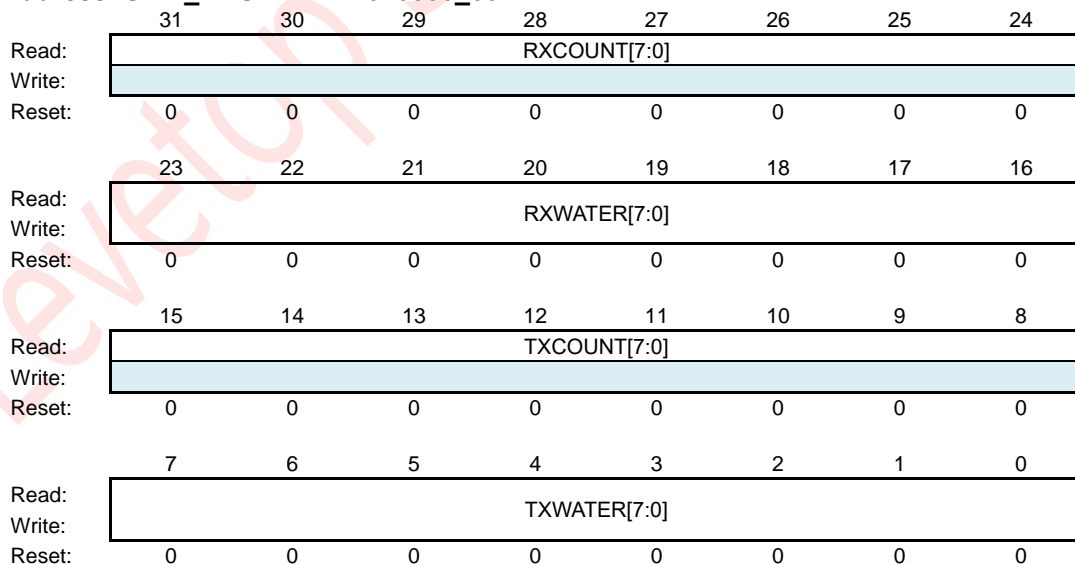
**RXFIFOSIZE[2:0]** — Receive FIFO/Buffer Depth

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;

**25.7.2.12. SCI Watermark Register (SCI\_WATER)**

**Address: SCIn\_BASEADDR+0x0000\_002C**



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

**Figure 25-14: SCI Watermark Register (SCI\_WATER)**

**RXCOUNT[7:0]** — 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[7:0]** — 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[7:0]** — 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[7:0]** — 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].

**25.7.2.13. SCI Oversampling Ratio Register (SCI\_OSR)**

Address: SCIn\_BASEADDR+0x0000\_0030

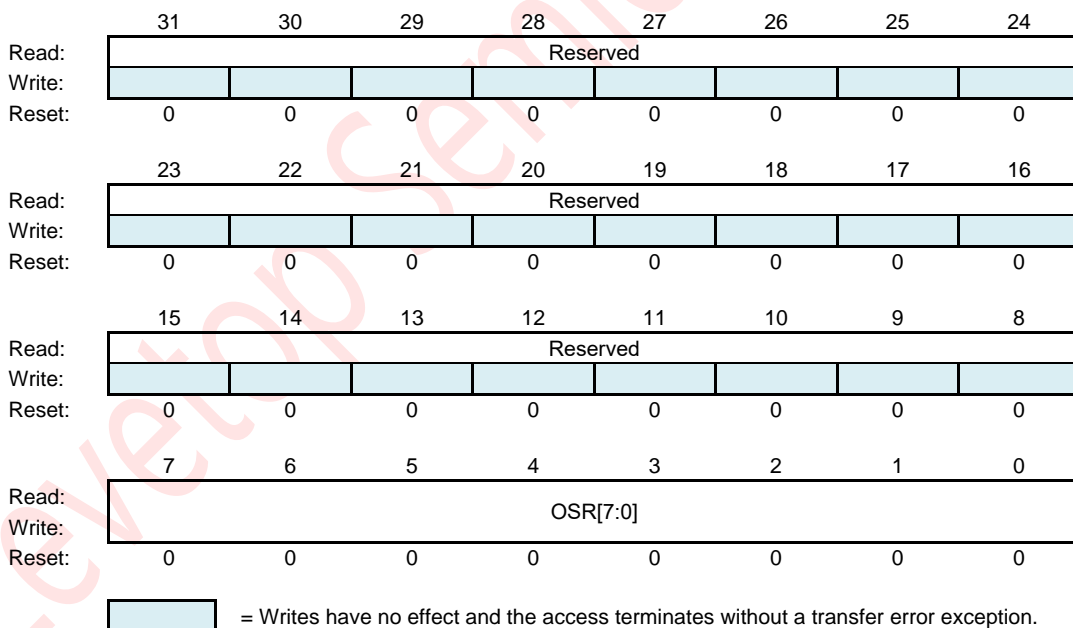


Figure 25-15: SCI Oversampling Ratio Register (SCI\_OSR)

**OSR[7:0]** — 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. The oversampling ratio = OSR + 1.

### 25.8. Functional Description

The SCI supports full-duplex, asynchronous, and 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.

### 25.9. Baud Rate Generation

A 13-bits 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

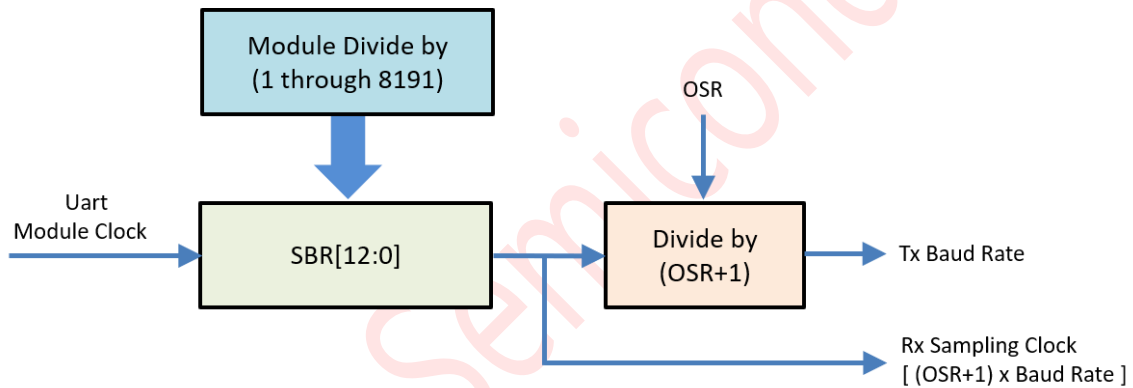


Figure 25-16: SCI Baud Rate Generation

$$\text{Baud Rate} = \frac{\text{Uart Module Clock}}{\text{SBR[12:0]} \times (\text{OSR}+1)}$$

**Note:** Baud Rate Generator off if SBR[12:0] = 0.

## 25.10. 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 10-bits 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 8-bits data mode. In 8-bits 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.

### 25.10.1. 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, 10-bits to 12-bits times including the start and stop bits. A longer break of 13-bits 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 cleared. 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 is not finished 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 25-3: Break Character Length**

BRK13	M	M10	SBNS	Break Character Length
0	0	0	0	10-bits times
0	0	0	1	11-bits times
0	1	0	0	11-bits times
0	1	0	1	12-bits times
0	X	1	0	12-bits times
0	X	1	1	13-bits times
1	0	0	0	13-bits times
1	0	0	1	13-bits times
1	1	0	0	14-bits times
1	1	0	1	14-bits times
1	X	1	0	15-bits times
1	X	1	1	15-bits times

## 25.11. 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 9-bits or 10-bits data mode, refer to 8-bits, 9-bits and 10-bits data modes. For the remainder of this discussion, assume the SCI is configured for normal 8-bits 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]` has already been set, it indicates the receive data register (buffer) has already been 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.

### 25.11.1. Data Sampling Technique

The SCI receiver supports a configurable oversampling rate of between  $4\times$  and  $256\times$  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  $OSR\times 2$ ). 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  $4\times$  to  $7\times$  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, if 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.



## 25.11.2. 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 25-4: 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	X0	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

### 25.11.2.1. 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 8-bits to 10-bits data mode and the SCI\_BAUD[SBNS] bit selects 1-bit or 2-bits stop bit number that determines how many bit times of idle are needed to constitute a full character time, 10 to 13-bits 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.

### 25.11.2.2. 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.

### 25.11.2.3. 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.

### 25.11.2.4. 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.

### 25.11.2.5. 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.

### 25.11.2.6. 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.

## 25.11.3. 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.

### 25.11.3.1. 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.

### 25.11.3.2. 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.

### 25.11.3.3. Lowbits 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.

#### 25.11.3.4. Highbits 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 lowbits is detected according to Lowbits 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.

### 25.12. Additional SCI Functions

The following sections describe additional SCI functions.

#### 25.12.1. 8-bits, 9-bits and 10-bits Data Modes

The SCI transmitter and receiver can be configured to operate in 9-bits data mode by setting the SCI\_CTRL[M] or 10-bits data mode by setting SCI\_CTRL[M10]. In 9-bits mode, there is a ninth data bit, and in 10-bits 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 16-bits or 32-bits accesses to the SCI\_DATA register.

For coherent 8-bits writes to the transmit data buffer, write to SCI\_CTRL[T8] and SCI\_CTRL[T9] before writing to SCI\_DATA[7:0]. For 16-bits and 32-bits 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 those of 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 9-bits 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 10-bits 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.

#### 25.12.2. 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.

### 25.12.3. 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.

### 25.12.4. 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.

## 25.13. 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.

### 25.13.1. 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.

### 25.13.2. 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.

## 25.14. Interrupts And Status Flags

The SCI transmitter has two status flags that can optionally generate hardware interrupt requests. The transmit data register empty SCI\_STAT[TDRE] bit 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. The transmit complete (SCI\_STAT[TC]) bit 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).

## 26. Synchronous Serial Interface (SSI)

### 26.1. Introduction

SSI is a programmable Synchronous Serial Interface (SSI) peripheral. This is an 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.

LT168's SSI module provides QSPI0, QSPI1 and QSPI2 interface to SPI Flash or other SPI devices. The QSPI0 signals are connected directly to the embedded SPI Flash memory, and the QSPI1 and QSPI2 signals are available for users.

### 26.2. Features

Features include:

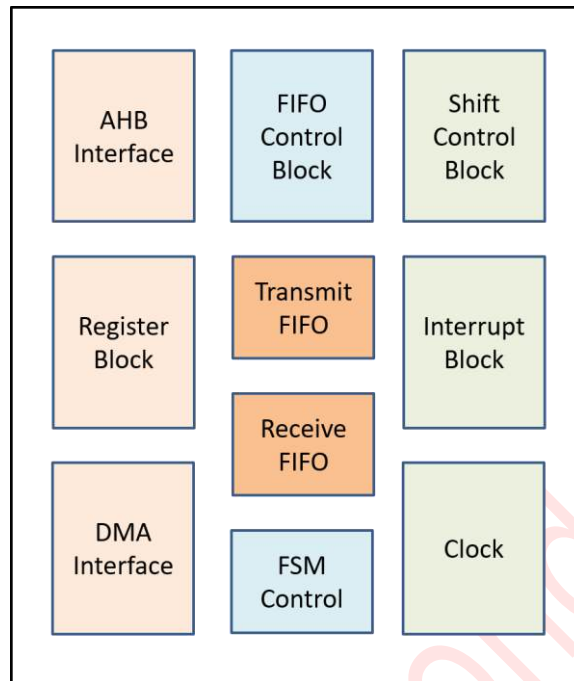
- Serial-master Operation.
- DMA Controller Interface – This enables the SSI to interface to a DMA controller over the bus using handshaking interface for transfer requests.
- Clock stretching support in enhanced SPI transfers.
- Data item size (4 to 32-bits) – Item size of each data transfer under control of the programmer.
- FIFO Depth – The transmit and receive FIFO buffers are 8 words deep. The FIFO width is fixed at 32-bits.
- Enhanced SPI Support.
- Execute in Place (XIP) Mode Support.

### 26.3. 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.

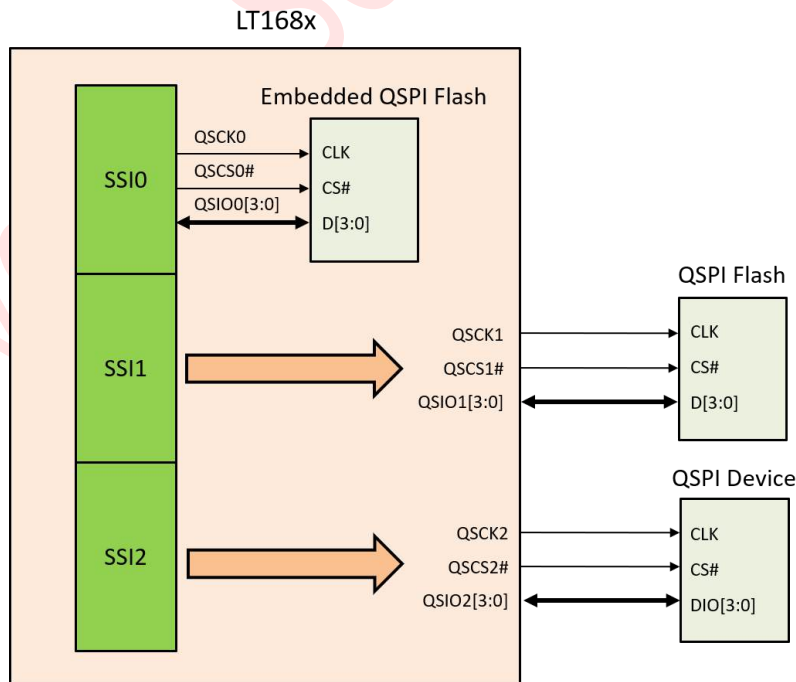
**26.4. Block Diagram**



**Figure 26-1: SSI Block Diagram**

**26.5. Application Diagram**

The following is the application of LT168's SSI module.



**Figure 26-2: SSI Application Diagram**



## 26.6. Memory Map and Registers

### 26.6.1. Memory Map

The SSI Module memory map is shown in **Table 26-1**. The QSPI0 base address is 0x6000\_0000, and QSPI1 is 0x7000\_0000, and QSPI2 is 0x8000\_0000.

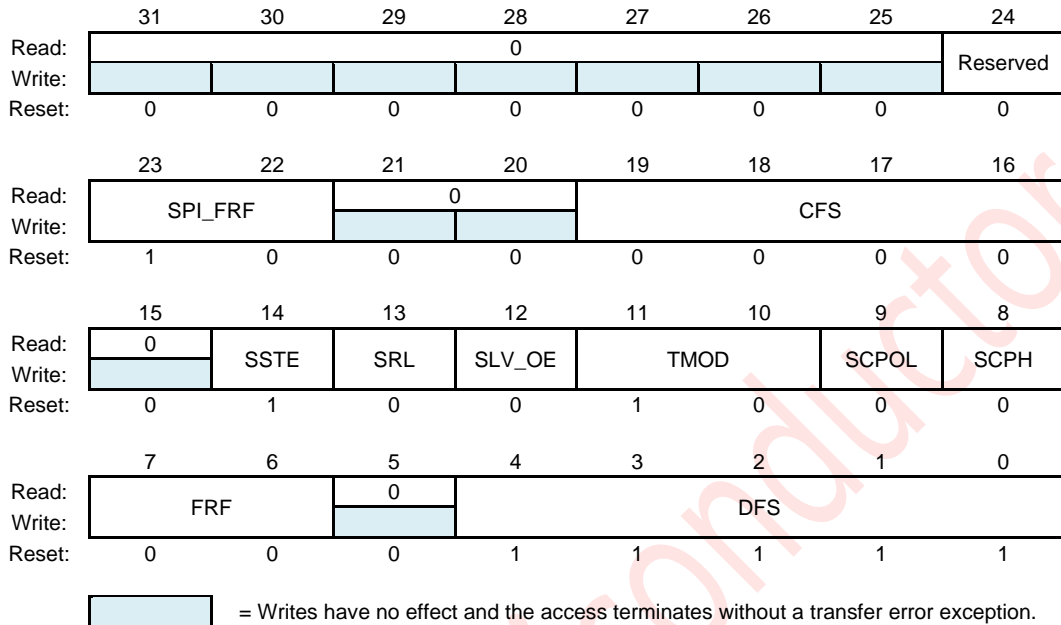
**Table 26-1: SSI Memory Map**

Offset Address	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
0x000C	Microwire Control Register (MWCR)	S/U
0x0010	Slave Select Register (SER)	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 (XIPHIR)	S/U
0x0104	XIP Wrap Inst Register (XIPWIR)	S/U
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 Continus Transfer Time Out Register (XIPCTTOR)	S/U

**26.6.2. Register Descriptions**

**26.6.2.1. Control Register 0 (CTRLR0)**

**Address: QSPIn\_BASEADDR+0x0000\_0000**



**Figure 26-3: 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): 1-bit Control Word
- 0x1 (SIZE\_02\_BIT): 2-bits Control Word
- 0x2 (SIZE\_03\_BIT): 3-bits Control Word
- 0x3 (SIZE\_04\_BIT): 4-bits Control Word
- 0x4 (SIZE\_05\_BIT): 5-bits Control Word
- 0x5 (SIZE\_06\_BIT): 6-bits Control Word
- 0x6 (SIZE\_07\_BIT): 7-bits Control Word
- 0x7 (SIZE\_08\_BIT): 8-bits Control Word
- 0x8 (SIZE\_09\_BIT): 9-bits Control Word
- 0x9 (SIZE\_10\_BIT): 10-bits Control Word

0xA (SIZE\_11\_BIT): 11-bits Control Word  
 0xB (SIZE\_12\_BIT): 12-bits Control Word  
 0xC (SIZE\_13\_BIT): 13-bits Control Word  
 0xD (SIZE\_14\_BIT): 14-bits Control Word  
 0xE (SIZE\_15\_BIT): 15-bits Control Word  
 0xF (SIZE\_16\_BIT): 16-bits 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 = Slave Output is disabled  
 0 = 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

### SCPOL — 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\_HIGH): Inactive state of serial clock is high  
 0 = (INACTIVE\_LOW): 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 32-bits, 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): 4-bits serial data transfer

0x4 (DFS\_05\_BIT): 5-bits serial data transfer

0x5 (DFS\_06\_BIT): 6-bits serial data transfer

0x6 (DFS\_07\_BIT): 7-bits serial data transfer

0x7 (DFS\_08\_BIT): 8-bits serial data transfer

0x8 (DFS\_09\_BIT): 9-bits serial data transfer

0x9 (DFS\_10\_BIT): 10-bits serial data transfer

0xA (DFS\_11\_BIT): 11-bits serial data transfer

0xB (DFS\_12\_BIT): 12-bits serial data transfer

0xC (DFS\_13\_BIT): 13-bits serial data transfer

0xD (DFS\_14\_BIT): 14-bits serial data transfer

0xE (DFS\_15\_BIT): 15-bits serial data transfer

0xF (DFS\_16\_BIT): 16-bits serial data transfer

0x10 (DFS\_17\_BIT): 17-bits serial data transfer

0x11 (DFS\_18\_BIT): 18-bits serial data transfer

0x12 (DFS\_19\_BIT): 19-bits serial data transfer

0x13 (DFS\_20\_BIT): 20-bits serial data transfer

0x14 (DFS\_21\_BIT): 21-bits serial data transfer

0x15 (DFS\_22\_BIT): 22-bits serial data transfer

0x16 (DFS\_23\_BIT): 23-bits serial data transfer

0x17 (DFS\_24\_BIT): 24-bits serial data transfer

0x18 (DFS\_25\_BIT): 25-bits serial data transfer

0x19 (DFS\_26\_BIT): 26-bits serial data transfer

0x1A (DFS\_27\_BIT): 27-bits serial data transfer

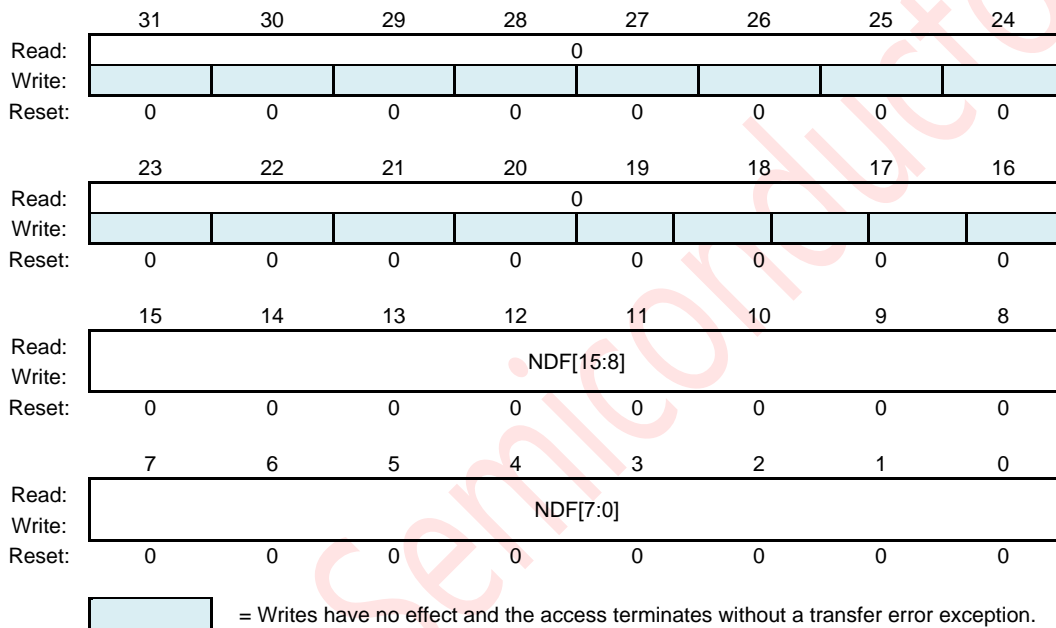
0x1B (DFS\_28\_BIT): 28-bits serial data transfer

- 0x1C (DFS\_29\_BIT): 29-bits serial data transfer
- 0x1D (DFS\_30\_BIT): 30-bits serial data transfer
- 0x1E (DFS\_31\_BIT): 31-bits serial data transfer
- 0x1F (DFS\_32\_BIT): 32-bits serial data transfer

**26.6.2.2. 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.

**Address: QSPIn\_BASEADDR+0x0000\_0004**



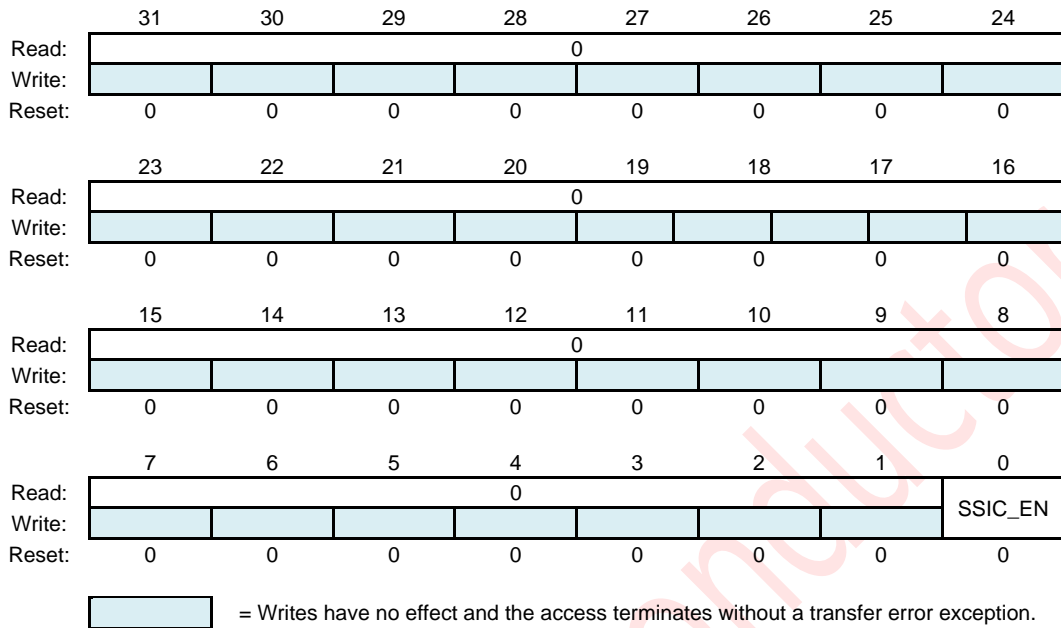
**Figure 26-4: Control Register 1 (CTRLR1)**

**NDF[15:0]** — 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.

**26.6.2.3. SSI Enable Register (SSIENR)**

**Address: QSPIn\_BASEADDR+0x0000\_0008**



**Figure 26-5: 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 = SSI is enabled.
- 0 = SSI is disabled.

26.6.2.4. Microwire Control Register (MWCR)

Address: QSPIn\_BASEADDR+0x0000\_000C

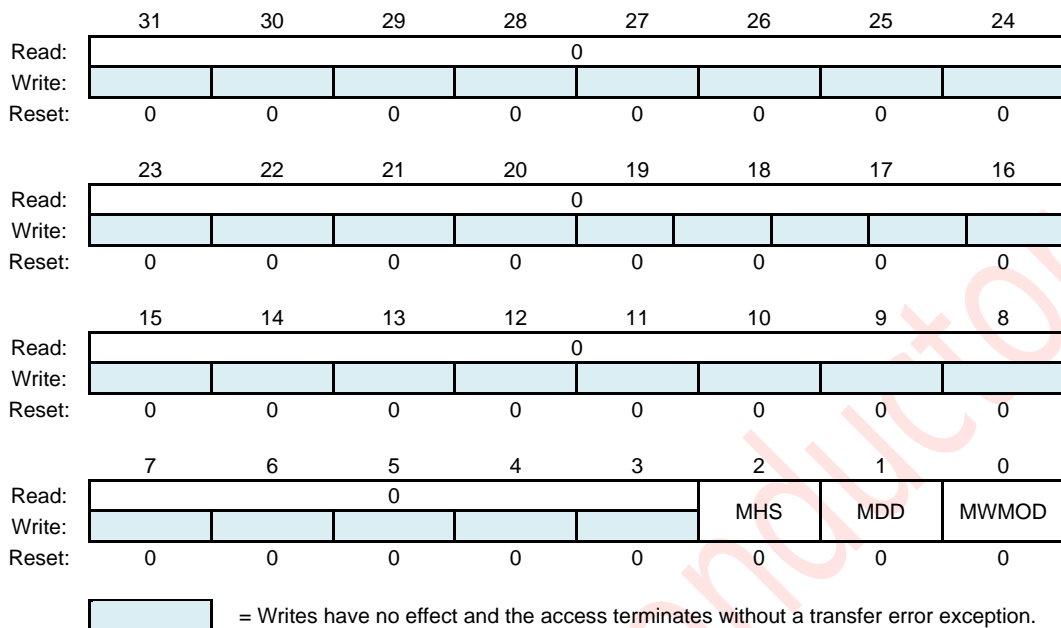


Figure 26-6: Microwire Control Register (MWCR)

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 = handshaking interface is enabled
- 0 = 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 = SSI transmits data
- 0 = SSI receives data

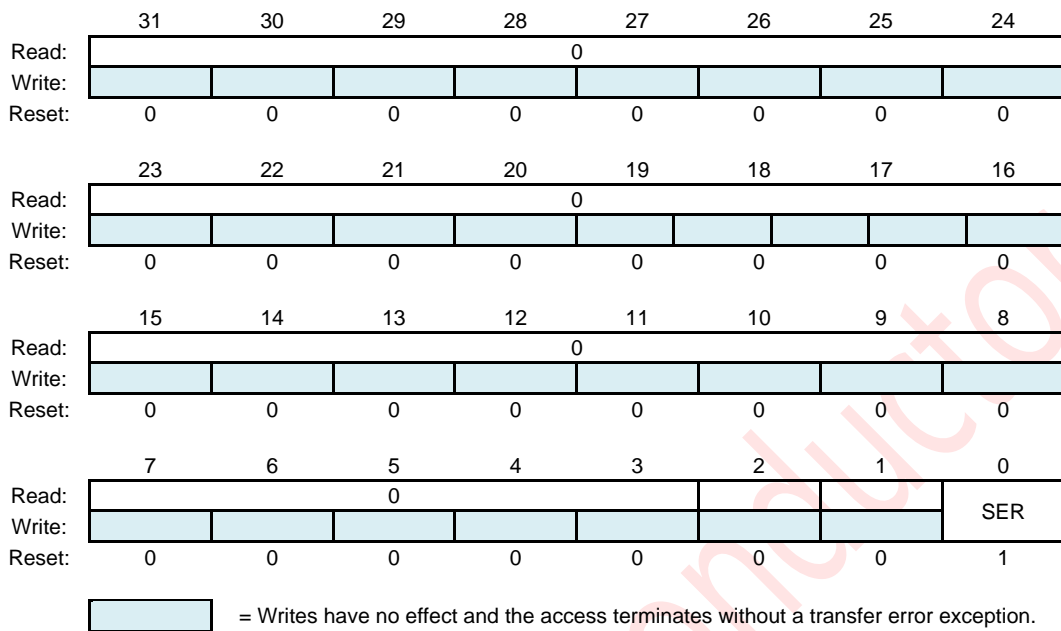
**MWMOD** — Microwire Transfer Mode.

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 Transfer
- 0 = Non-Sequential Transfer

**26.6.2.5. Slave Enable Register (SER)**

**Address: QSPIn\_BASEADDR+0x0000\_0010**



**Figure 26-7: 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.

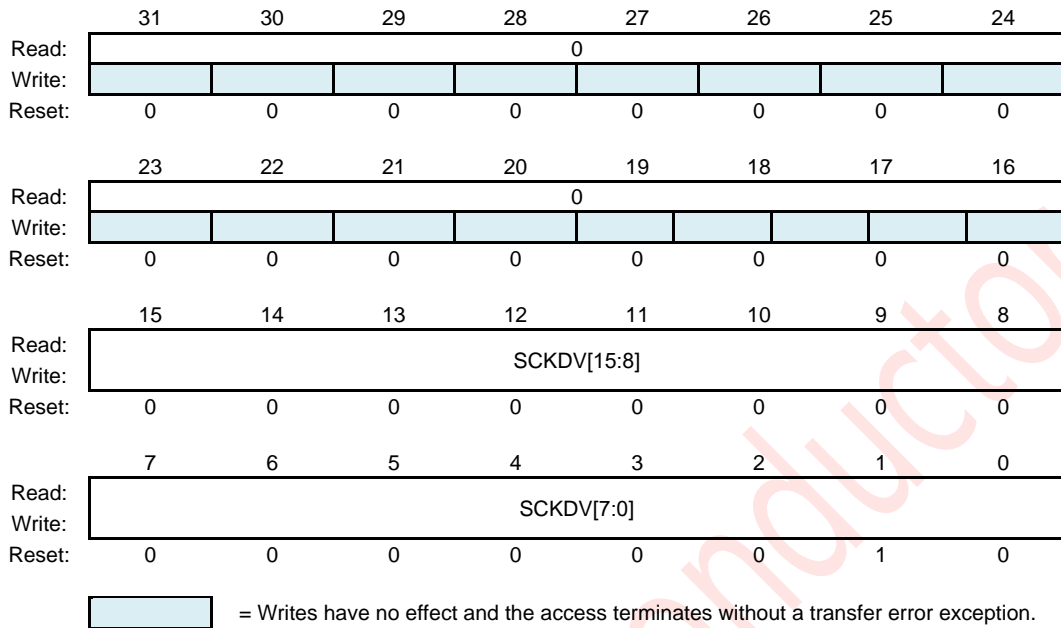
1 = Selected

0 = Not Selected



**26.6.2.6. Baud Rate Select (BAUDR)**

**Address: QSPIn\_BASEADDR+0x0000\_0014**



**Figure 26-8: Baud Rate Select (BAUDR)**

**SCKDV[15:0]** — 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:

$$F_{sclk\_out} = F_{ssi\_clk} / SCKDV$$

Where SCKDV is any even value between 2 and 65534. For example: for  $F_{ssi\_clk} = 3.6864\text{MHz}$  and  $SCKDV = 2$   $F_{sclk\_out} = 3.6864/2 = 1.8432\text{MHz}$ .

26.6.2.7. Transmit FIFO Threshold Level (TXFTLR)

Address: QSPIn\_BASEADDR+0x0000\_0018

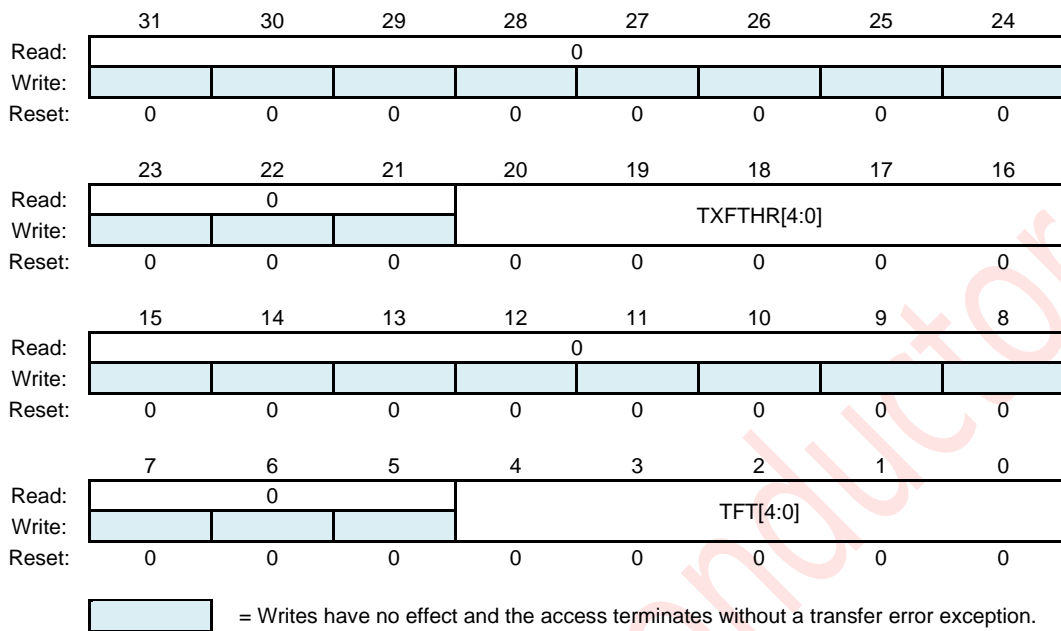


Figure 26-9: Transmit FIFO Threshold Level (TXFTLR)

This register controls the threshold value for the transmit FIFO memory. The SSI is enabled and disabled by writing to the SSIENR register.

**TXFTHR[4:0]** — Transfer start FIFO level.

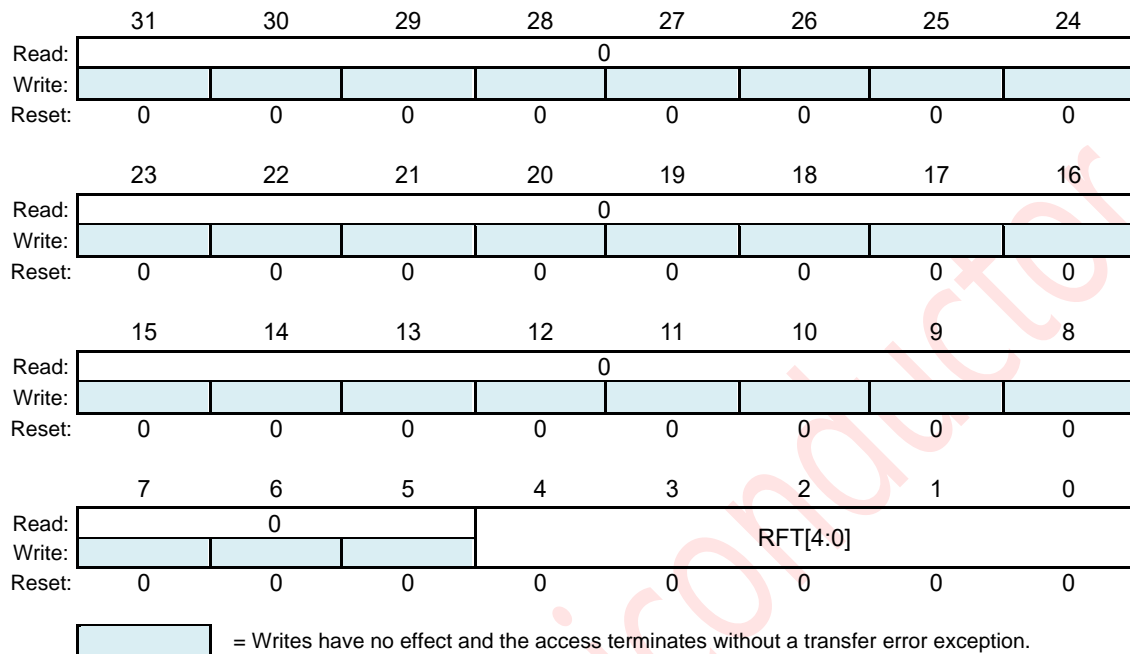
Used to control the level of entries in transmit FIFO above which transfer will start on serial line. This register can be used to ensure that sufficient data is present in transmit FIFO before starting a write operation on serial line. This field is valid only for Master mode of operation.

**TFT[4:0]** — 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.

**26.6.2.8. Receive FIFO Threshold Level (RXFTLR)**

**Address: QSPIn\_BASEADDR+0x0000\_001C**



**Figure 26-10: Receive FIFO Threshold Level (RXFTLR)**

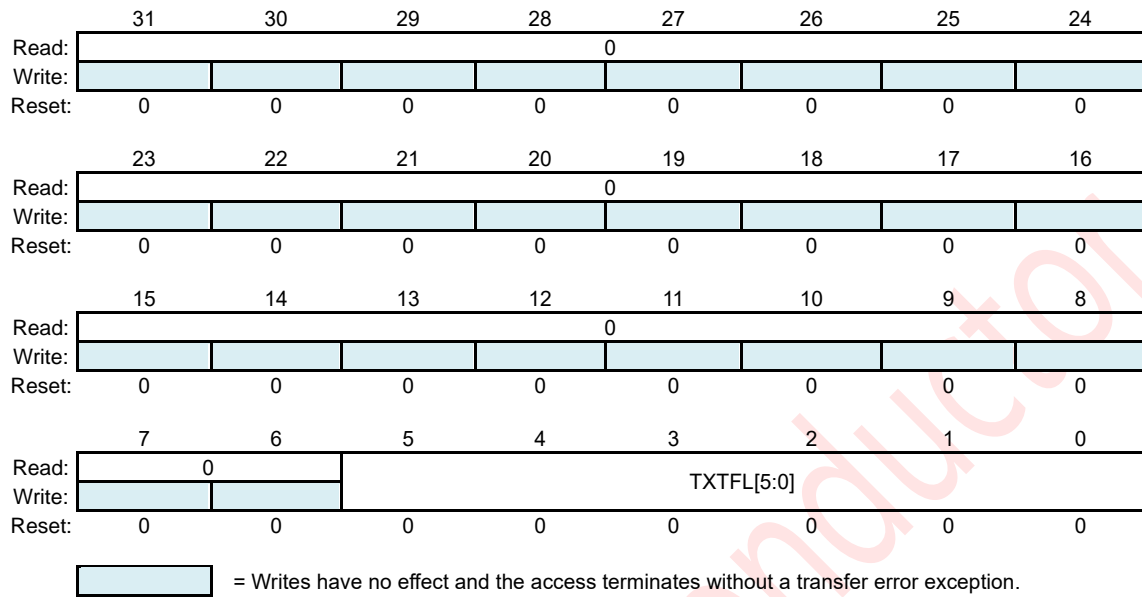
This register controls the threshold value for the receive FIFO memory. The SSI is enabled and disabled by writing to the SSIENR register.

**RFT[4:0]** — 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.

**26.6.2.9. Transmit FIFO Level Register (TXFLR)**

**Address: QSPIn\_BASEADDR+0x0000\_0020**



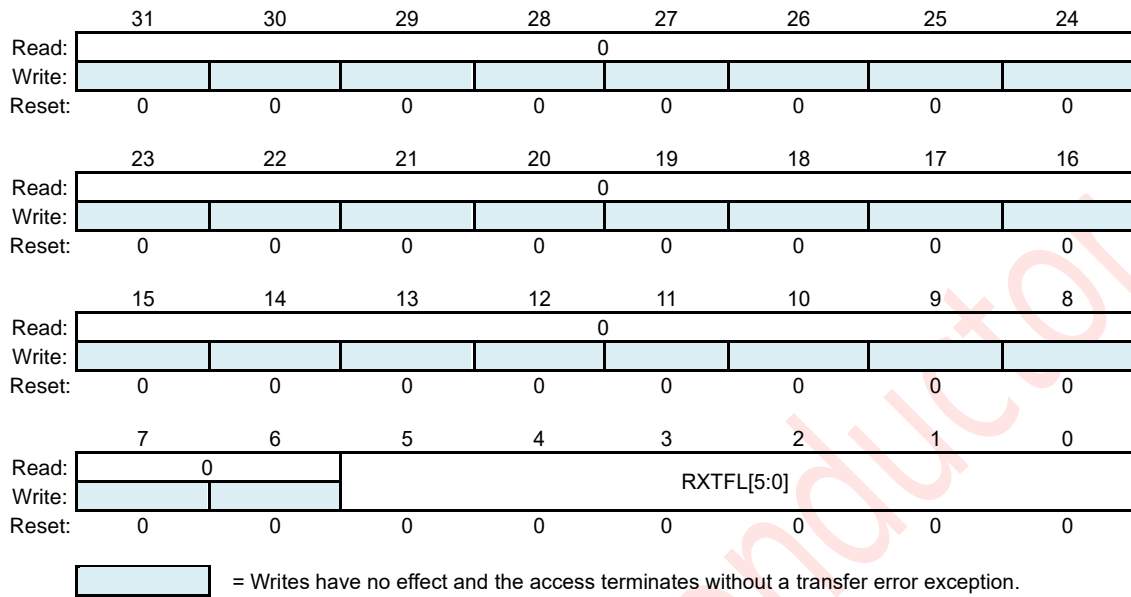
**Figure 26-11: Transmit FIFO Level Register (TXFLR)**

**TXTFL[5:0]** — Transmit FIFO Level.

Contains the number of valid data entries in the transmit FIFO.

**26.6.2.10. Receive FIFO Level Register (RXFLR)**

**Address: QSPIn\_BASEADDR+0x0000\_0024**



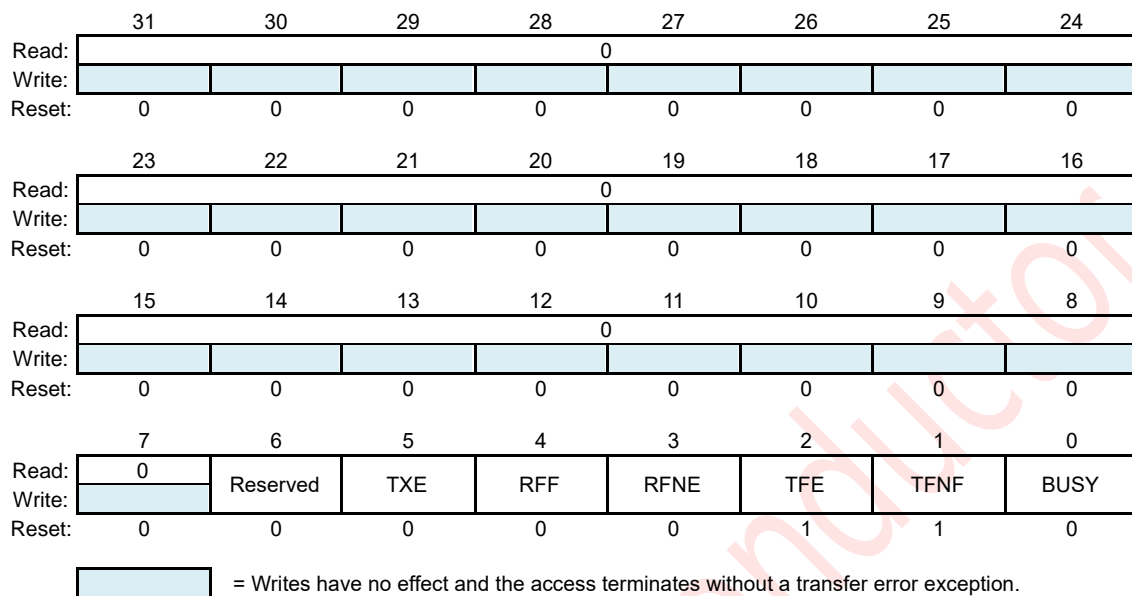
**Figure 26-12: Receive FIFO Level Register (RXFLR)**

**RXTFL[5:0]** — Receive FIFO Level.

Contains the number of valid data entries in the receive FIFO.

**26.6.2.11. Status Register (SR)**

**Address: QSPIn\_BASEADDR+0x0000\_0028**



**Figure 26-13: 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.

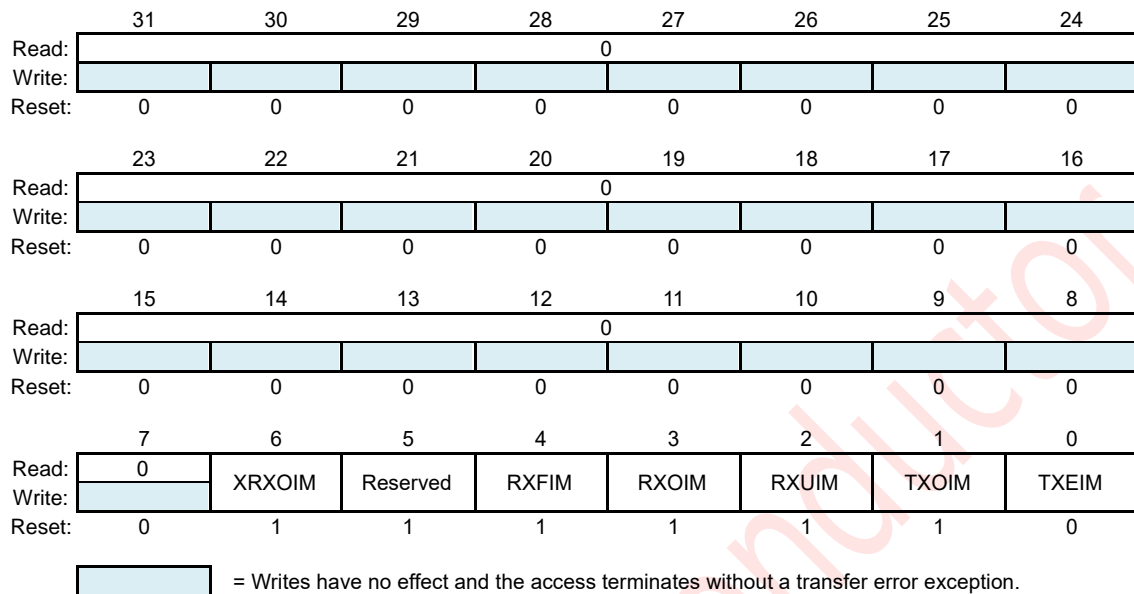
1 = (ACTIVE): SSI is actively transferring data

0 = (INACTIVE): SSI is idle or disabled

Levetop Semiconductor

**26.6.2.12. Interrupt Mask Register (IMR)**

**Address: QSPIn\_BASEADDR+0x0000\_002C**



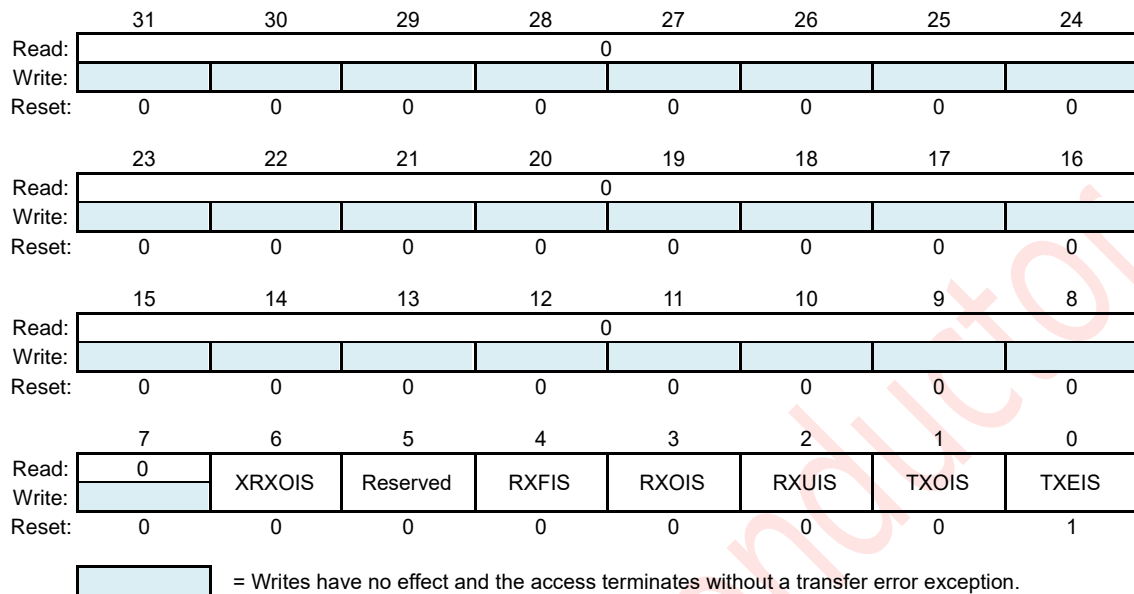
**Figure 26-14: Interrupt Mask Register (IMR)**

- XR XOIM** — 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



**26.6.2.13. Interrupt Status Register (ISR)**

**Address: QSPIn\_BASEADDR+0x0000\_0030**



**Figure 26-15: Interrupt Status Register (ISR)**

This register reports the status of the SSI interrupts after they have been masked.

**XR XOIS** — 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

**RX FIS** — 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

**RX OIS** — 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

**RX UIS** — 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

**TX OIS** — 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

**TX EIS** — 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

26.6.2.14. Raw Interrupt Status Register (RISR)

Address: QSPIn\_BASEADDR+0x0000\_0034

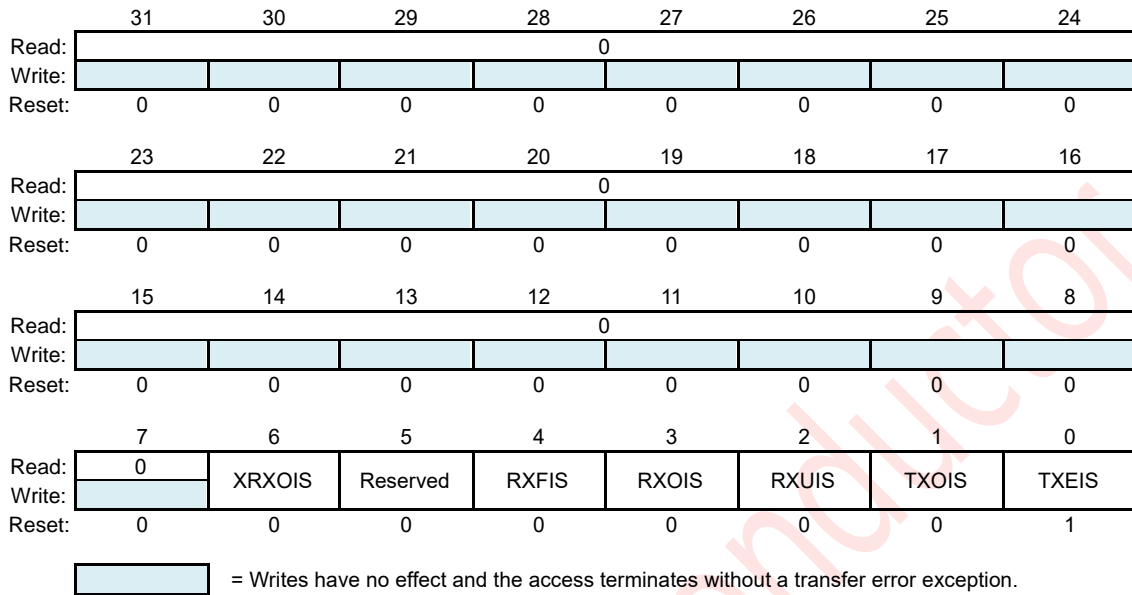


Figure 26-16: Raw Interrupt Status Register (RISR)

**XR XOIS** — 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

**RXFIS** — 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

**RXOIS** — 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

**RXUIS** — 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

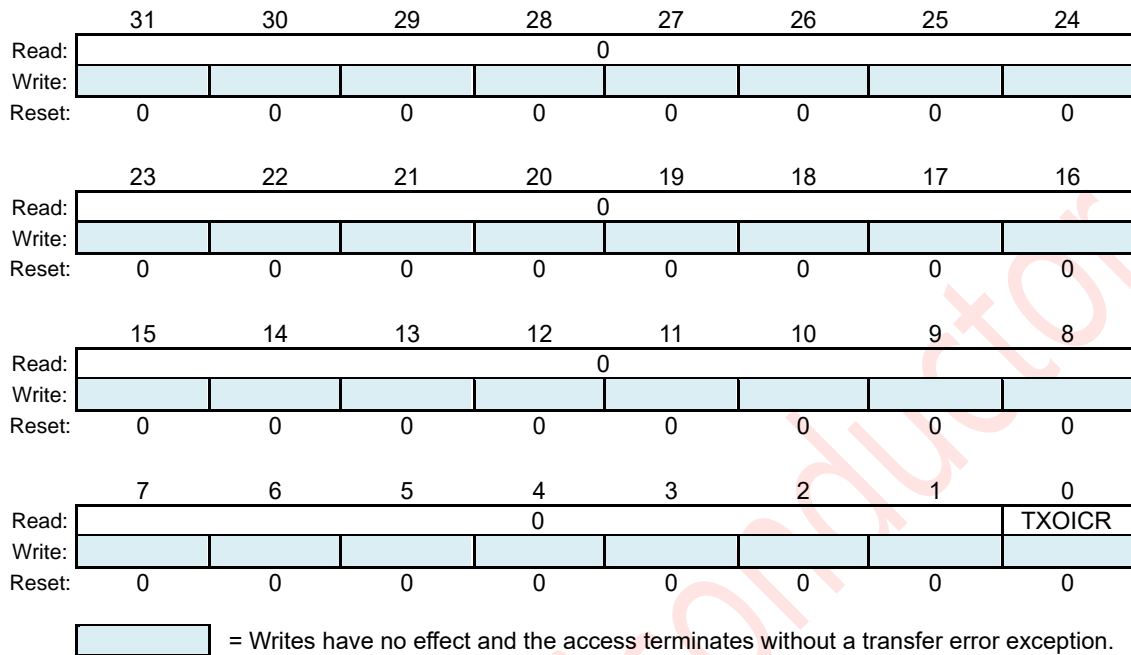
- 1 = (ACTIVE): ssi\_txo\_intr interrupt is active prior to masking
- 0 = (INACTIVE): ssi\_txo\_intr interrupt is not active prior masking

**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

**26.6.2.15. Transmit FIFO Overflow Interrupt Clear Registers (TXOICR)**

**Address: QSPIn\_BASEADDR+0x0000\_0038**



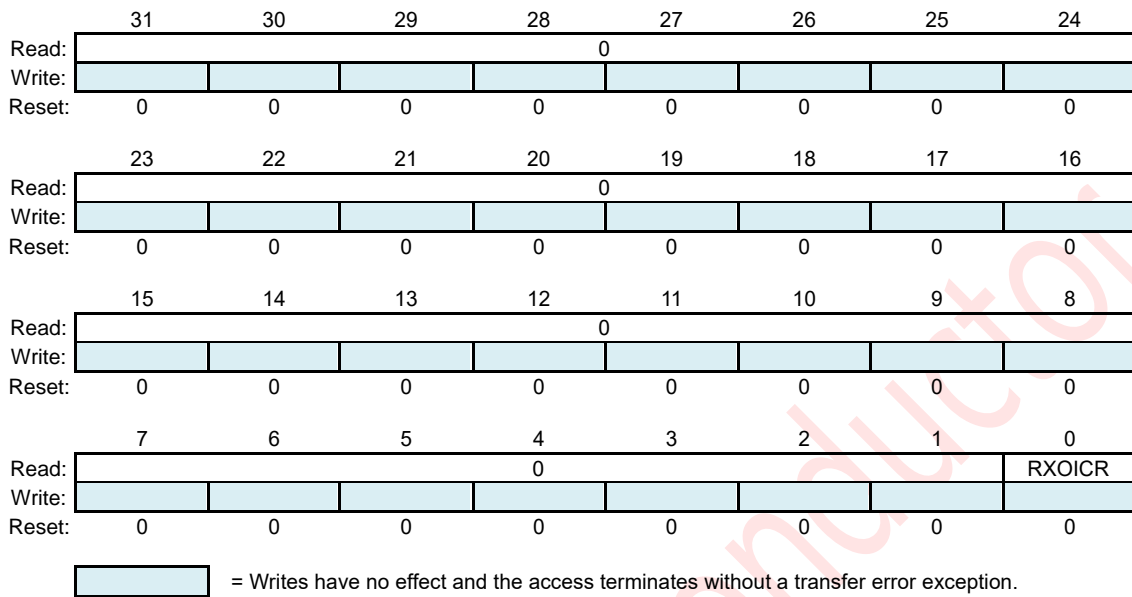
**Figure 26-17: Transmit FIFO Overflow Interrupt Clear Registers (TXOICR)**

**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.

**26.6.2.16. Receive FIFO Overflow Interrupt Clear Register (RXOICR)**

**Address: QSPIn\_BASEADDR+0x0000\_003C**



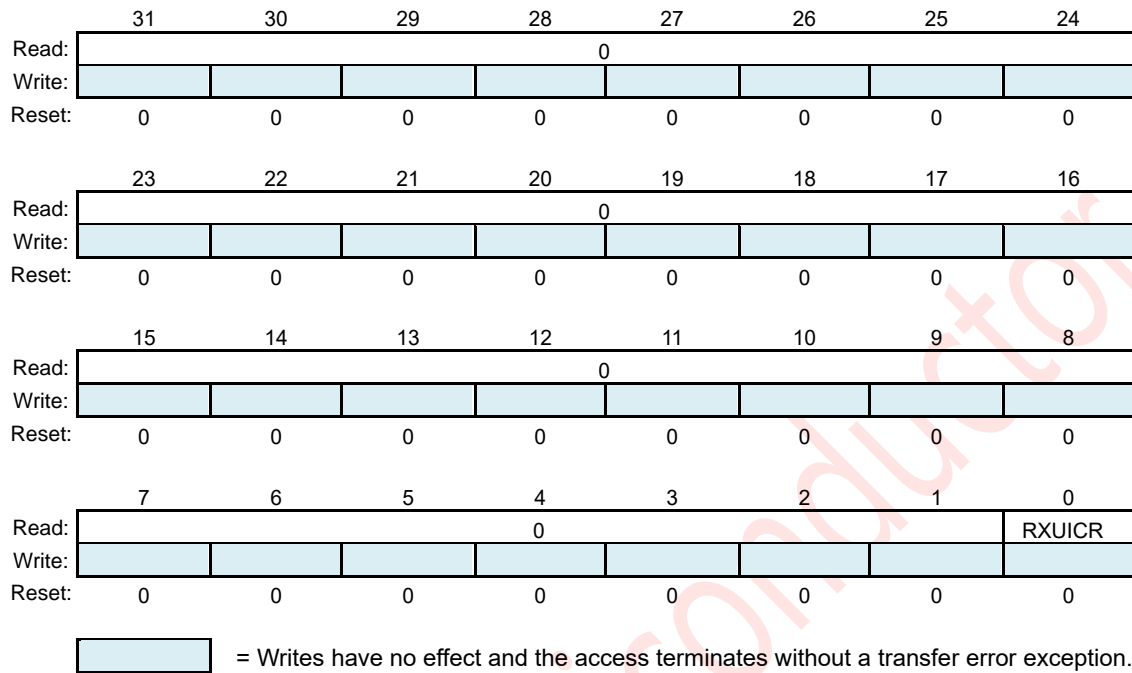
**Figure 26-18: Receive FIFO Overflow Interrupt Clear Register (RXOICR)**

**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.

**26.6.2.17. Receive FIFO Underflow Interrupt Clear Register (RXUICR)**

**Address: QSPIn\_BASEADDR+0x0000\_0040**



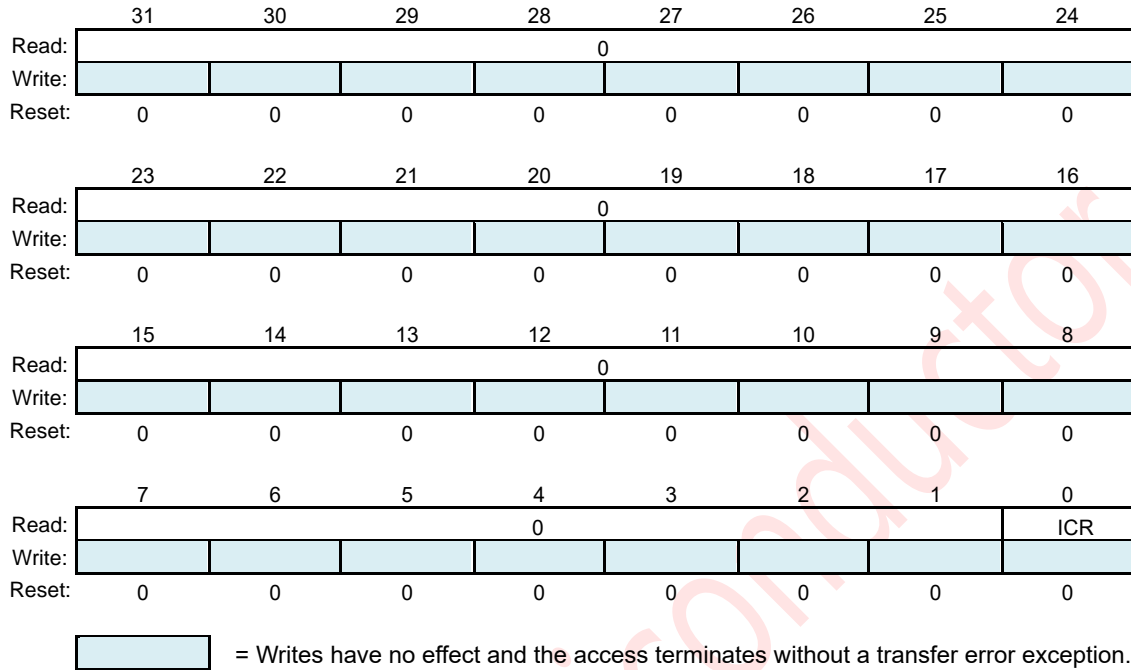
**Figure 26-19: Receive FIFO Underflow Interrupt Clear Register (RXUICR)**

**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.

**26.6.2.18. Interrupt Clear Register (ICR)**

**Address: QSPIn\_BASEADDR+0x0000\_0048**



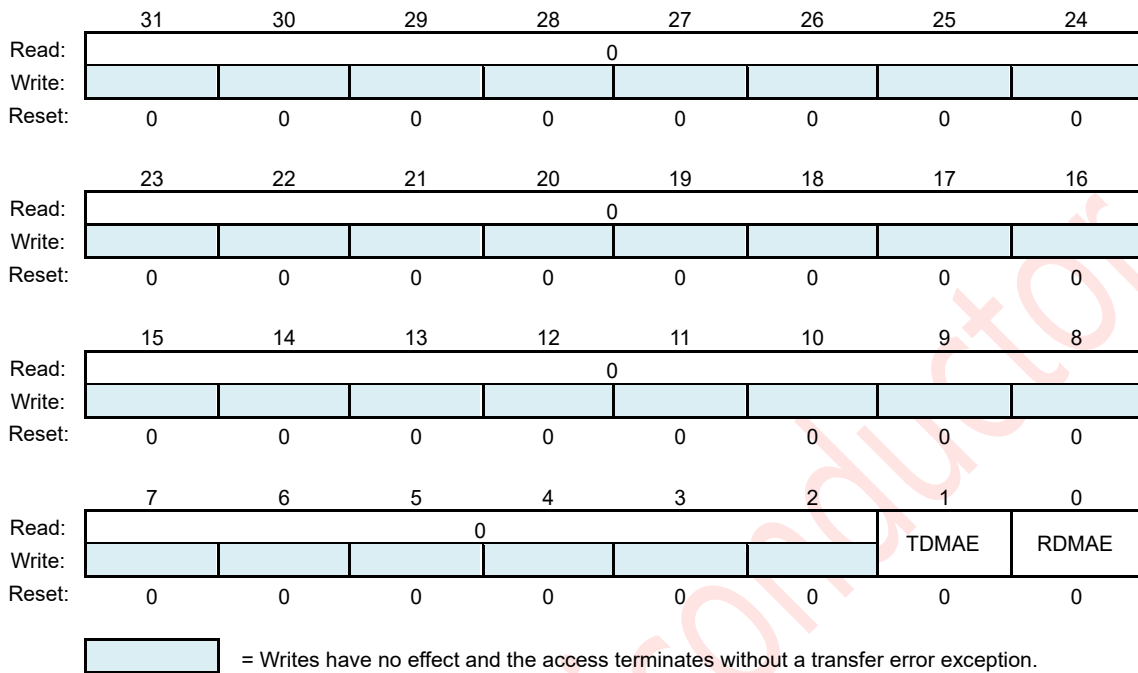
**Figure 26-20: Interrupt Clear Register (ICR)**

**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.

**26.6.2.19. DMA Control Register (DMACR)**

**Address: QSPIn\_BASEADDR+0x0000\_004C**



**Figure 26-21: 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 = Transmit DMA is enabled
- 0 = Transmit DMA is disabled

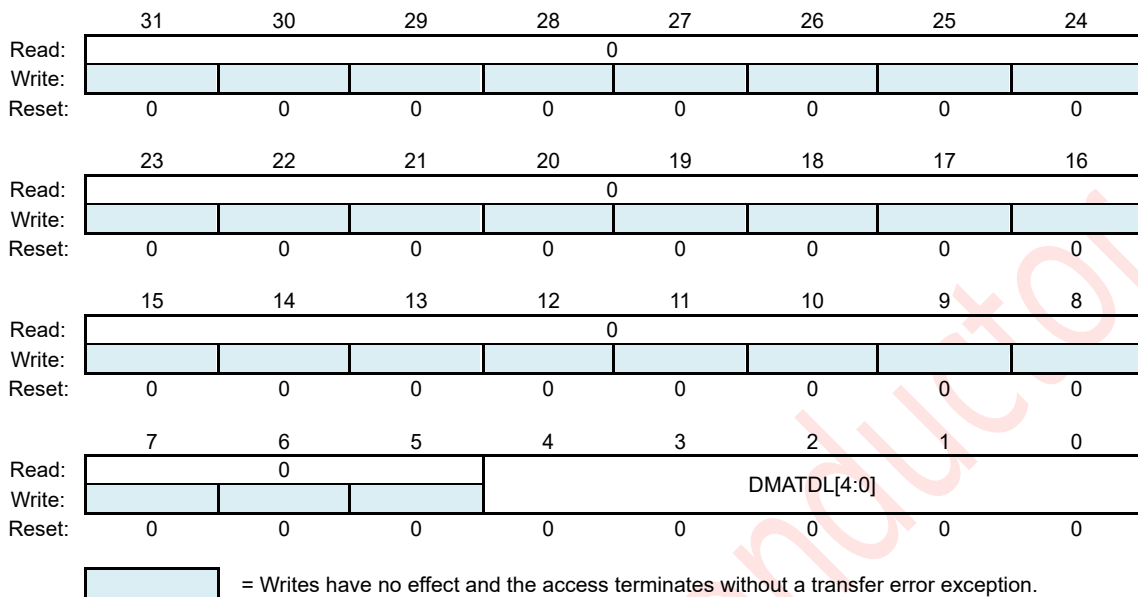
**RDMAE** — Receive DMA Enable

This bit enables/disables the receive FIFO DMA channel.

- 1 = Receive DMA is enabled
- 0 = Receive DMA is disabled

**26.6.2.20. DMA Transmit Data Level (DMATDLR)**

**Address: QSPIn\_BASEADDR+0x0000\_0050**



**Figure 26-22: 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.

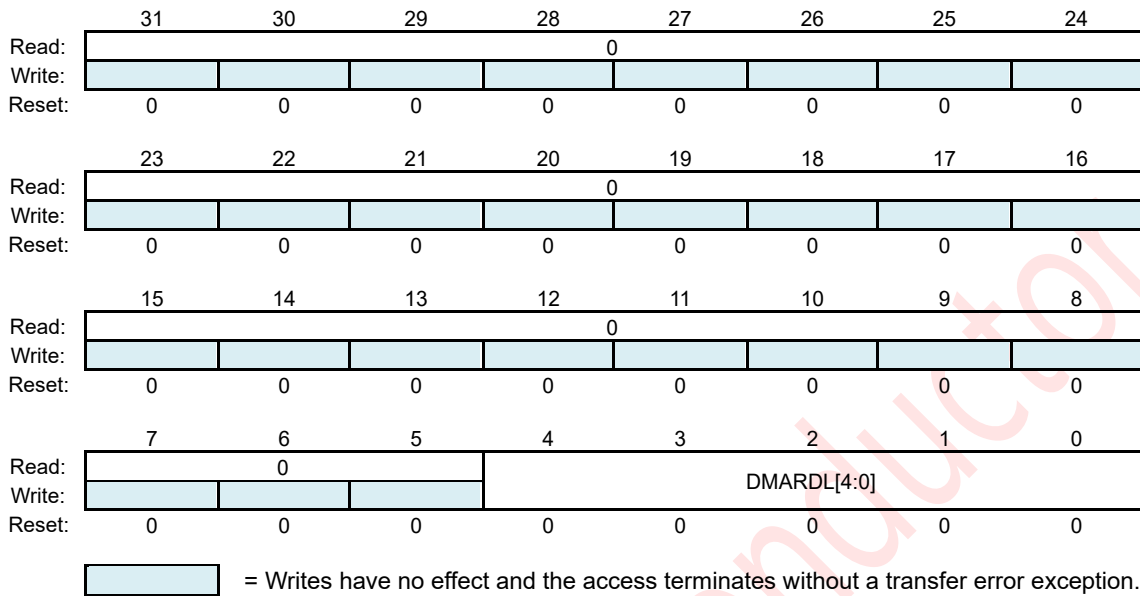
**DMATDL[4:0]** — Transmit Data Level.

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.



**26.6.2.21. DMA Receive Data Level (DMARDLR)**

**Address: QSPIn\_BASEADDR+0x0000\_0054**



**Figure 26-23: DMA Receive Data Level (DMARDLR)**

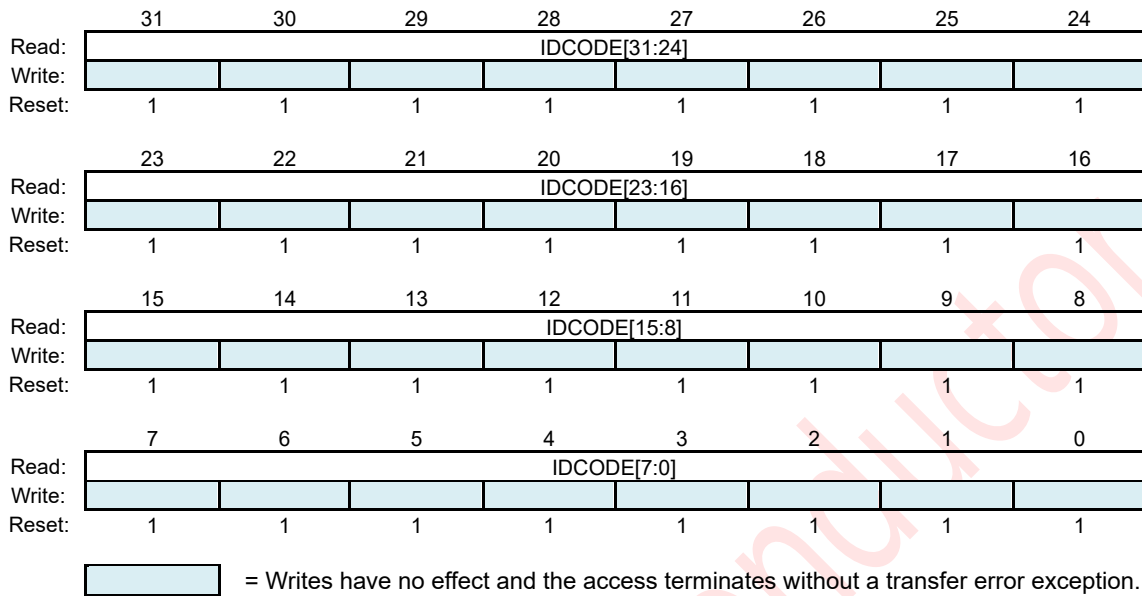
This register is only valid when SSI is configured with a set of DMA interface signals (SSIC\_HAS\_DMA = 1). When SSI is not configured for DMA operation, this register will not exist and writing to its address will have no effect; reading from its address will return zero.

**DMARDL[4:0]** — 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.

**26.6.2.22. Identification Register (IDR)**

**Address: QSPIn\_BASEADDR+0x0000\_0058**



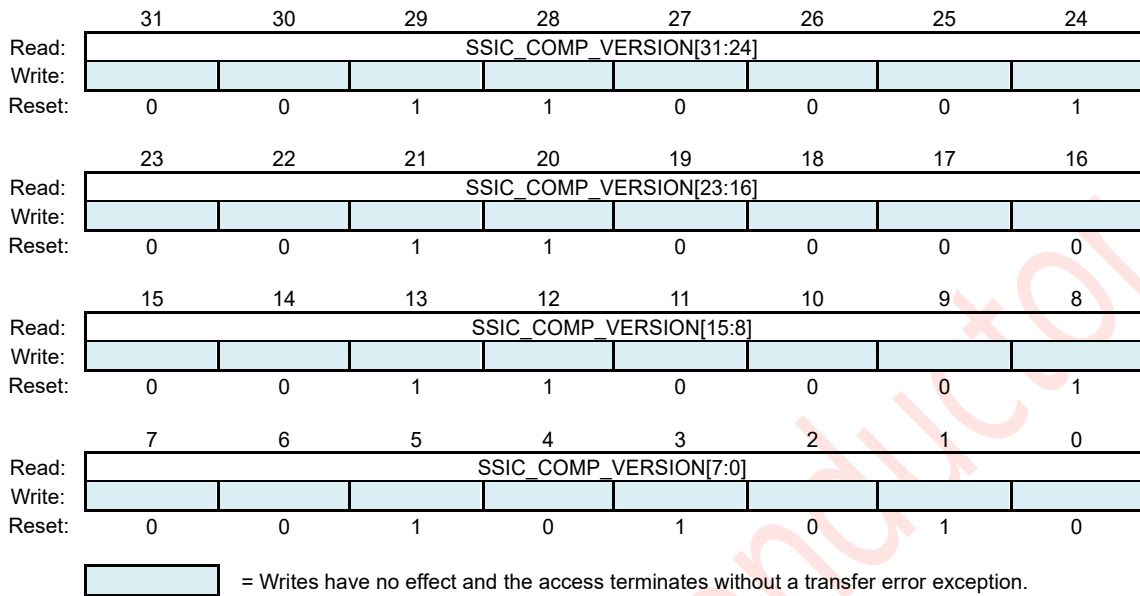
**Figure 26-24: Identification Register (IDR)**

**IDCODE[31:0]** — Identification code.

The register contains the peripheral's identification code, which is written into the register at configuration time using CoreConsultant.

**26.6.2.23. Version ID Register (VIDR)**

**Address: QSPIn\_BASEADDR+0x0000\_005C**



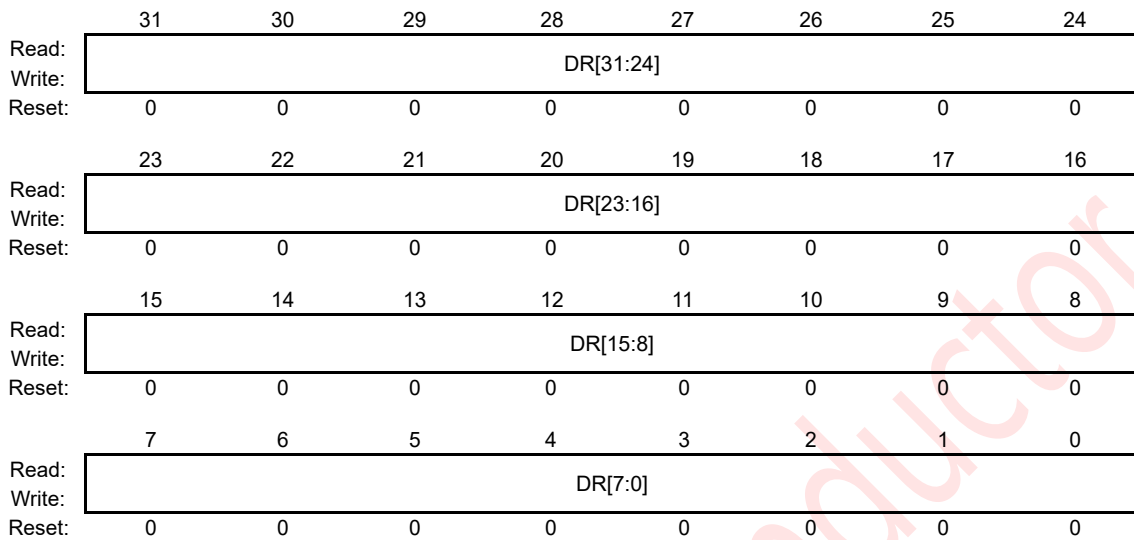
**Figure 26-25: Version ID Register (VIDR)**

**SSIC\_COMP\_VERSION[31:0]**

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\*.

**26.6.2.24. SSI Data Register (DRx)**

**Address: QSPIn\_BASEADDR+0x0000\_0060**



**Figure 26-26: SSI Data Register (DRx)**

The SSI data register is a 32-bits 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[31:0]** — Data Register.

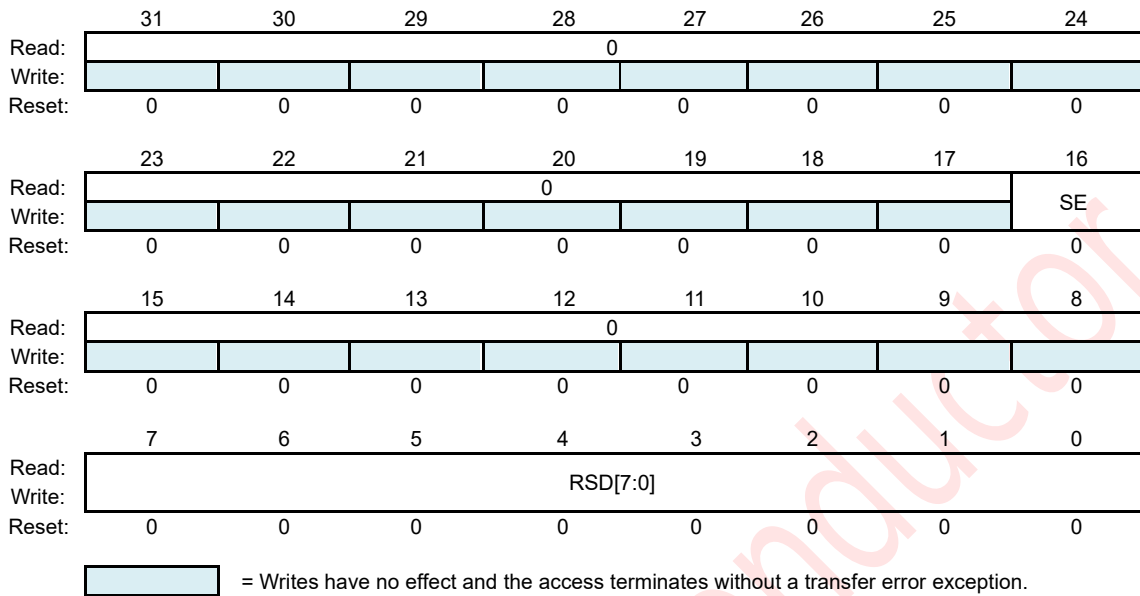
When writing to this register, you must right-justify the data. Read data are automatically right-justified.

Read = Receive FIFO buffer

Write = Transmit FIFO buffer.

**26.6.2.25. RX Sample Delay Register (RXSDR)**

**Address: QSPIn\_BASEADDR+0x0000\_00F0**



**Figure 26-27: 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

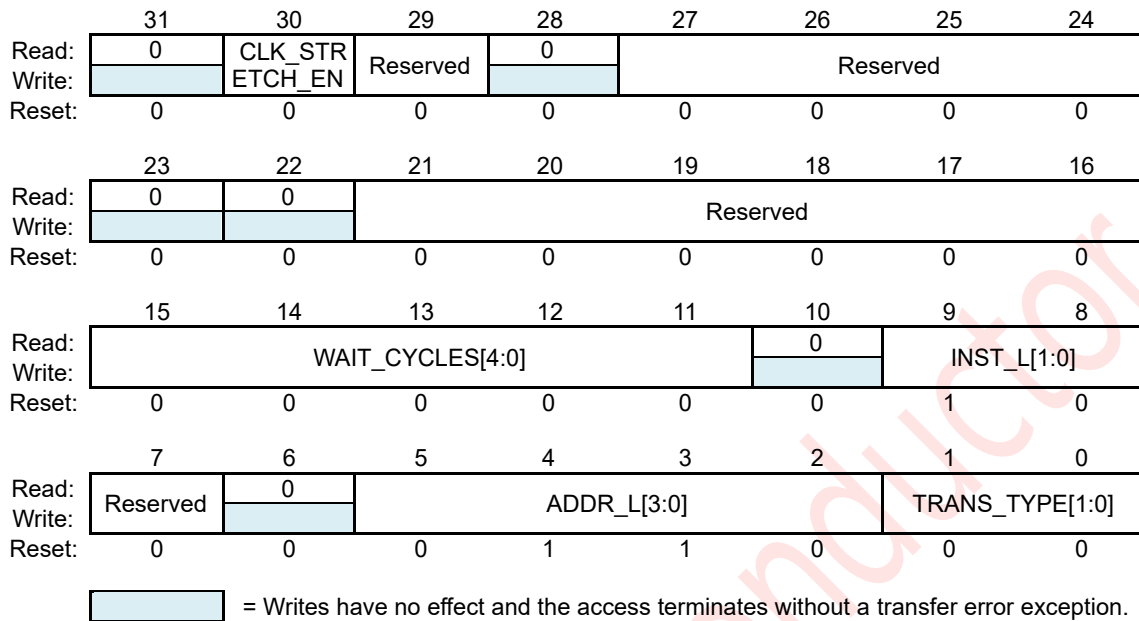
0 = positive edge of ssi\_clk will be used to sample the incoming data

**RSD[7:0]** — 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.

**26.6.2.26. SPI Control Register 0 (SPICTRLR0)**

**Address: QSPIn\_BASEADDR+0x0000\_00F4**



**Figure 26-28: 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[4:0]** — Wait cycles in Dual/Quad/Octal mode between control frames transmit and data reception. Specified as number of SPI clock cycles.

**INST\_L[1:0]** — Dual/Quad/Octal mode instruction length in bits.

- 0x0 (INST\_L0): No Instruction
- 0x1 (INST\_L4): 4-bits Instruction length
- 0x2 (INST\_L8): 8-bits Instruction length
- 0x3 (INST\_L16): 16-bits Instruction length

**ADDR\_L[3:0]** — Length of Address to be transmitted

- 0x0 (ADDR\_L0): No Address
- 0x1 (ADDR\_L4): 4-bits Address length
- 0x2 (ADDR\_L8): 8-bits Address length
- 0x3 (ADDR\_L12): 12-bits Address length
- 0x4 (ADDR\_L16): 16-bits Address length
- 0x5 (ADDR\_L20): 20-bits Address length

- 0x6 (ADDR\_L24): 24-bits Address length
- 0x7 (ADDR\_L28): 28-bits Address length
- 0x8 (ADDR\_L32): 32-bits Address length
- 0x9 (ADDR\_L36): 36-bits Address length
- 0xA (ADDR\_L40): 40-bits Address length
- 0xB (ADDR\_L44): 44-bits Address length
- 0xC (ADDR\_L48): 48-bits Address length
- 0xD (ADDR\_L52): 52-bits Address length
- 0xE (ADDR\_L56): 56-bits Address length
- 0xF (ADDR\_L60): 60-bits Address length

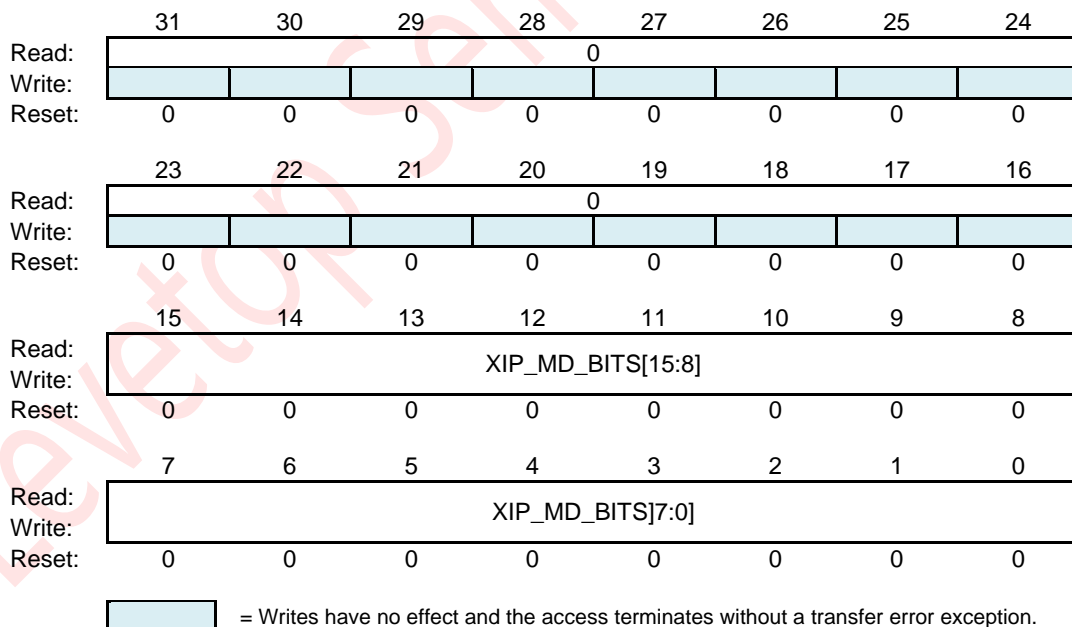
**TRANS\_TYPE[1:0]** — Address and instruction transfer format.

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.

**26.6.2.27. XIP Mode Bits (XIPMBR)**

**Address: QSPIn\_BASEADDR+0x0000\_00FC**



**Figure 26-29: XIP Mode Bits (XIPMBR)**

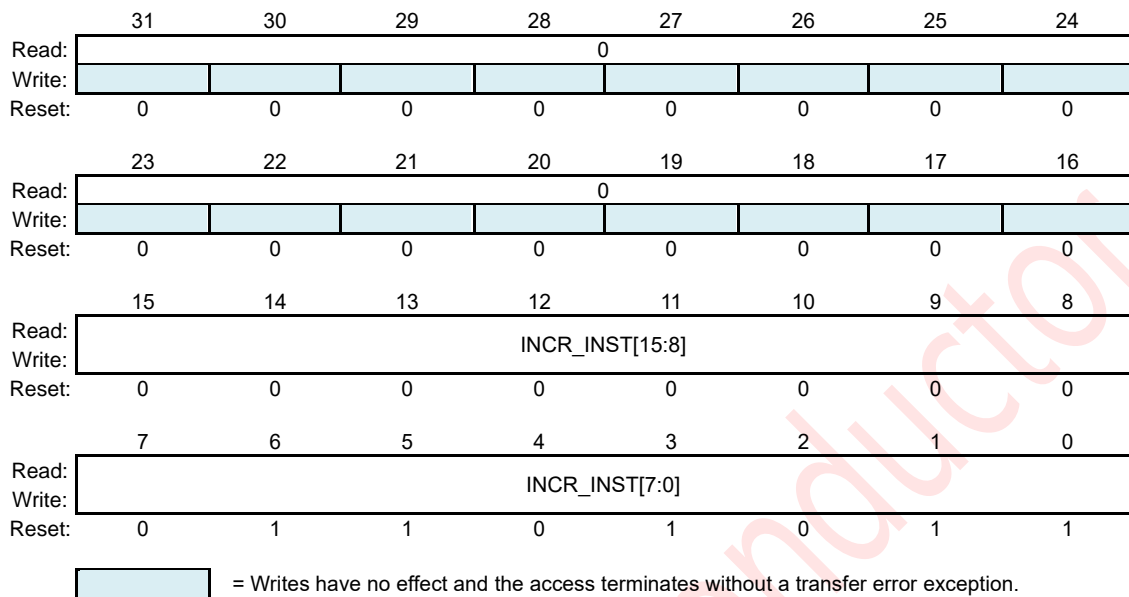
This register carries the mode bits which are sent in the XIP mode of operation after address phase. This is an 8-bits register and can only be written when SSIENR register is set to 0.

**XIP\_MD\_BITS[15:0]**

XIP mode bits to be sent after address phase of XIP transfer.

**26.6.2.28. XIP Incr Inst Register (XIPIIR)**

**Address: QSPIn\_BASEADDR+0x0000\_0100**



**Figure 26-30: 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[15:0]** — 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\_CTRLR0.INST\_L field.



26.6.2.29. XIP Wrap Inst Register (XIPWIR)

Address: QSPIn\_BASEADDR+0x0000\_0104

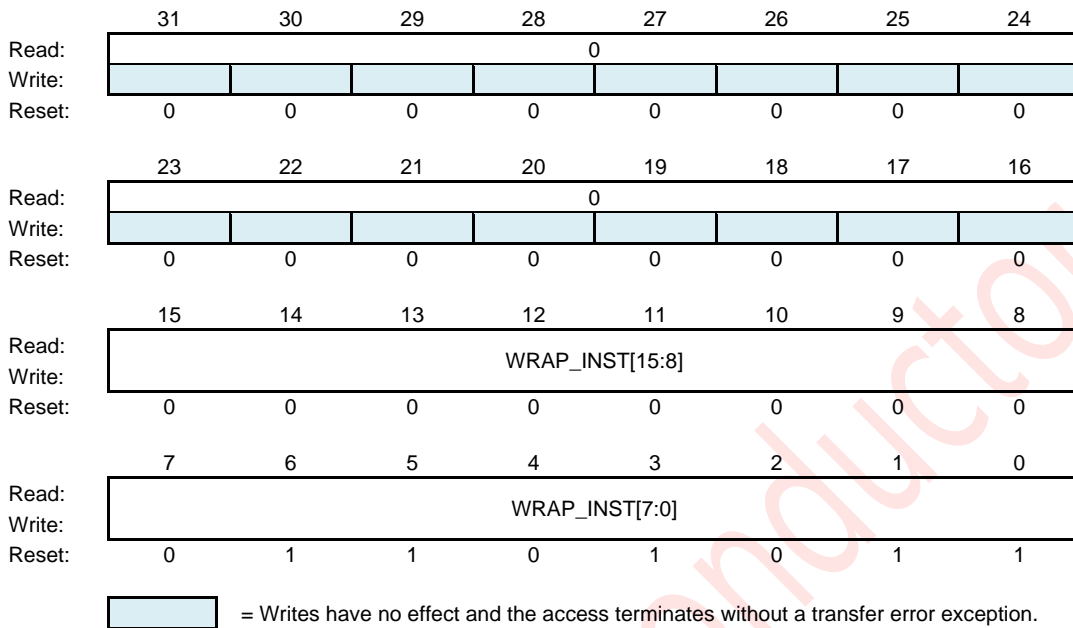


Figure 26-31: XIP Wrap Inst Register (XIPWIR)

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[15:0]** — 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 an WRAP type transfer is requested on AHB bus. The number of bits to be send in instruction phase is determined by SPI\_CTRLR0.INST\_L field.

**26.6.2.30. XIP Control Register (XIPCR)**

**Address: QSPIn\_BASEADDR+0x0000\_0108**

	31	30	29	28	27	26	25	24
Read:	0	0	XIP_PREF	0	XIP_MBL[1:0]		Reserved	
Write:			ETCH_EN					
Reset:	0	0	1	0	1	0	0	0
	23	22	21	20	19	18	17	16
Read:	CONT_XF	INST_EN	Reserved				WAIT_CYCLES[4:3]	
Write:	ER_EN							
Reset:	1	1	0	0	0	0	0	1
	15	14	13	12	11	10	9	8
Read:	WAIT_CYCLES[2:0]			MD_BITS_	0	INST_L[1:0]		0
Write:				EN				
Reset:	0	0	0	0	0	1	0	0
	7	6	5	4	3	2	1	0
Read:	ADDR_L[3:0]				TRANS_TYPE		FRF	
Write:								
Reset:	0	1	1	0	0	0	1	0

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

**Figure 26-32: 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[1:0]** — 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[4:0]** — Wait cycles in Dual/Quad/Octal mode between control frames transmit and data reception. Specified as number of SPI clock 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[1:0]**

Dual/Quad/Octal mode instruction length in bits.

- 0x0 (INST\_L0): No Instruction
- 0x1 (INST\_L4): 4-bits Instruction length
- 0x2 (INST\_L8): 8-bits Instruction length
- 0x3 (INST\_L16): 16-bits Instruction length

#### **ADDR\_L[3:0]**

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): 4-bits Address length
- 0x2 (ADDR\_L8): 8-bits Address length
- 0x3 (ADDR\_L12): 12-bits Address length
- 0x4 (ADDR\_L16): 16-bits Address length
- 0x5 (ADDR\_L20): 20-bits Address length
- 0x6 (ADDR\_L24): 24-bits Address length
- 0x7 (ADDR\_L28): 28-bits Address length
- 0x8 (ADDR\_L32): 32-bits Address length
  
- 0x9 (ADDR\_L36): 36-bits Address length
- 0xA (ADDR\_L40): 40-bits Address length
- 0xB (ADDR\_L44): 44-bits Address length
- 0xC (ADDR\_L48): 48-bits Address length
- 0xD (ADDR\_L52): 52-bits Address length
- 0xE (ADDR\_L56): 56-bits Address length
- 0xF (ADDR\_L60): 60-bits Address length

#### **TRANS\_TYPE[1:0]**

Address and instruction transfer format.

Selects whether SSI will transmit instruction/address either in Standard SPI mode or the SPI mode selected in CTRL0.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[1:0]** — 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

26.6.2.31. XIP Slave Enable Register (XIPSER)

Address: QSPIn\_BASEADDR+0x0000\_010C

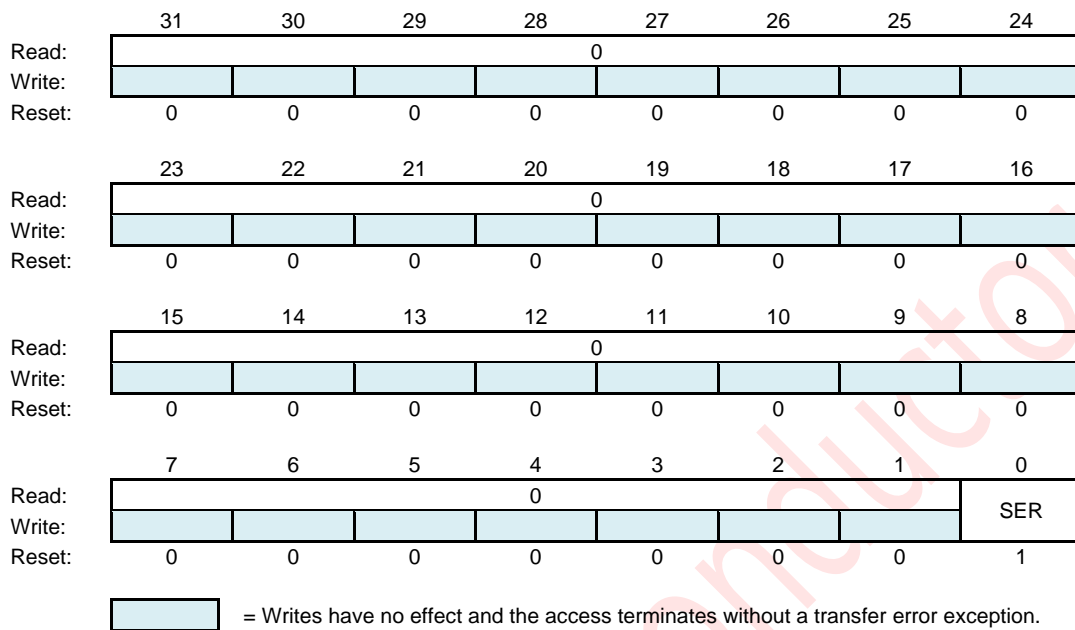


Figure 26-33: XIP Slave Enable Register (XIPSER)

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

XIP Receive FIFO Overflow Interrupt Clear Register (XRXIOCR)

26.6.2.32. XIP Receive FIFO Overflow Interrupt Clear Register (XRXIOCR)

Address: QSPIn\_BASEADDR+0x0000\_0110

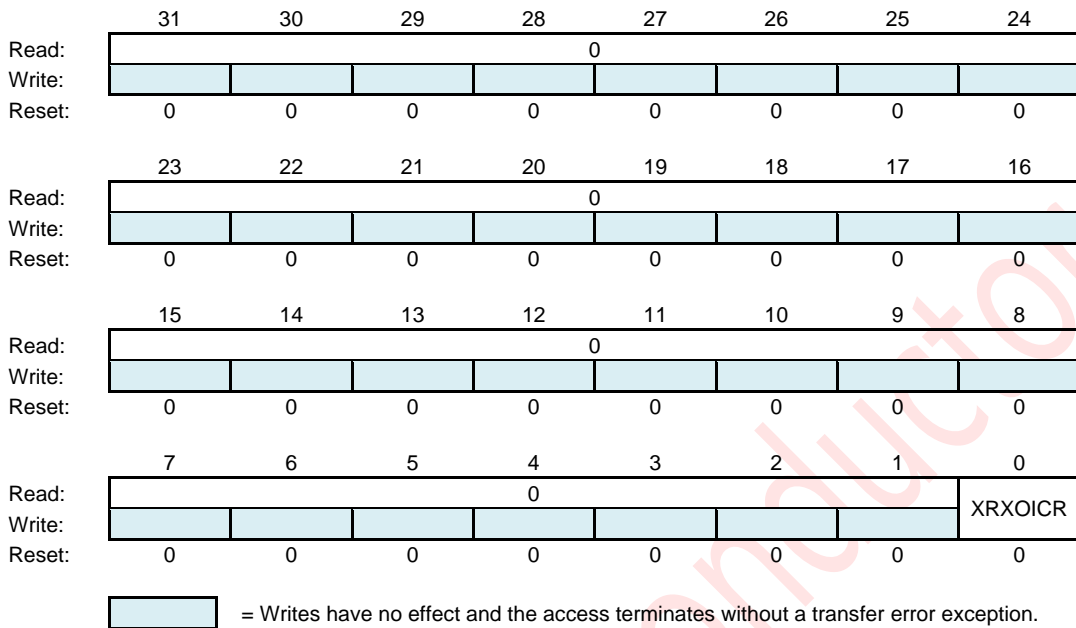


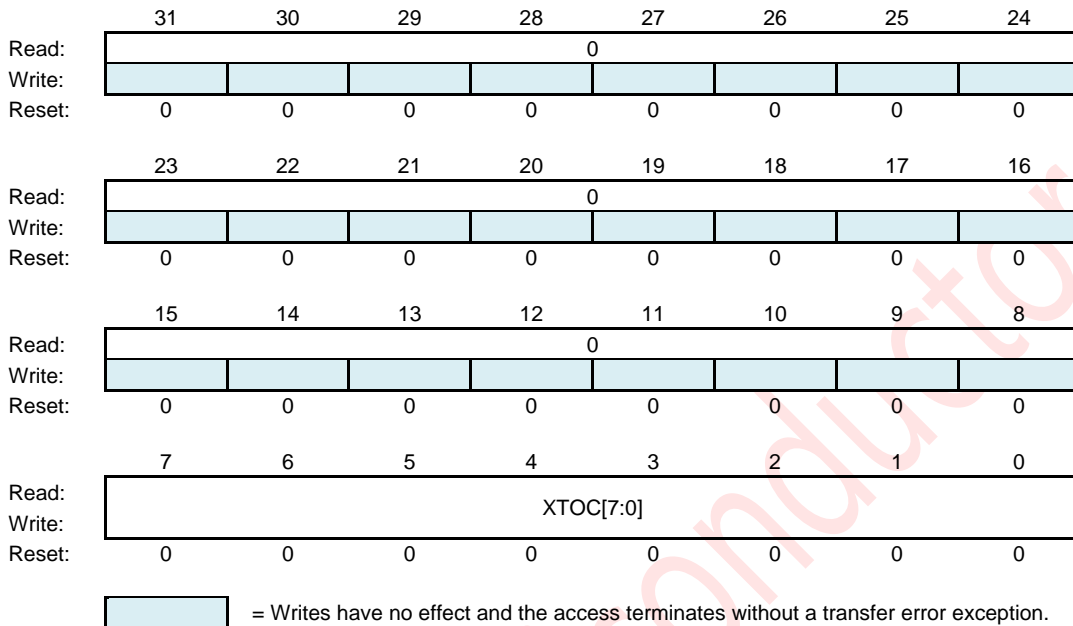
Figure 26-34: XIP Receive FIFO Overflow Interrupt Clear Register (XRXIOCR)

**XRXIOCR** — 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.

**26.6.2.33. XIP Continus Transfer Time Out Register (XIPCTTOR)**

**Address: QSPIn\_BASEADDR+0x0000\_0114**



**Figure 26-35: XIP Continus 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[7:0]** — 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.

## 26.7. Functional Description

### 26.7.1. 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 26-36** 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.

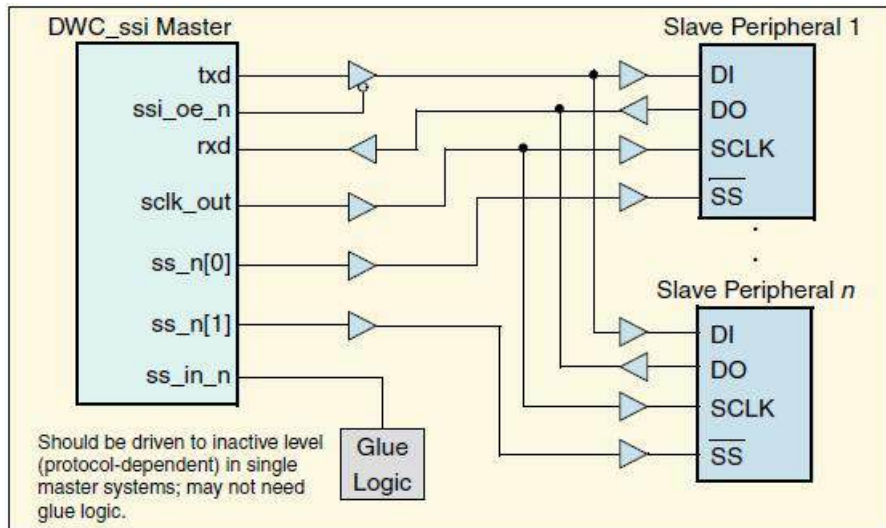


Figure 26-36: SSI Configured as Master Device

### 26.7.2. 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.

### 26.7.3. 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 32-bits due to the serial specifications, which state that a serial transfer (data frame) can be 4 to 32-bits in length. Data frames that are less than 32-bits 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.

#### 26.7.3.1. 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.

#### 26.7.3.2. 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.

### 26.7.4. 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.

### 26.7.5. 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.



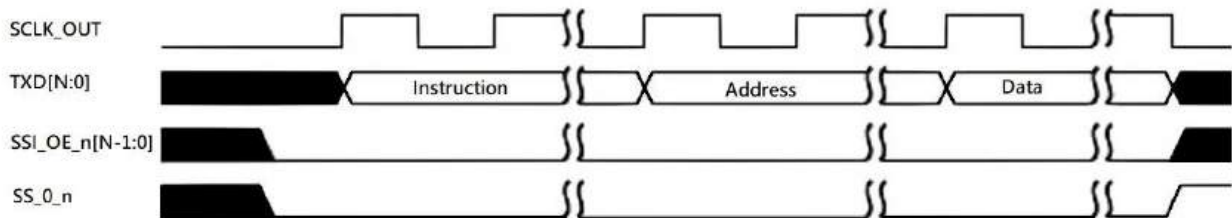
Receive FIFO Overflow Interrupt (ssi\_rxo\_intr) – Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. When set, newly received data are discarded. This interrupt remains set until you read the receive FIFO overflow interrupt clear register (RXOICR).

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).

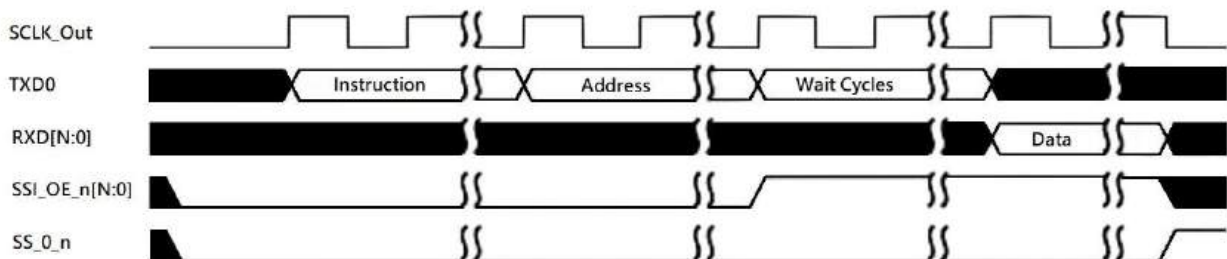
Combined Interrupt Request (ssi\_intr) – OR'ed result of all the above interrupt requests after masking. To mask this interrupt signal, you must mask all other SSI interrupt requests.

**26.7.6. 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.



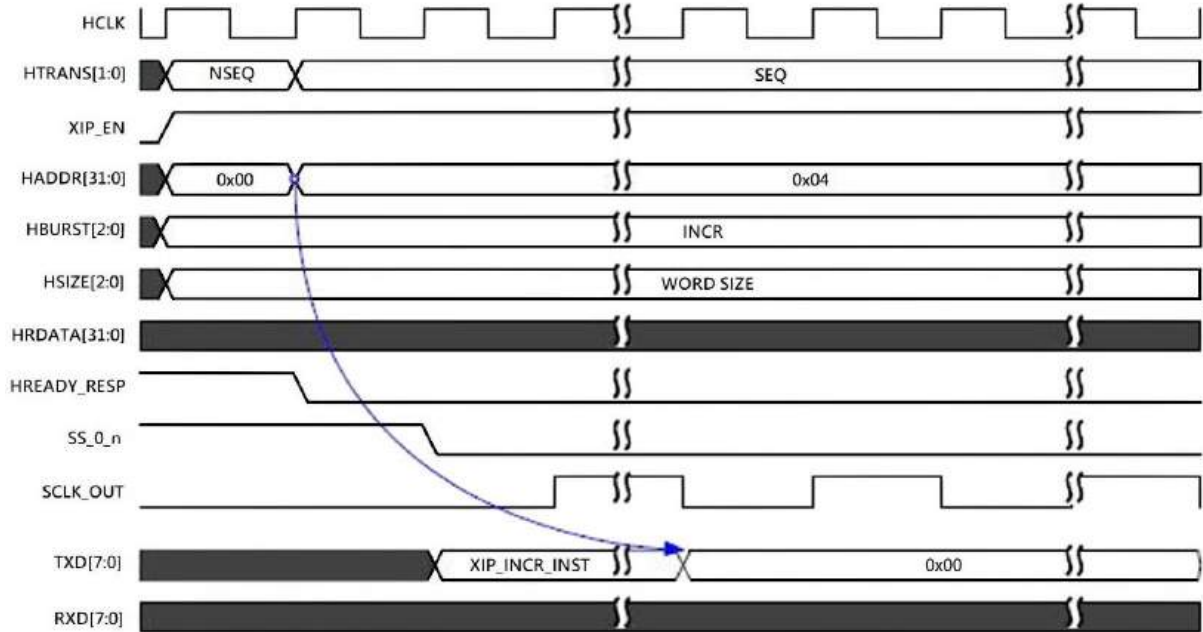
**Figure 26-37: Typical Write Operation Dual/Quad/Octal SPI Mode**



**Figure 26-38: Typical Read Operation Dual/Quad/Octal SPI Mode**

**26.7.7. 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 26-39: XIP Transfer with Instruction Phase**

### 26.7.8. 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.

### 26.7.9. 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.

## 27. Inter-Integrated Circuit (I2C)

### 27.1. 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.

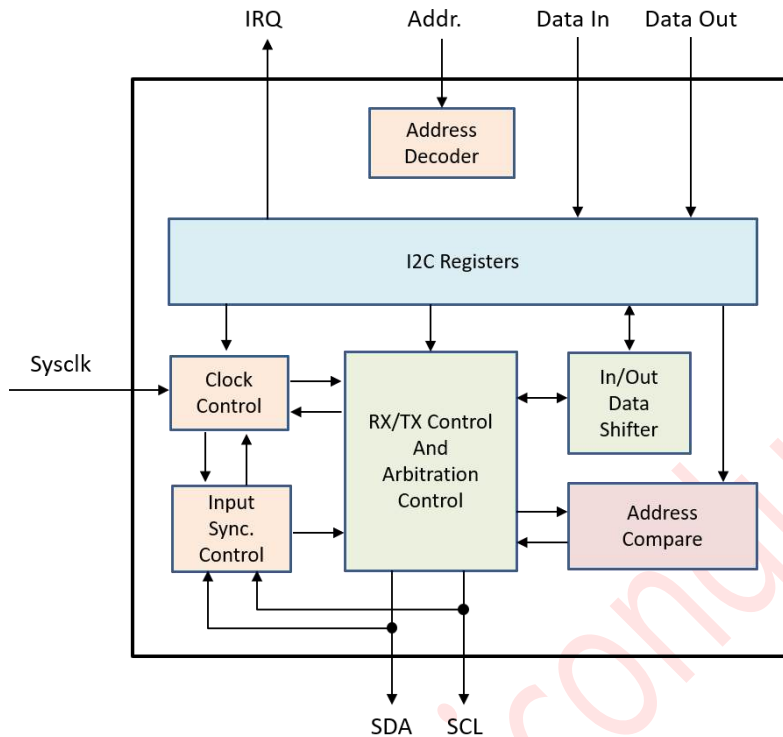
The two bus lines provide data transfer rates up to 100Kbits/s IN Standard Mode, data rates up to 400Kbits/s in Fast Mode, and data rates up to 3.4Mbits/s in High-Speed Mode.

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.

### 27.2. Features

- Supports 7-bits 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

**27.3. System And Block Diagram**



**Figure 27-1: I2C Block Diagram**

**27.4. Memory Map and Registers**

**27.4.1. Memory Map**

The I2C Module base address is **0x4015\_0000**. There are ten registers in the I2C memory map, as shown in **Table 27-1**.

**Table 27-1: I2C Memory Map**


Address Offset	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	Reserved
0x0009	Reserved
0x000A	I2C Port Direction Register (I2CDDR)
0x000B	I2C Port Data Register (I2CPDR)

**27.4.2. Register Descriptions**

**27.4.2.1. I2C Status Register (I2CS)**

**Address: I2C\_BASEADDR+0x0000\_0000**

	7	6	5	4	3	2	1	0
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 27-2: The I2CS Register Shows the Status of I2C Module.**

**AACK** — Address acknowledge error.

This bit indicates whether the acknowledge bit is detected or not by master during address phase. It is set by the rising edge of ninth clock of the address phase and cleared by changing MSMOD from 1 to 0 or repeat start condition.

0 = No address acknowledge error.

1 = Address acknowledge error.

**DACK** — Data acknowledge received.

This bit 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 is received.

1 = Acknowledge bit is received.

**RXTX** — Receive or transmit.

Indicates the I2C module function as receiver or transmitter. It is valid on the falling edge of the eighth clock.

0 = Receive, receive data.

1 = Transmit, transmit data.

**ARBL** — Arbitration lost

This bit 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 reading the I2CS register through software.

0 = Arbitration not lost.

1 = Arbitration lost.

**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.

**AASLV** — Addressed as a slave

This bit is set on 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 cleared if START or STOP bit is detected

- 0 = Not as a slave.
- 1 = Addressed as a slave.

**RC** — Receive Complete

This bit indicates it is time to configure the receiver. For master receiver, It is set on the falling edge of the ninth clock of data or address byte transferred regardless of acknowledge bit detected or not. For slave receiver, if the acknowledgement bit was detected, this bit will be set by the falling edge of the ninth clock of address or data byte 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 write I2CC.

- 0 = meaningless,
- 1 = Indicate it is time to configure the receiver.

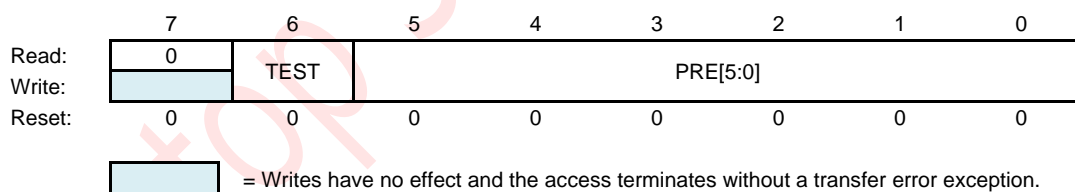
**TF** — Transfer Complete Flag

This bit 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 no matter the acknowledge bit is detected or not. For master transmitter, this bit is set by the falling edge of the ninth clock of a data or address byte sent, no matter the acknowledge bit is detected or not. For slave transmitter, if the acknowledgement bit was detected, this bit will be set by the falling edge of the ninth clock of address or data byte transferred. If the IEN bit in I2CC is also set, an interrupt will be generated. Clear TF by reading I2CS with TF set, and then access I2CD or master-transmit mode to write I2CC.

- 0 = meaningless,
- 1 = Data or address transfer complete.

**27.4.2.2. I2C Clock Prescaler Register (I2CP)**

**Address: I2C\_BASEADDR+0x0000\_0001**



**Figure 27-3: I2C Clock Prescaler Register (I2CP)**

I2CP is a prescaler which generates a bit-rate clock for the data transceiver. Due to the 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.

**TEST** — Clock Test Enable

This bit enables test mode for the I2C clock. In normal mode, its frequency is  $figgclk/(396x(PRE[5:0]+1)+1)$ . In test mode, the frequency is  $figgclk/(8x(PRE[5:0]+1)+1)$ .

- 1 = Clock Test Mode Enable
- 0 = Normal Mode

**PRE[5:0]** — Prescaler Divider Value

**27.4.2.3. I2C Control Register (I2CC)**

**Address: I2C\_BASEADDR+0x0000\_0002**

	7	6	5	4	3	2	1	0
Read:	SLV_HSIE	HSM_EN	AMIE	REPSTA	ACKEN	MSMOD	IEN	EN
Write:								
Reset:	0	0	0	0	1	0	0	0

**Figure 27-4: I2C Control Register (I2CC)**

I2CC is used to control the working of I2C module.

**SLV\_HSIE** — Slave High-Speed Mode Interrupt Enable

This bit selects whether Slave High-Speed Mode status will generate interrupt requests if I2C interrupt enable control (IEN) is set to 1.

- 1 = Enable Slave High-Speed Mode status request.
- 0 = Disable Slave High-Speed Mode status request.

**HSM\_EN** — High Speed Mode Enable

This bit 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 lose the arbitration, HSM\_EN should be set by software, and if users prepare to clear MSMOD to transfer STOP, HSM\_EN should be cleared by software after MSMOD.

- 0 = Fast/Standard mode operation (Default)
- 1 = High-Speed Mode operation

**AMIE** — Address Match Interrupt Enable

This bit selects whether Slave address match will generate interrupt requests if I2C interrupt enable control (IEN) is set to 1. AMIE should be set before entering stop mode to wakeup the system when slave address matched and must be cleared at normal work mode.

- 1 = Enable address match interrupt request.
- 0 = Disable address match interrupt request.

**REPSTA** — Repeat Start

In these cases: 1) The master receiver is not 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 sent when configuring slave address(by writing I2CD) after REPSTA bit is set.

- 1 = Generate repeat start.
- 0 = No repeat start.

**ACKEN** — Acknowledge enable control.

This bit 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 bytes. 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.



**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.

**IEN** — I2C interrupt enable control.

It enables the I2C interrupt when it is set.

- 1 = I2C interrupt is enabled.
- 0 = I2C interrupt is disabled.

**EN** — I2C enable/disable control.

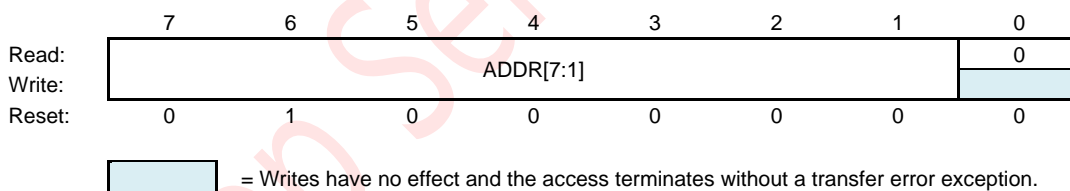
- 1 = module is enabled.
- 0 = module is disabled.

It enables/disables the module, and 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.

**27.4.2.4. I2C Slave Address Register (I2CSA)**

**Address: I2C\_BASEADDR+0x0000\_0003**



**Figure 27-5: I2C Slave Address Register (I2CSA)**

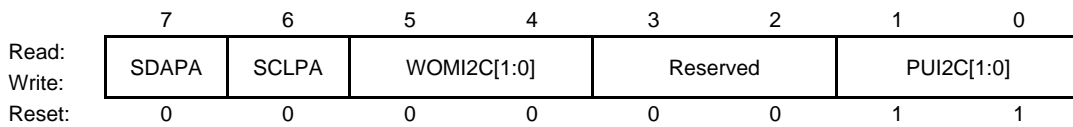
The I2CSA stores the address when the I2C works as slave and responds to the address sent by a master.

**ADDR[7:1]** — Module Slave Address.

These bits indicate slave address when the I2C module is in slave mode. (I2C is slave by default).

**27.4.2.5. I2C Port Control Register (I2CPCR)**

**Address: I2C\_BASEADDR+0x0000\_0004**



**Figure 27-6: I2C Port Control Register (I2CPCR)**

**SDAPA** — SDA Port Assignment Bit.

The read/write bit selects the SDA’s function mode.

- 1 = Pin is configured as GPIO\_SDA.
- 0 = Pin is configured as primary function.

**SCLPA** — SCL Port Assignment Bit.

The read/write bit selects the SCL’s function mode.

- 1 = Pin is configured as GPIO\_SCL.
- 0 = Pin is 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.

**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 input mode and main function mode.

- 1 = Pullup is Enabled.
- 0 = Pullup is Disabled.

**27.4.2.6. I2C Slave High-Speed Mode Indicator Register (I2CSHIR)**

**Address: I2C\_BASEADDR+0x0000\_0005**

	7	6	5	4	3	2	1	0
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 27-7: I2C Slave High-Speed Mode Indicator Register (I2CSHIR)**

**SLV\_HS** — Slave High Speed Mode.

When I2C is selected as a 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 forced to be LOW state.

0 = I2C is selected for Fast/Standard data transfer (Default).

1 = I2C is selected for High-Speed mode data transfer.

**27.4.2.7. I2C Slave SDA Hold Time Register (I2CSHT)**

**Address: I2C\_BASEADDR+0x0000\_0006**

	7	6	5	4	3	2	1	0
Read:	SCL_FILTE	SDA_FILTE	SLVHT					
Write:	R_EN	R_EN						
Reset:	0	0	0	0	1	0	0	1

**Figure 27-8: 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 filtered out. If the F/S mode transfer is ongoing, the 50ns pulse occurred on SCL line will be filtered on.

**SDA\_FILTER\_EN** — SDA Filter enable.

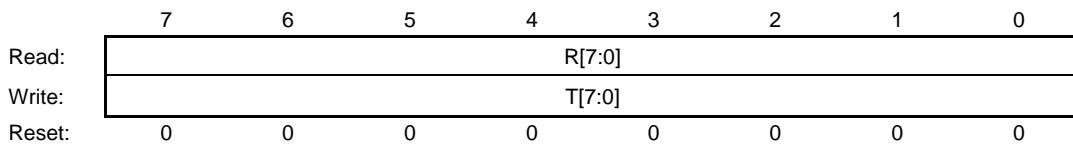
If SDA filter is enabled, when the HS mode transfer is ongoing, the 10ns pulse occurred on SDA line will be filtered out. If the F/S mode transfer is ongoing, the 50ns pulse occurred on SDA line will be filtered on.

**SLVHT** — slave SDA line hold time configuration.

When I2C works as the slave output mode, the data will be changed after SCL falling edge and the value of the internal SDA hold register is equal to SLVHT. Writing value of 0 is not allowed.

**27.4.2.8. I2C Data Register (I2CD)**

**Address: I2C\_BASEADDR+0x0000\_0007**



**Figure 27-9: I2C Data Register (I2CD)**

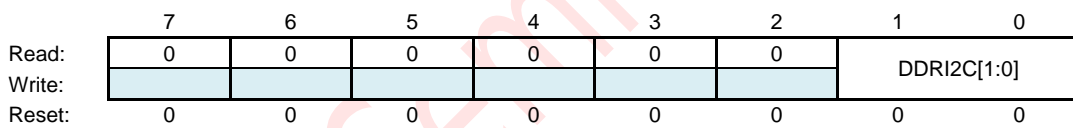
The I2CD register holds the data to be transmitted (next byte) or the data received. In master mode it also holds the slave address and transfer direction to be transmitted. Bits [7:1] form the slave address and bit [0] is the transfer direction (R/W). In master-receiver mode, reading I2CD allows a read 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]

**27.4.2.9. I2C Port Direction Register (I2CDDR)**

**Address: I2C\_BASEADDR+0x0000\_000A**



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

**Figure 27-10: I2C Port Direction Register (I2CDDR)**

Read: Anytime

Write: Anytime

**DDR12C[1:0]** — I2C Data Direction Bits

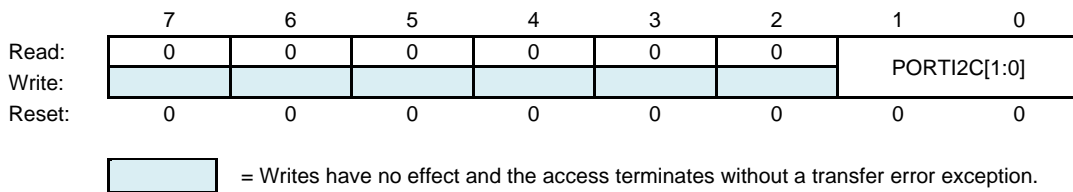
The read/write bits control the data direction of the I2C pins. DDR12C[1] is for GPIO\_SDA pin and DDR12C[0] is for GPIO\_SCL pin. These bits are available only in GPIO mode.

1 = Corresponding pin is configured as an output

0 = Corresponding pin is configured as an input

**27.4.2.10. I2C Port Data Register (I2CPDR)**

**Address: I2C\_BASEADDR+0x0000\_000B**



**Figure 27-11: I2C Port Data Register (I2CPDR)**

Read: Anytime

Write: Anytime

**PORTI2C[1:0]** — I2C Port Data Bits

Writing these bits sets the output data for the corresponding I2C pins configured as GPIO. PORTI2C[1] is for GPIO\_SDA pin and PORTI2C[0] is for GPIO\_SCL pin. Reading these bits returns the I2C pins' level.

**27.5. 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. Also, 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.

**27.5.1. Master Mode**

The I2C module may initial a transfer if the bus is free (the BBUSY bit of I2CS is cleared) 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 the 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.

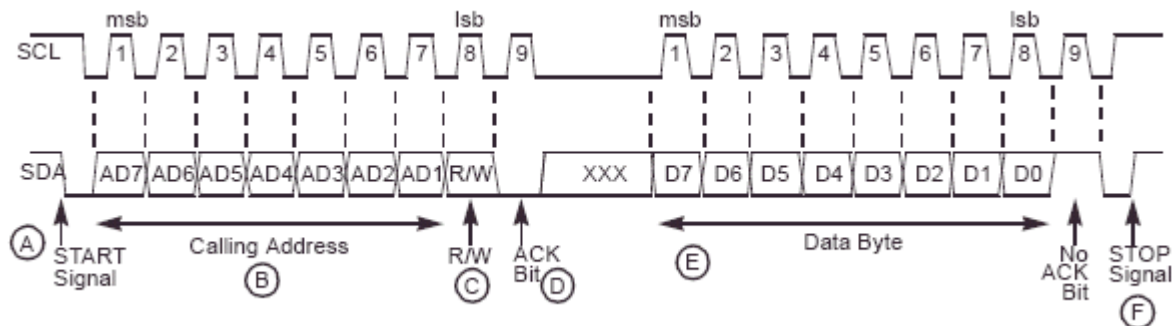
**27.5.2. Slave Mode**

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 cannot work in slave mode and master mode at the same time.

**27.5.3. 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 27-12**, and described in the following text.



**Figure 27-12: 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 27-12**). 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-bits 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.

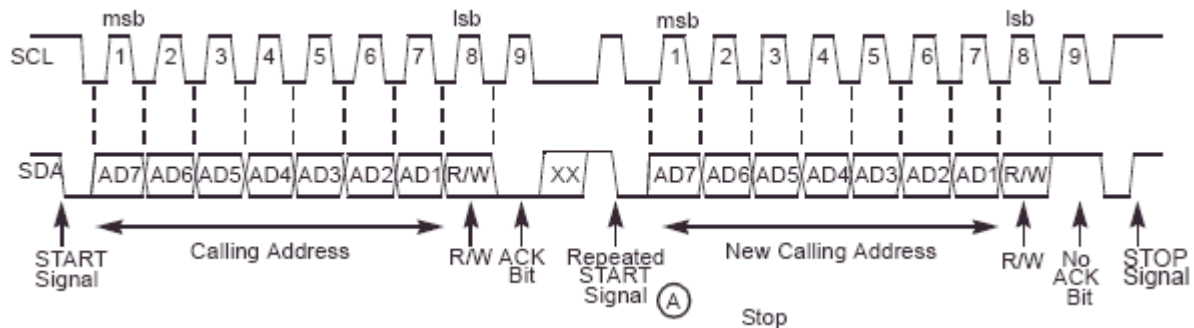
Data can be changed only while SCL is low and must be held stable while SCL is high, as **Figure 27-12** shows. SCL is pulsed once for each data bit, with the MSB being sent first. The receiving device must acknowledge each byte by pulling SDA low at the ninth clock; therefore, a data byte transfer takes 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 27-13**) 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 27-13**). 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 27-13: Repeat Start of I2C Protocol**

**27.5.4. 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.

**27.5.5. 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 27-14**). When all devices involved have counted off their low period, the synchronized clock SCL is released and pulled high.

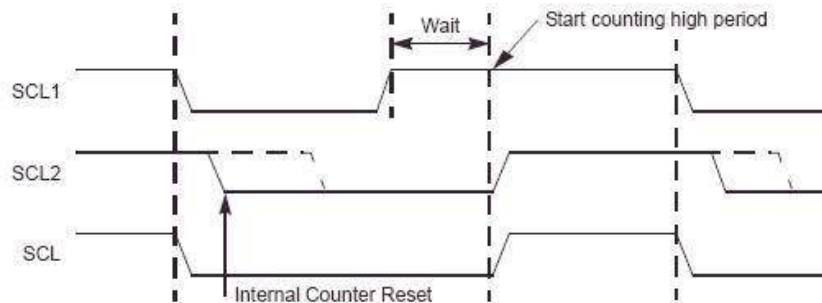
There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

**27.5.6. 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.

**27.5.7. 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 27-14: SCL Synchronization**

**27.5.8. 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. 8-bits master code (00001XXX)
3. not-acknowledge bit (A)

(see **Figure 27-15 Data transfer format in HS mode**) and (see **Figure 27-16 A Complete HS mode transfer**) show this in more detail. The HS master code has two main functions:

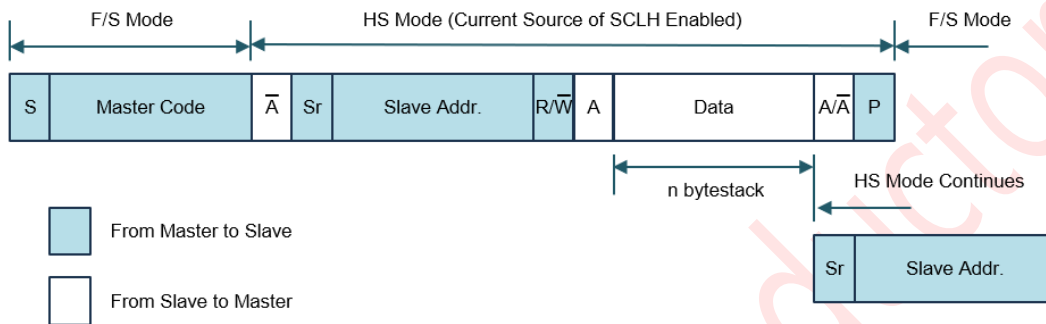
1. It allows arbitration and synchronization between competing masters at F/S-mode speeds, resulting in one winning master.
2. It indicates the beginning of an HS-mode transfer.

HS-mode master codes are reserved 8-bits 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 t<sub>H</sub>, refer to (see **Figure 27-16 A Complete HS mode transfer**) the current-source pull-up circuit for the SCL signal. As other devices can delay the serial transfer before t<sub>H</sub> 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 SCL 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 (S<sub>r</sub>) followed by a 7-bits slave address with a R/W bit address, and receives an acknowledge bit (A) from the selected slave.

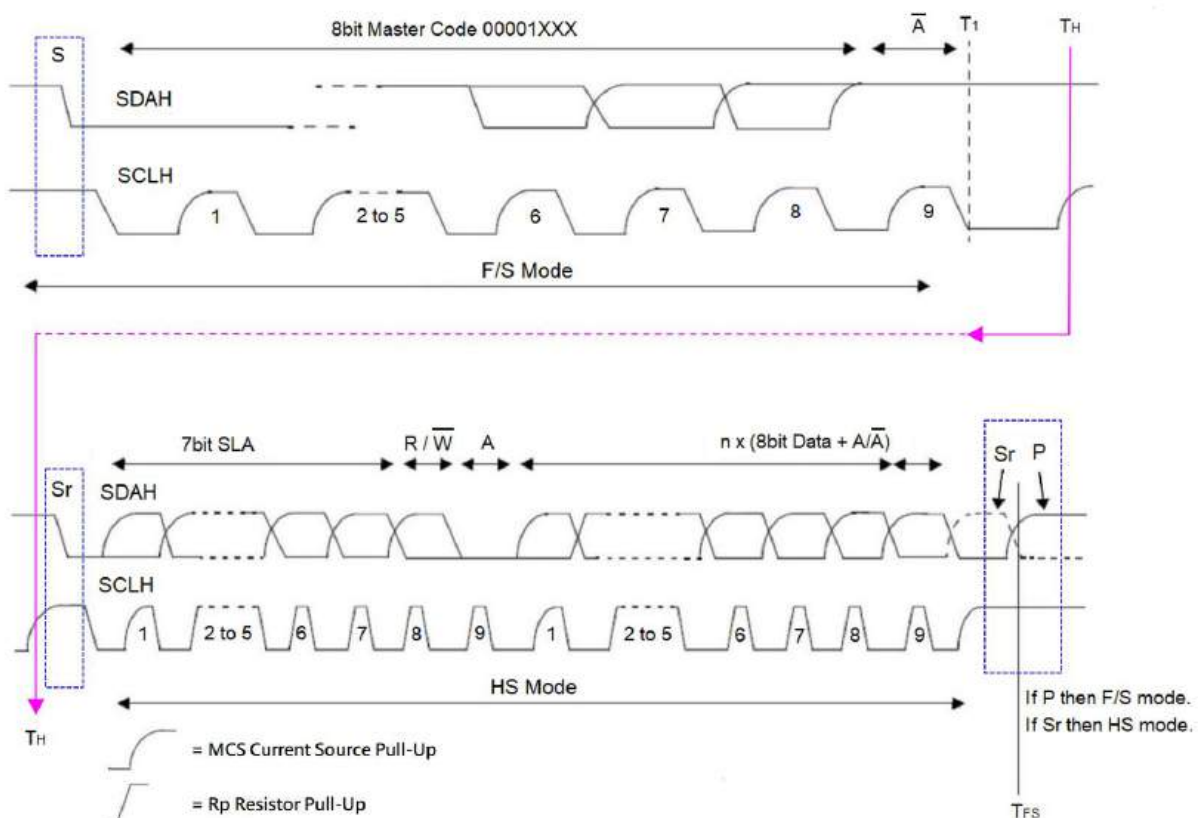


After a repeated START condition and after each acknowledge bit (A) or not-acknowledge bit ( $\bar{A}$ ), the active master disables its current-source pull-up circuit. This enables other devices to delay the serial transfer by stretching the LOW period of the SCL signal. The active master re-enables 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 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 is possible that a master links a number of HS-mode transfers, separated by repeated START conditions (Sr).



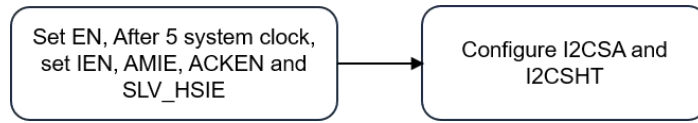
**Figure 27-15: Data Transfer Format in HS Mode**



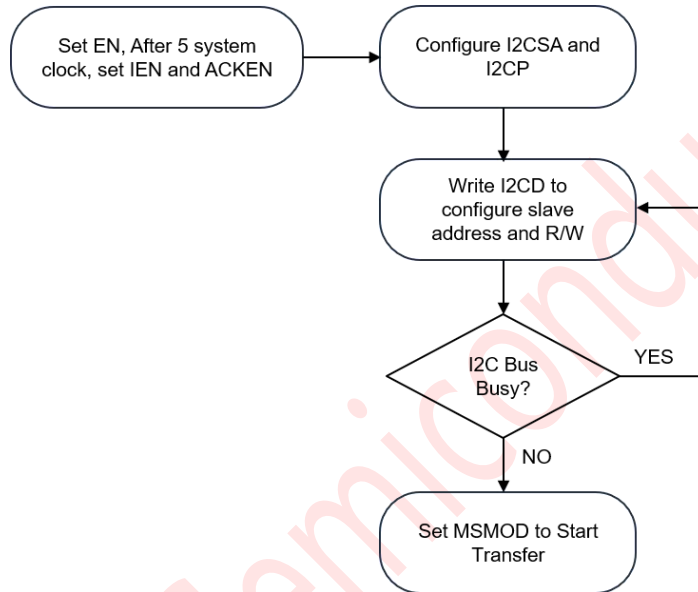
**Figure 27-16: A Complete HS Mode Transfer**

**27.5.9. Software Transaction Flow Diagrams**

1. Initialization.

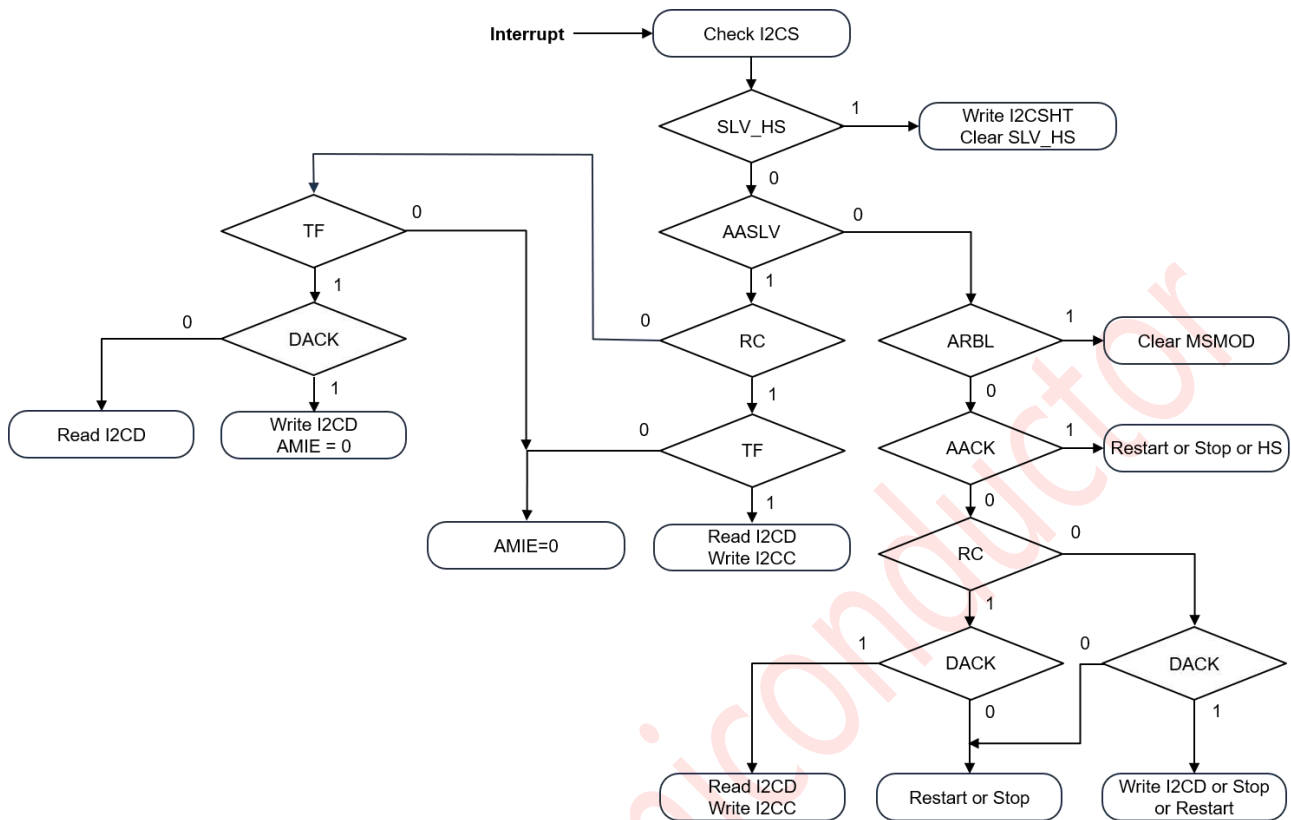


**Figure 27-17: Slave Mode Initialization**



**Figure 27-18: Master Mode Initialization**

2. Interrupt Transaction



**Figure 27-19: 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, it means I2C is in the slave-receiver mode then check TF, if TF has also been set, it 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, it means only slave address has been received, then write I2CC to clear RC and AMIE, and then receive the 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, it means I2C is in the slave-transmit mode, then write I2CD to clear TF, write I2CC to clear AMIE , and transmit the 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, it 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, it 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, it 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 choose 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, it means I2C is in the master-transmit mode, then check DACK, if DACK has been set, write I2CD to clear TF and transmit the 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.

Levetop Semiconductor

## 28. Pulse Width Modulator (PWM)

### 28.1. Introduction

LT168 has 4 PWM-Timers embedded. The 4 PWM-Timers has 2 Pre-scale, 2 clock divider, 4 clock selectors, 4 16-bits counters, 4 16-bits comparators, and 2 Dead-Zone generators. Each can be used as a timer and issue 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 16-bits counter in each timer receives clock signal from clock selector and can be used to handle one PWM period. The 16-bits 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 utilizes PWM clock as the clock source. Once Dead-Zone generator is enabled, the outputs of two PWM-Timers are blocked. Two output pins are all used as Dead-Zone generator output signals 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 is included in PWM 0; and the Capture 1 and PWM 1 share another timer, and etc. Therefore users must setup the PWM-Timer before turning 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 interrupt trigger condition. Capture channel 1&2&3 have the same feature. Whenever Capture issues Interrupt 0/1/2/3, the PWM counter 0/1/2/3 will be reloaded at this moment. The maximal capture frequency should be decided by interrupt process time. If interrupt process time is  $T_0$ , then the capture channel input signal should not change in  $T_0$ , the maximal capture frequency is  $1/T_0$ .

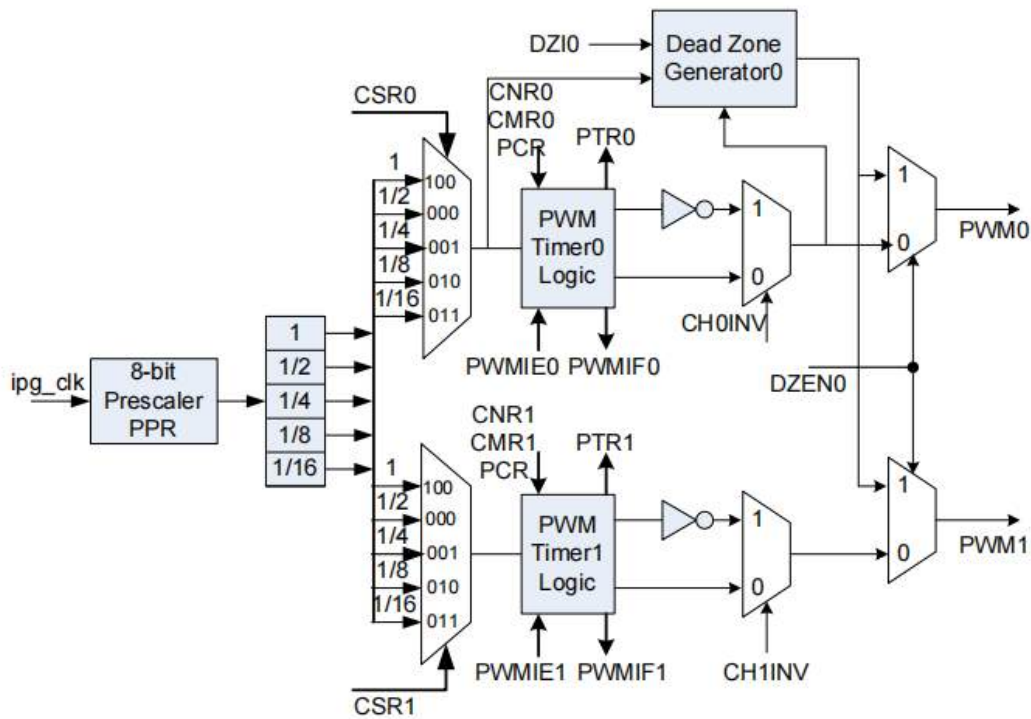
There are only four interrupts from PWM to interrupt controller (INTC). PWM 0 and Capture 0 share the same interrupt; PWM1 and Capture 1 share the same interrupt and so on. Therefore, PWM function and Capture function in the same channel cannot be used at the same time.

### 28.2. Features

The Pulse Width Modulator includes below distinctive features:

- Programmable Period
- Programmable Duty Cycle
- Two Dead-Zone Generator
- Capture Function
- Pins can be configured as General-Purpose I/O

**28.3. Block Diagram**



**Figure 28-1: PWM Block Diagram**

**28.4. Signal Description**

**Table 28-1: 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.

## 28.5. Memory Map and Registers

The PWM Module memory map is shown in **Table 28-2**. The PWM0 base address is **0x400D\_0000**, and PWM1 is **0x400E\_0000**. This subsection describes the memory map and register structure of PWM.

### 28.5.1. Memory Map

**Table 28-2: Module Memory Map**

Address	Bit[31:0]	Access <sup>(1)</sup>
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

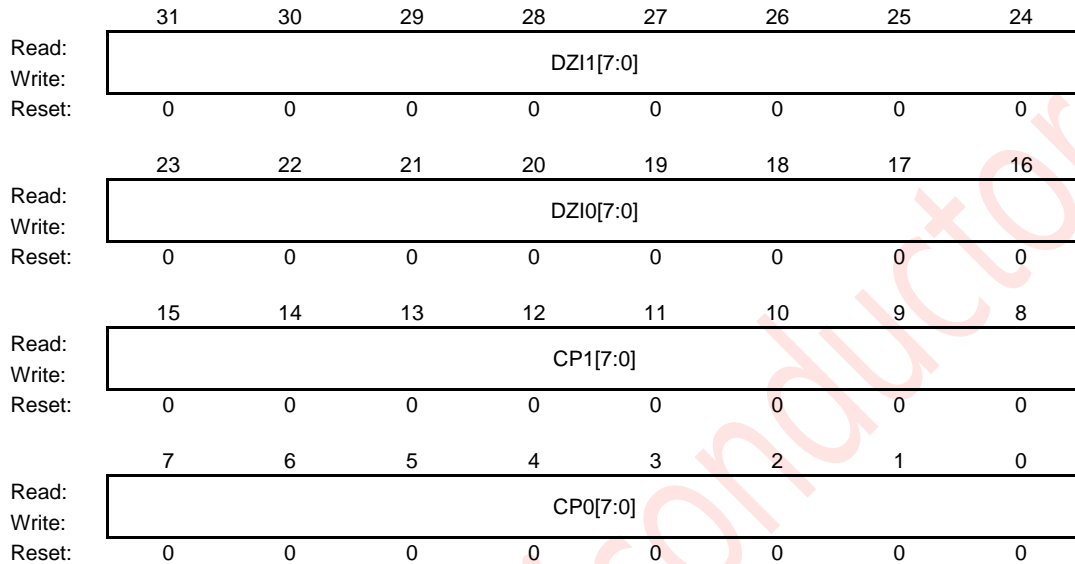
**Note (1)** : S/U = CPU supervisor or user mode access.

**28.5.2. Register Descriptions**

**28.5.2.1. PWM Pre-scale Register (PPR)**

The register(*PPR*) is used to set prescaler and set dead zone length.

**Address: PWMn\_BASEADDR+0x0000\_0000**



**Figure 28-2: PWM Pre-scale Register (PPR)**

**DZI1[7:0]** — Dead zone interval register 1 for PWM2 and PWM3  
 These 8-bits 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 8-bits 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.



28.5.2.2. 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 8-bits pre-scale.

Address: PWMn\_BASEADDR+0x0000\_0004

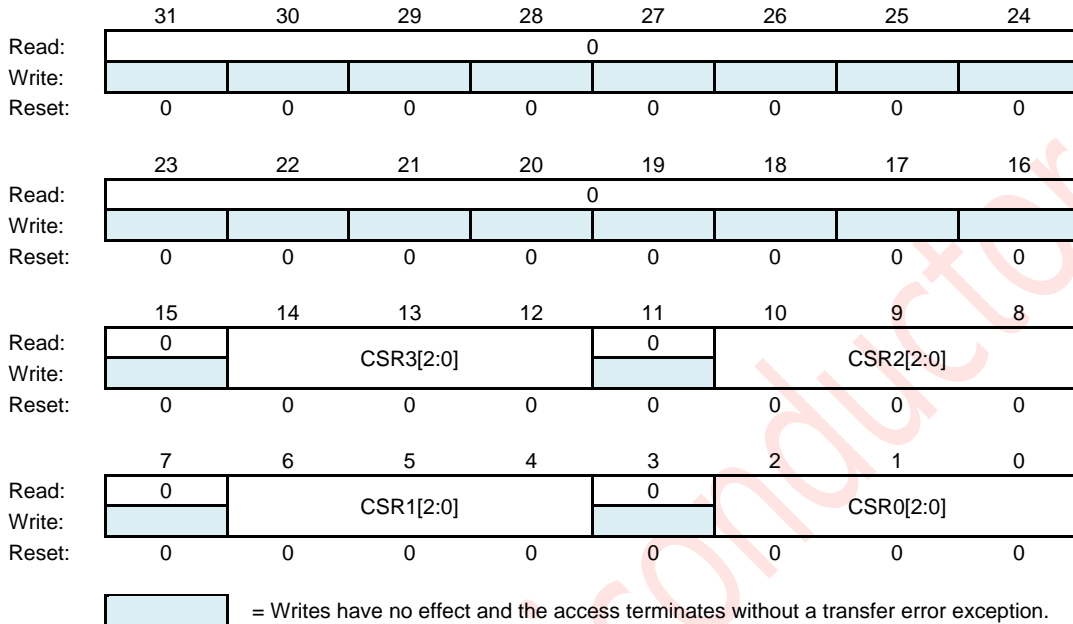


Figure 28-3: PWM Clock Select Register(PCSR)

**CSR3[2:0]** — Timer 3 Clock Source Selection

Select clock input for timer 3.

Table 28-3: Timer 3 Clock Source Selection

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.

28.5.2.3. PWM Control Register (PCR)

This register is the PWM control register.

Address: PWMn\_BASEADDR+0x0000\_0008

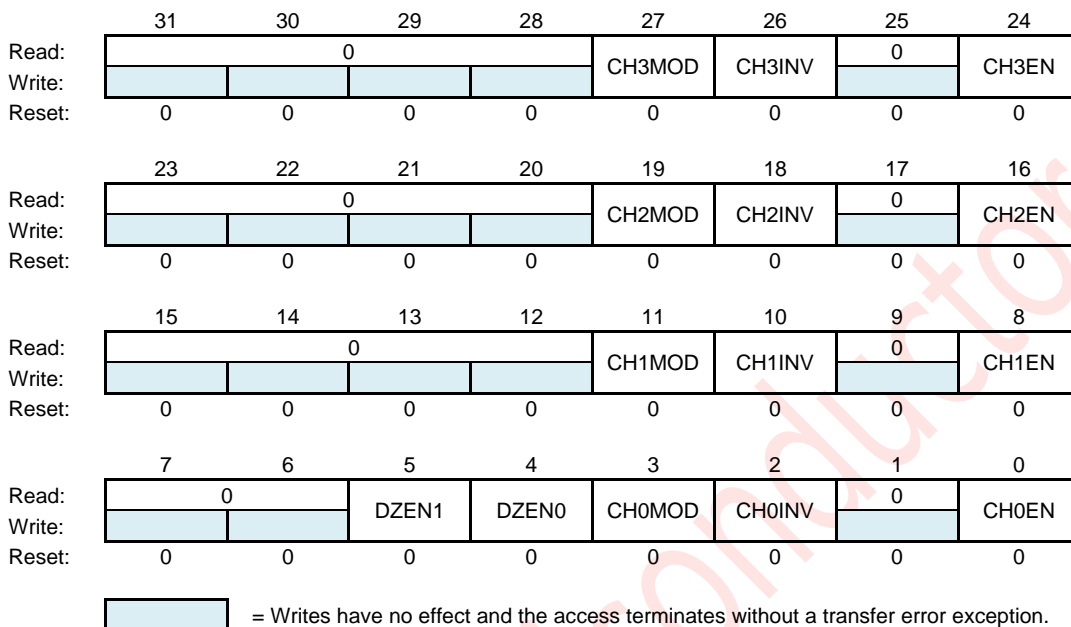


Figure 28-4: PWM Control Register (PCR)

**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 to be cleared.

**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 to be cleared.

**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 to be cleared.

**CH1INV** — Timer 1 Inverter ON/OFF

1 = Inverter ON  
0 = Inverter OFF

**CH1EN** — Timer 1 Enable/Disable

1 = Enable  
0 = Disable

**DZEN1** — Dead-Zone 1 Generator Enable/Disable

1 = Enable  
0 = Disable

**Note:** When DZEN1 is enabled, CH3EN should be disabled. Because channel3 and channel2 outputs are both decided by channel2.

**DZEN0** — Dead-Zone 0 Generator Enable/Disable

1 = Enable  
0 = Disable

**Note:** When DZEN0 is enabled, CH1EN should be disabled. Because channel1 and channel0 outputs are both decided by channel0.

**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 to be cleared.

**CH0INV** — Timer 0 Inverter ON/OFF

1 = Inverter ON  
0 = Inverter OFF

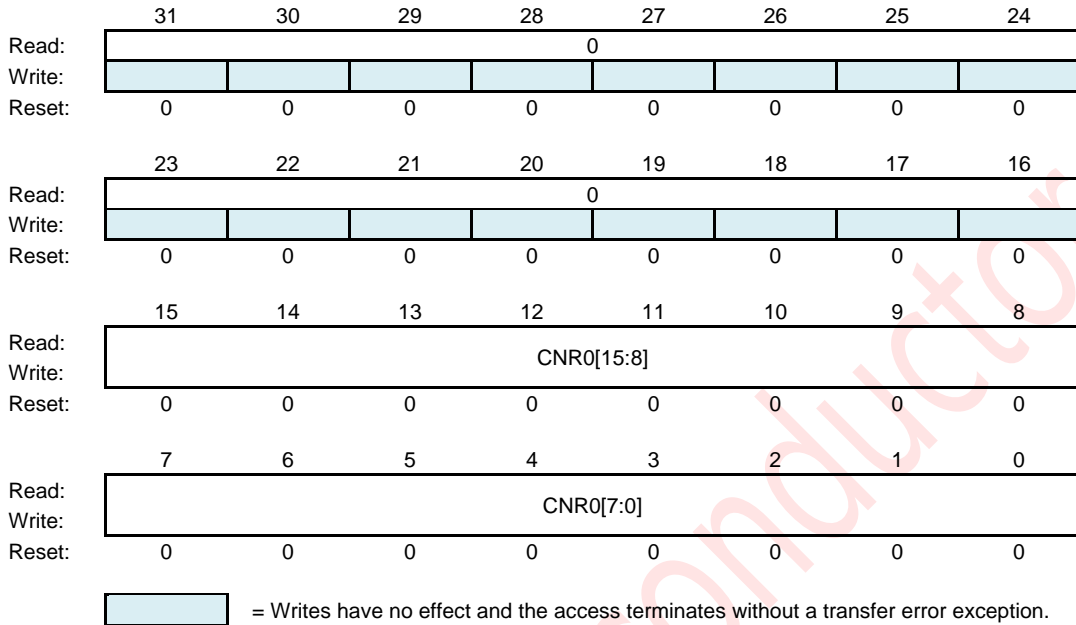
**CH0EN** — Timer 0 Enable/Disable

1 = Enable  
0 = Disable

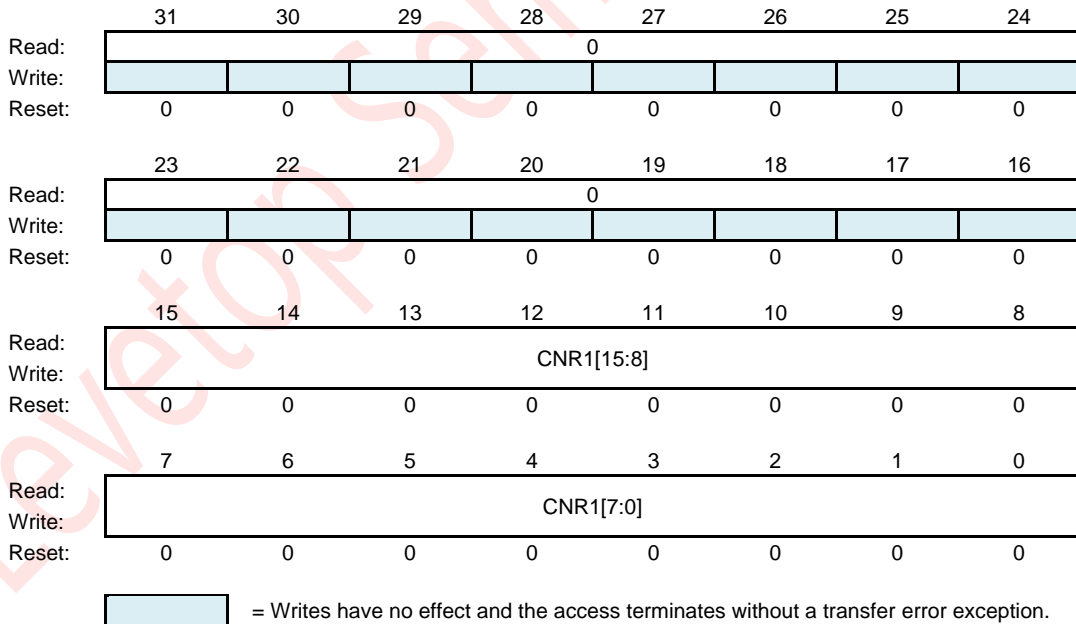
**28.5.2.4. PWM Counter Register (PCNRn)**

These registers control the period of the PWM by defining the number of the count\_pulse in the period.

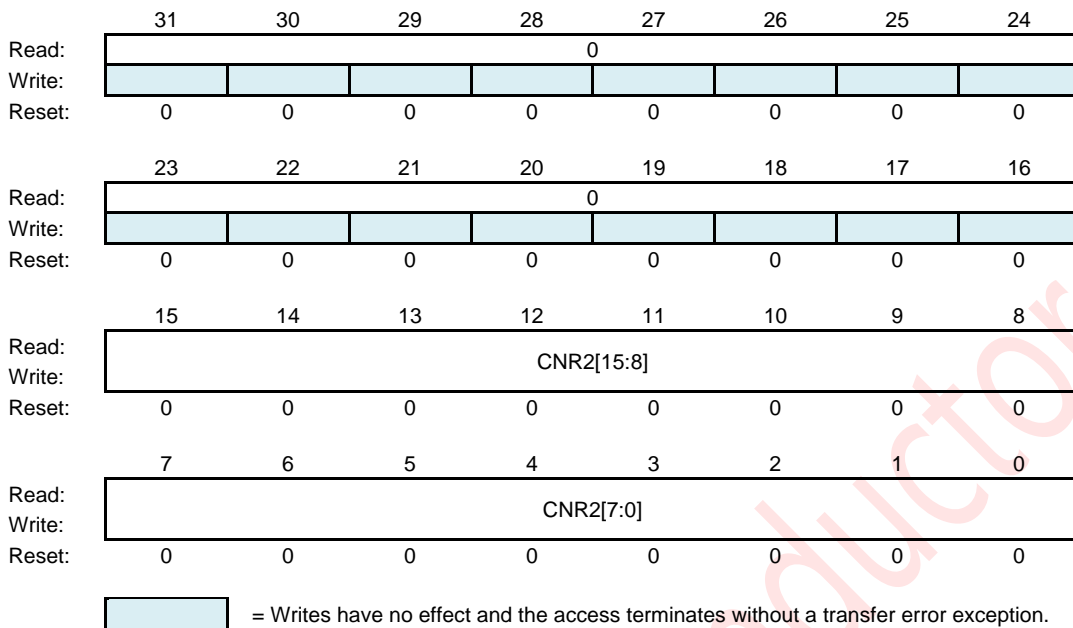
**Address: PWMn\_BASEADDR+0x0000\_000C**



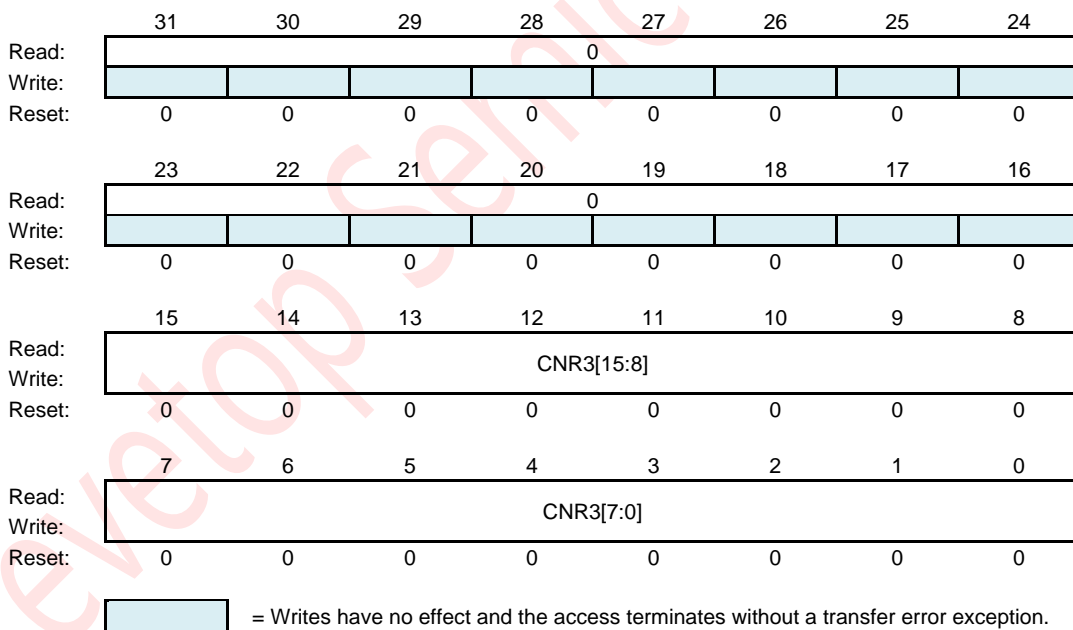
**Address: PWMn\_BASEADDR+0x0000\_0018**



**Address: PWMn\_BASEADDR+0x0000\_0024**



**Address: PWMn\_BASEADDR+0x0000\_0030**



**Figure 28-5: PWM Counter Register (PCNR)**

**CNRx[15:0]** — Loaded Value for PWM Counter/Timer

Data range: 65535~0 (Unit: 1 PWM clock cycle)

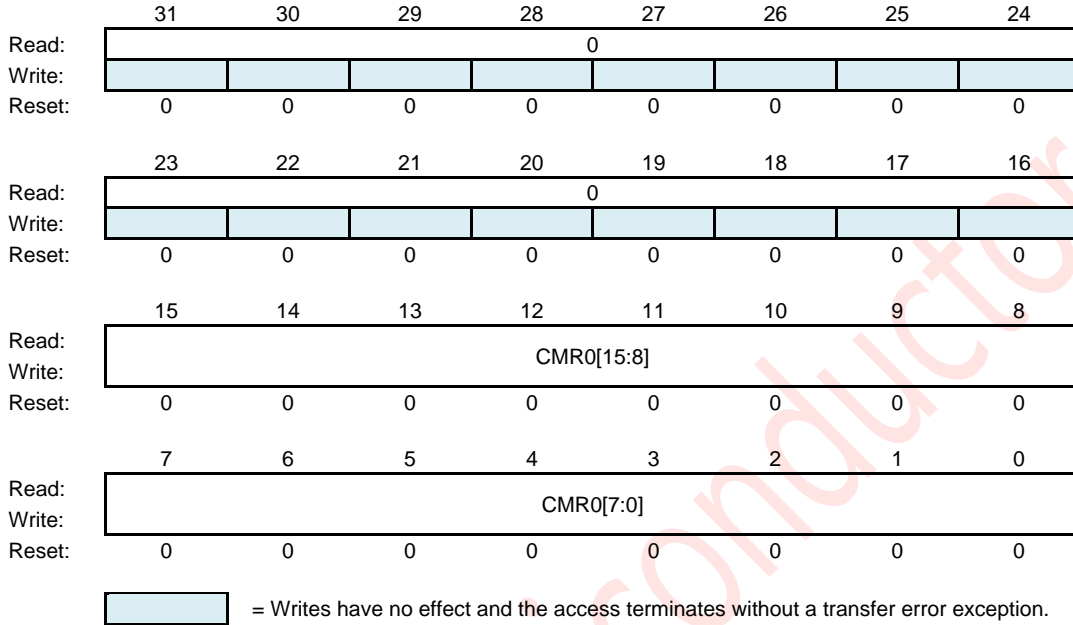
**Note:**

- 1: One PWM cycle width = CNR + 1. If CNR equal zero, PWM counter/timer will be stopped.
- 2: Whenever a value is written to CNR, it will take effect in the next PWM Counter cycle.

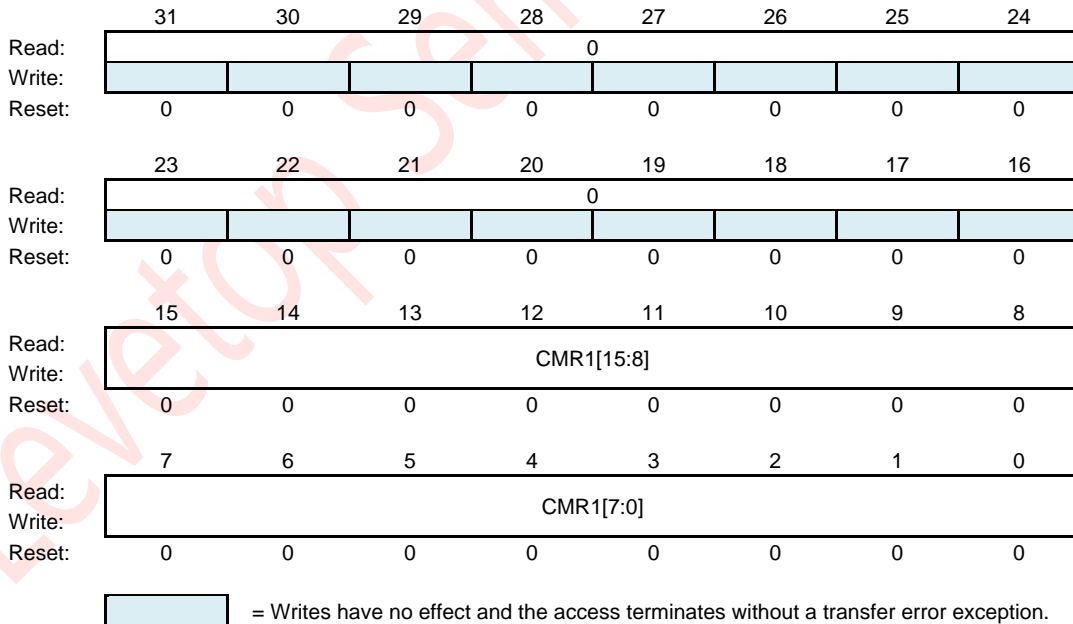
**28.5.2.5. PWM Comparator Register (PCMRn)**

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.

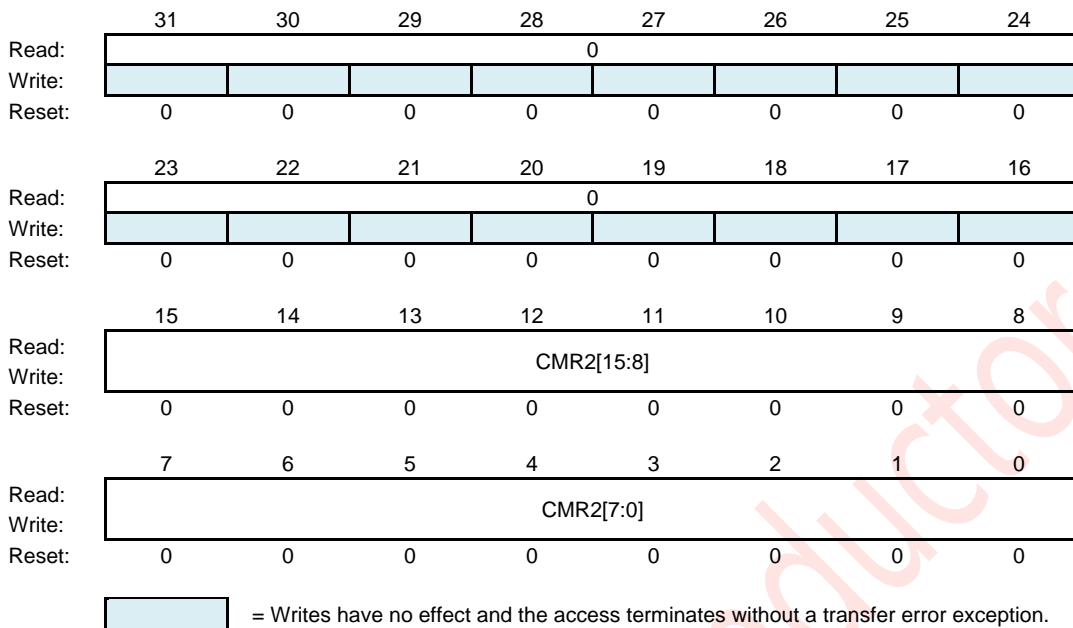
**Address: PWMn\_BASEADDR+0x0000\_0010**



**Address: PWMn\_BASEADDR+0x0000\_001C**



Address: PWMn\_BASEADDR+0x0000\_0028



Address: PWMn\_BASEADDR+0x0000\_0034

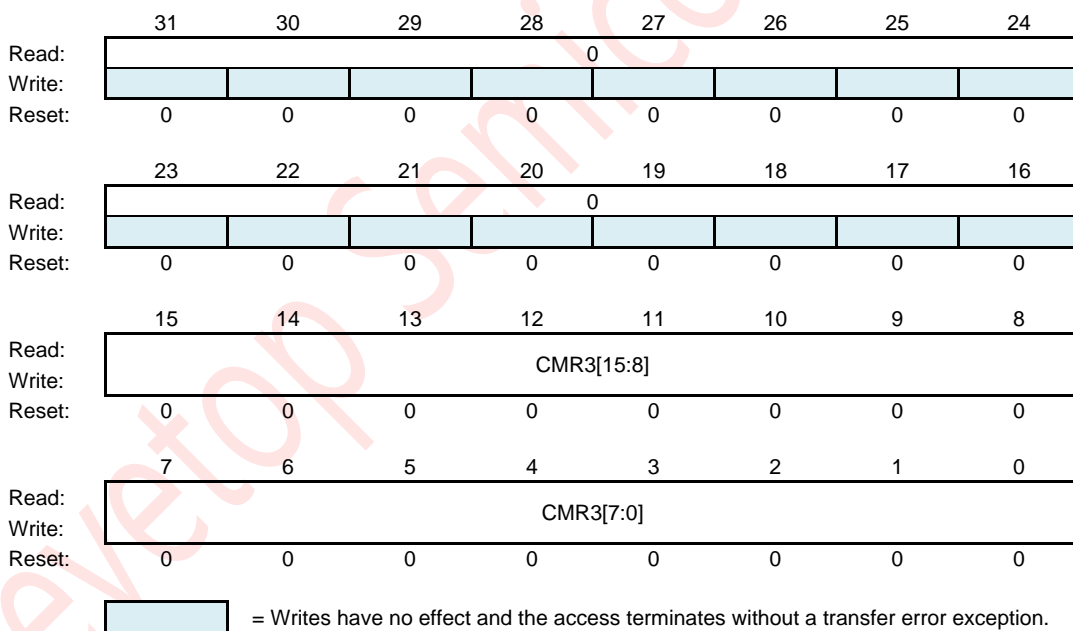


Figure 28-6: PWM Comparator Register (PCMR)

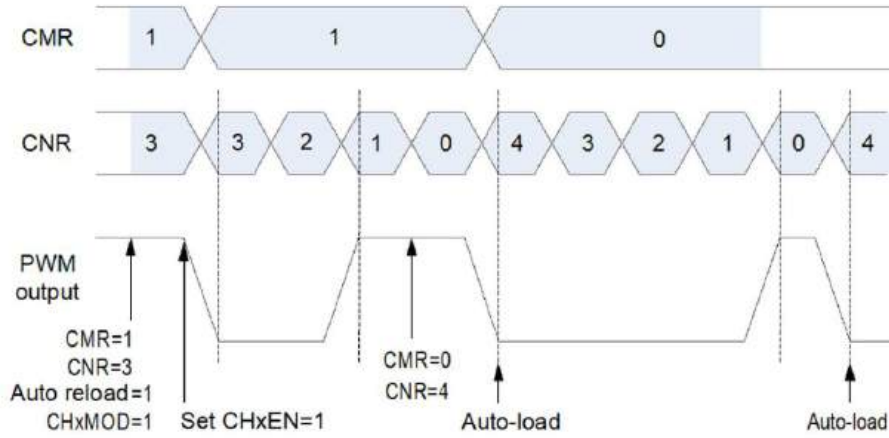
**CMRx[15:0]** — PWM Comparator Register  
 Data range: 65535~0 (Unit: 1 PWM clock cycle)

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

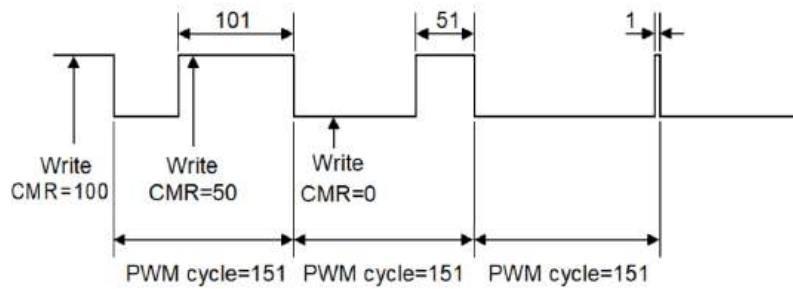
**Note:**

- 1: PWM duty = CMR + 1. If CMR equals zero, PWM duty = 1
- 2: Whenever a value is written to CMR, it will take effect in the next PWM Counter cycle.



**Figure 28-7: PWM Output**

Modulate PWM Controller output duty ratio (CNR = 150)



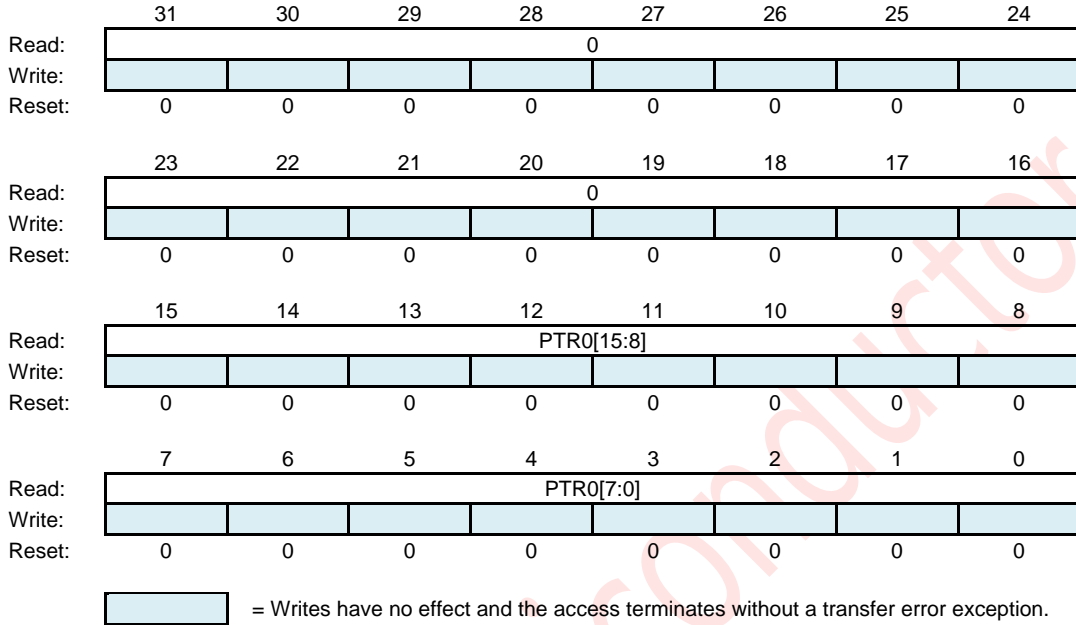
**Figure 28-8: PWM Duty Cycle**



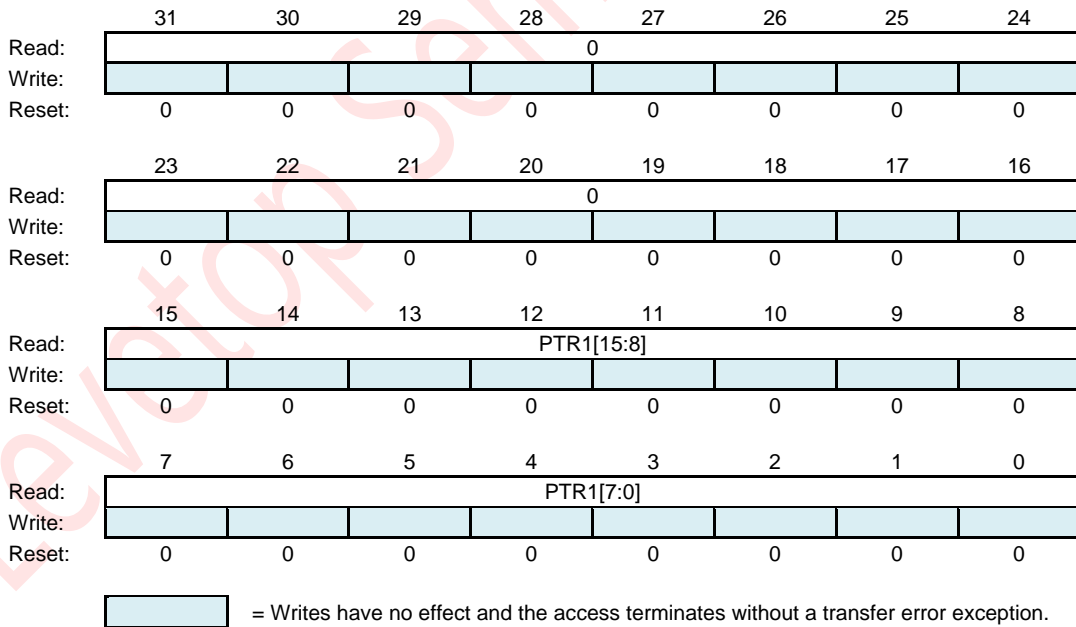
**28.5.2.6. PWM Timer Register (PTRn)**

The read-only PWM timer registers hold the current count value. It can be read at any time without disturbing the counter.

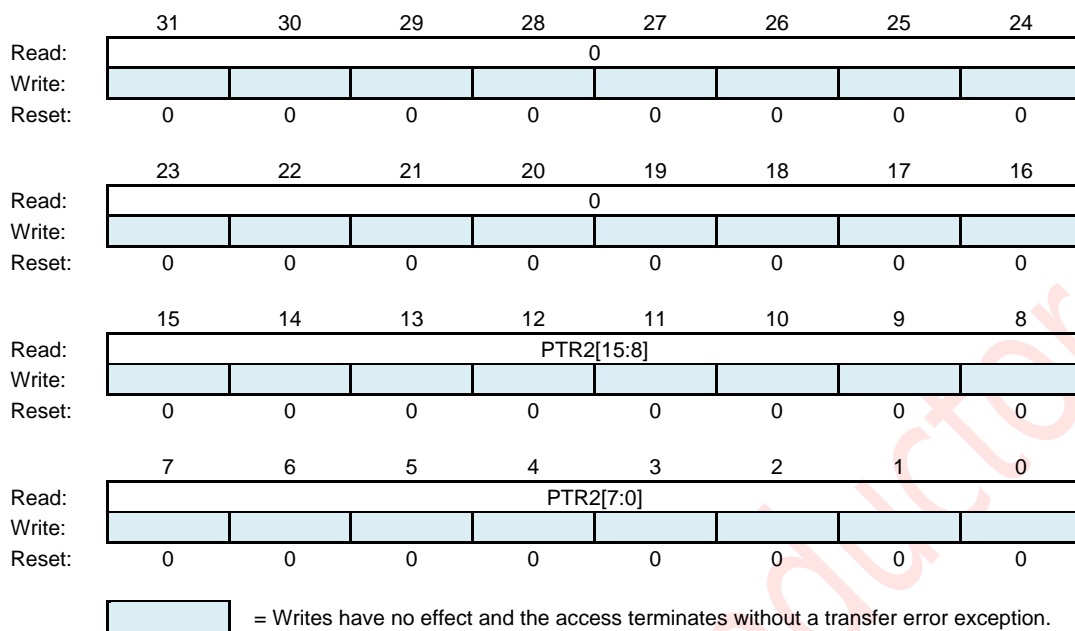
**Address: PWMn\_BASEADDR+0x0000\_0014**



**Address: PWMn\_BASEADDR+0x0000\_0020**



Address: PWMn\_BASEADDR+0x0000\_002C



Address: PWMn\_BASEADDR+0x0000\_0038

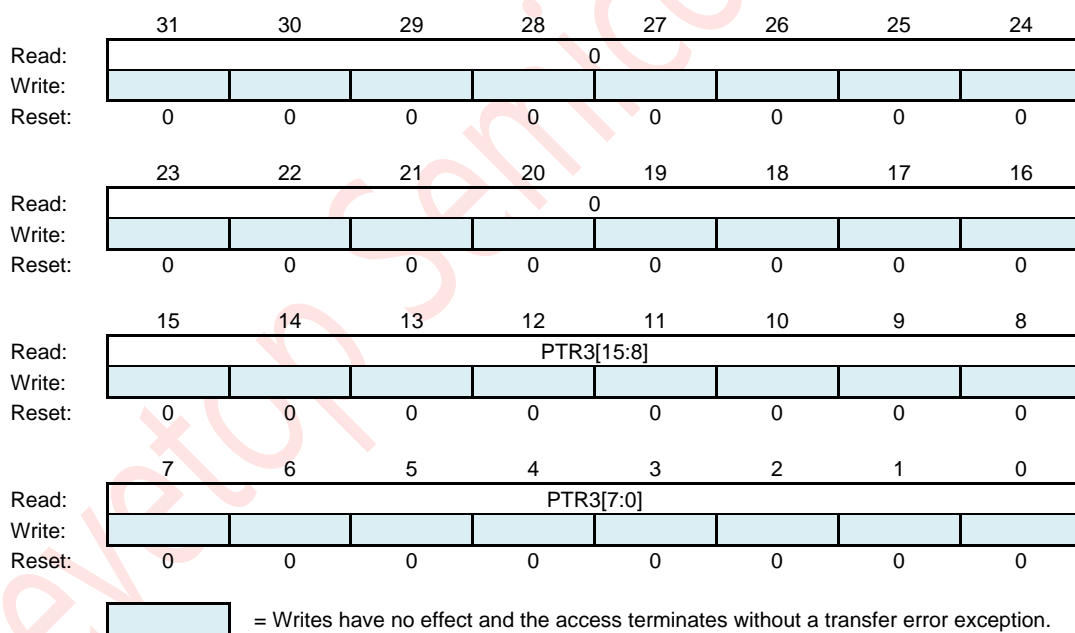


Figure 28-9: PWM Timer Register (PTR)

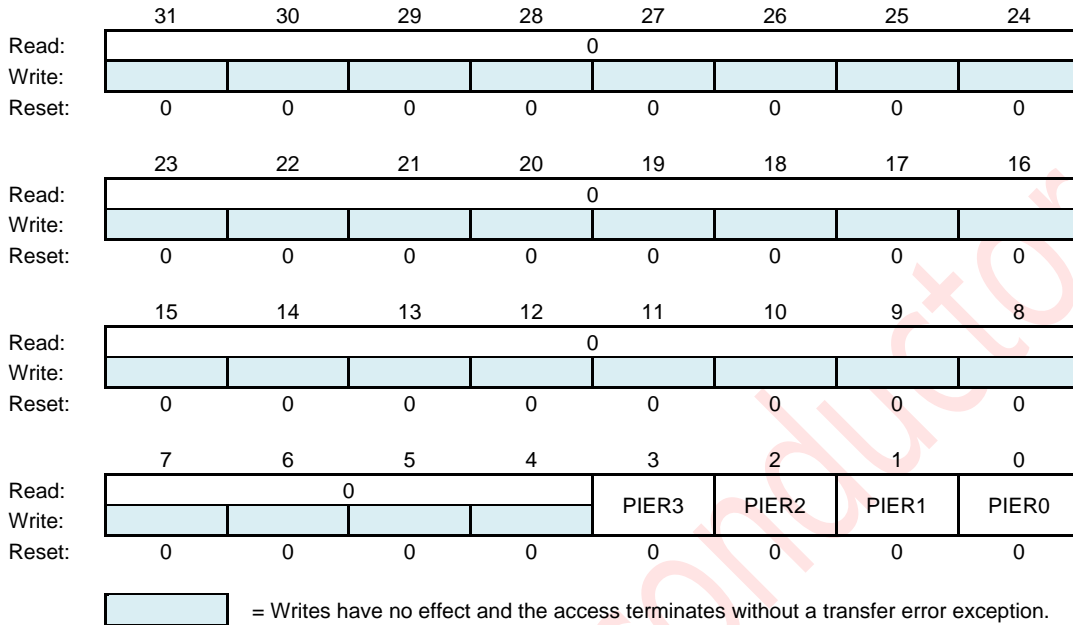
**PTRx[15:0]** — PWM Timer value

The read-only PTR bits holds the current count value. Users can monitor PTR to obtain the current value in the 16-bits down counter.

**28.5.2.7. PWM Interrupt Enable Register (PIER)**

This register is used to enable PWM timer interrupt.

**Address: PWMn\_BASEADDR+0x0000\_003C**



**Figure 28-10: 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

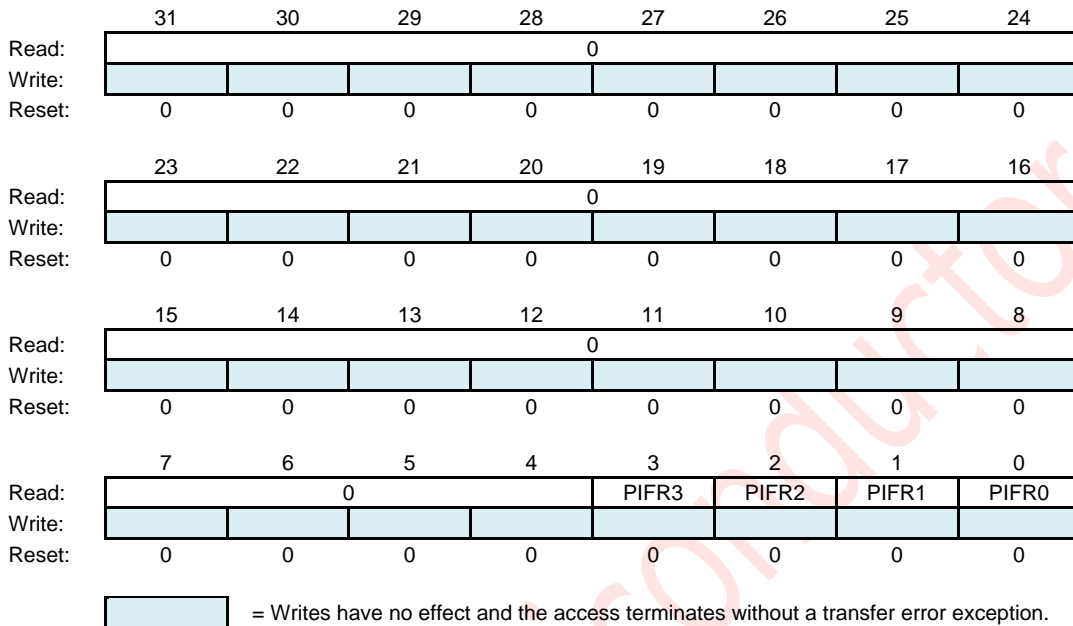
**PIER0** — PWM Timer 0 Interrupt Enable

- 1 = Enable
- 0 = Disable

**28.5.2.8. PWM Interrupt Flag Register (PIFR)**

This register is used to indicate PWM timer interrupt flag.

**Address: PWMn\_BASEADDR+0x0000\_0040**



**Figure 28-11: PWM Interrupt Flag Register (PIFR)**

**PIFR3** — PWM Timer 3 Interrupt Flag.

When PWM timer3 counts to 0, and PIER3 = 1, PIFR3 will be set to 1. Writing 1 to this bit will clear PIFR3.

- 1 = Interrupt Flag on
- 0 = Interrupt Flag off

**PIFR2** — PWM Timer 2 Interrupt Flag.

When PWM timer2 counts to 0, and PIER2 = 1, PIFR2 will be set to 1. Writing 1 to this bit will clear PIFR2.

- 1 = Interrupt Flag on
- 0 = Interrupt Flag off

**PIFR1** — PWM Timer 1 Interrupt Flag.

When PWM timer1 counts to 0, and PIER1 = 1, PIFR1 will be set to 1. Writing 1 to this bit will clear PIFR1.

- 1 = Interrupt Flag on
- 0 = Interrupt Flag off

**PIFR0** — PWM Timer 0 Interrupt Flag.

When PWM timer0 counts to 0, and PIER0 = 1, PIFR0 will be set to 1. Writing 1 to this bit will clear PIFR0.


- 1 = Interrupt Flag on
- 0 = Interrupt Flag off

**28.5.2.9. PWM Capture Control Register (PCCR0/1)**

These registers are used to control the capture function.


**Address: PWMn\_BASEADDR+0x0000\_0044**

Read:	0							
Write:								
Reset:	0	0	0	0	0	0	0	0
Read:	CFLRD1	CRLRD1	0	CAPIF1	CAPCH1 EN	FL_IE1	RL_IE1	INV1
Write:								
Reset:	0	0	0	0	0	0	0	0
Read:	0							
Write:								
Reset:	0	0	0	0	0	0	0	0
Read:	CFLRD0	CRLRD0	0	CAPIF0	CAPCH0 EN	FL_IE0	RL_IE0	INV0
Write:								
Reset:	0	0	0	0	0	0	0	0

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

**Address: PWMn\_BASEADDR+0x0000\_0048**

Read:	0							
Write:								
Reset:	0	0	0	0	0	0	0	0
Read:	CFLRD3	CRLRD3	0	CAPIF3	CAPCH3 EN	FL_IE3	RL_IE3	INV3
Write:								
Reset:	0	0	0	0	0	0	0	0
Read:	0							
Write:								
Reset:	0	0	0	0	0	0	0	0
Read:	CFLRD2	CRLRD2	0	CAPIF2	CAPCH2 EN	FL_IE2	RL_IE2	INV2
Write:								
Reset:	0	0	0	0	0	0	0	0

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

**Figure 28-12: 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 does not have a falling transition.

write 1 to 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 does not have a rising transition.
- write 1 to clear this bit.

**CAPIFx** — Capture Channel x interrupt flag

- 1 = When input channel x has a falling transition, and FL\_IEx bit is enabled, this interrupt flag will be set. When input channel x has a rising transition, and RL\_IEx bit is enabled, this interrupt flag will also be set.
  - 0 = Capture channel x interrupt flag is not set.
- write 1 to clear this bit.

**CAPCHxEN** — Capture Channel x Enable/Disable

- 1 = Enable
- 0 = Disable

When this bit is set to 1, Capture latched the PMW-counter and the value is saved to CRLR (Rising latch) and CFLR (Falling latch). When this bit is set to 0, Capture does not update CRLR and CFLR, and disables Channel x Interrupt.

**FL\_IEx** — Channel x Falling Interrupt Enable ON/OFF

- 1 = Enable
- 0 = Disable

When this bit is set to 1, if Capture detects Channel x has a falling transition, Capture issues an Interrupt.

**RL\_IEx** — Channel x Rising Interrupt Enable ON/OFF

- 1 = Enable
- 0 = Disable

When this bit is set to 1, if Capture detects Channel x has a rising transition, Capture issues an Interrupt.

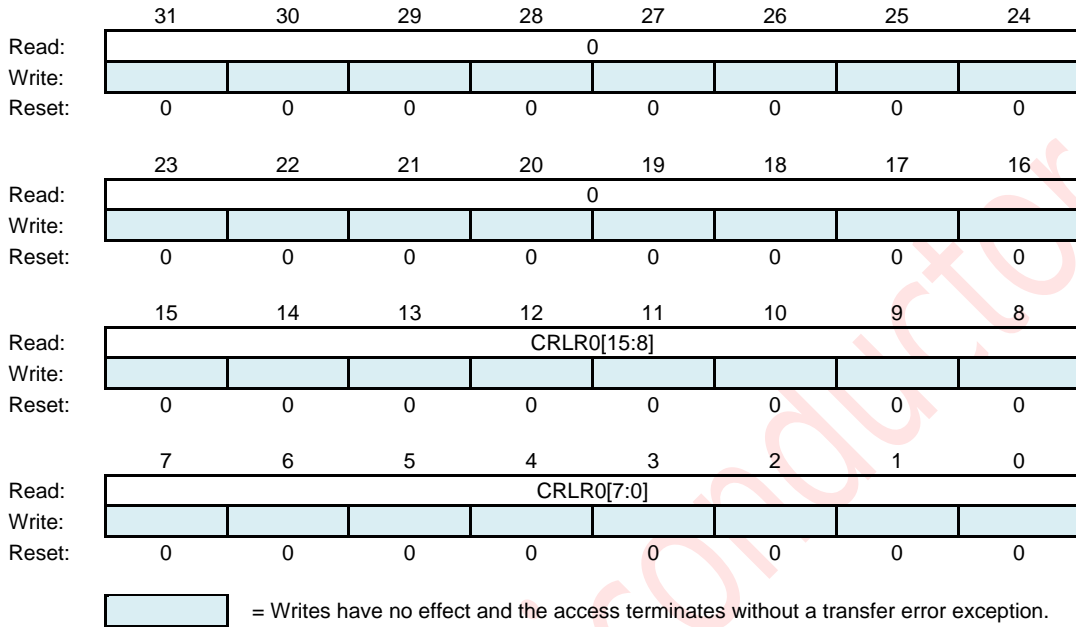
**INVx** — Channel x Inverter ON/OFF

- 1 = Inverter ON
- 0 = Inverter OFF

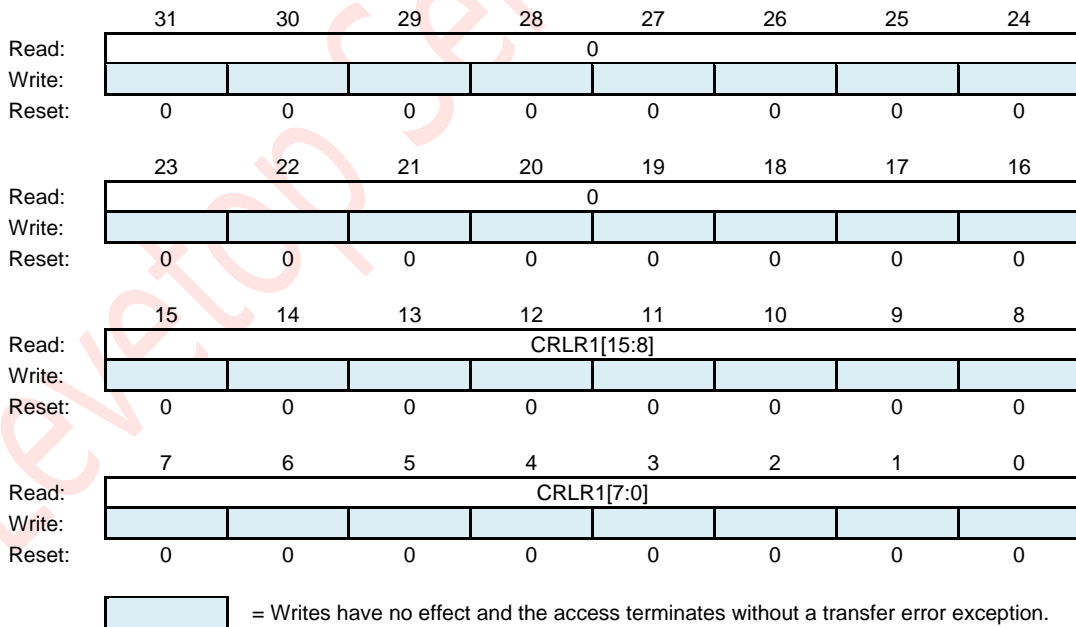
**28.5.2.10. PWM Capture Rising Latch Register (PCRLRn)**

These registers are used to latch the PWM counter when capture rising transition.

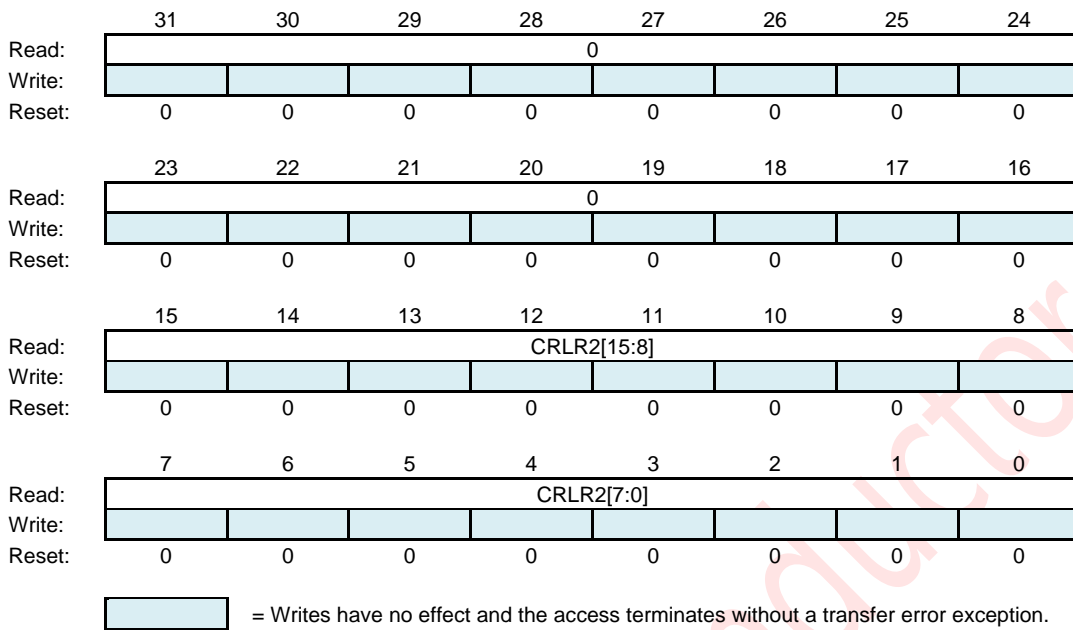
**Address: PWMn\_BASEADDR+0x0000\_004C**



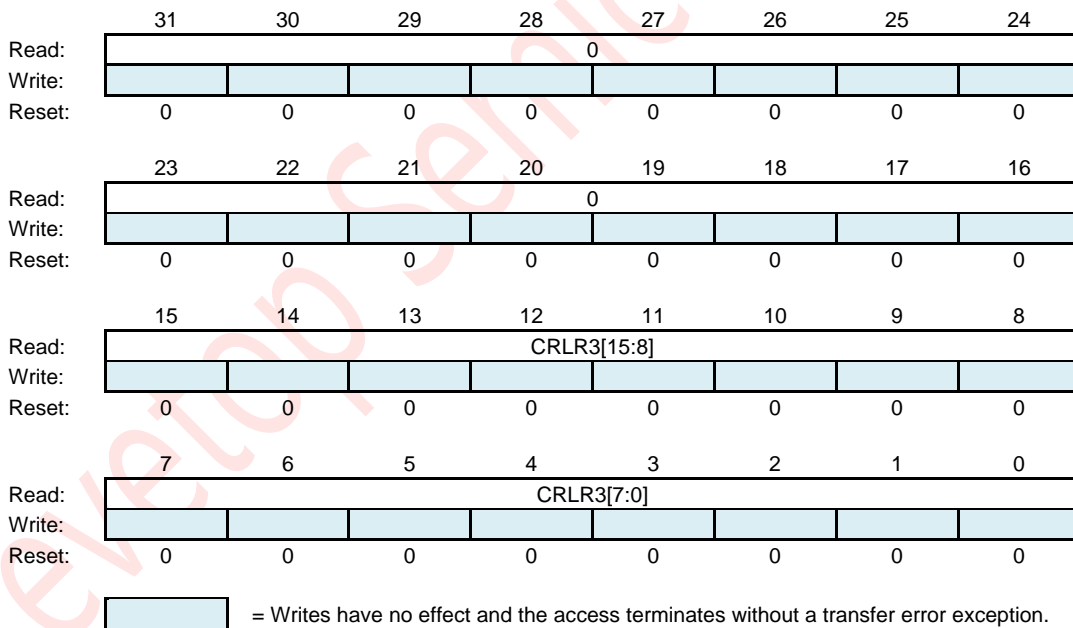
**Address: PWMn\_BASEADDR+0x0000\_0054**



**Address: PWMn\_BASEADDR+0x0000\_005C**



**Address: PWMn\_BASEADDR+0x0000\_0064**



**Figure 28-13: PWM Capture Rising Latch Register (PCRLR0/1/2/3)**

**CRLRx[15:0]** — Capture Rising Latch Registerx

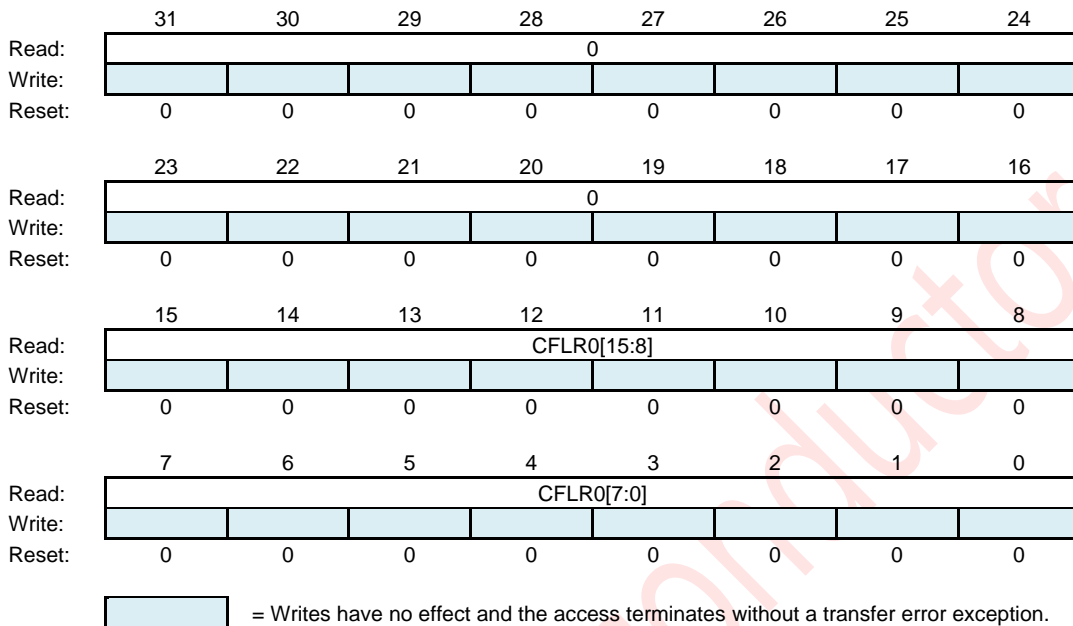
Latch the PWM counter when Channel x has a rising transition.



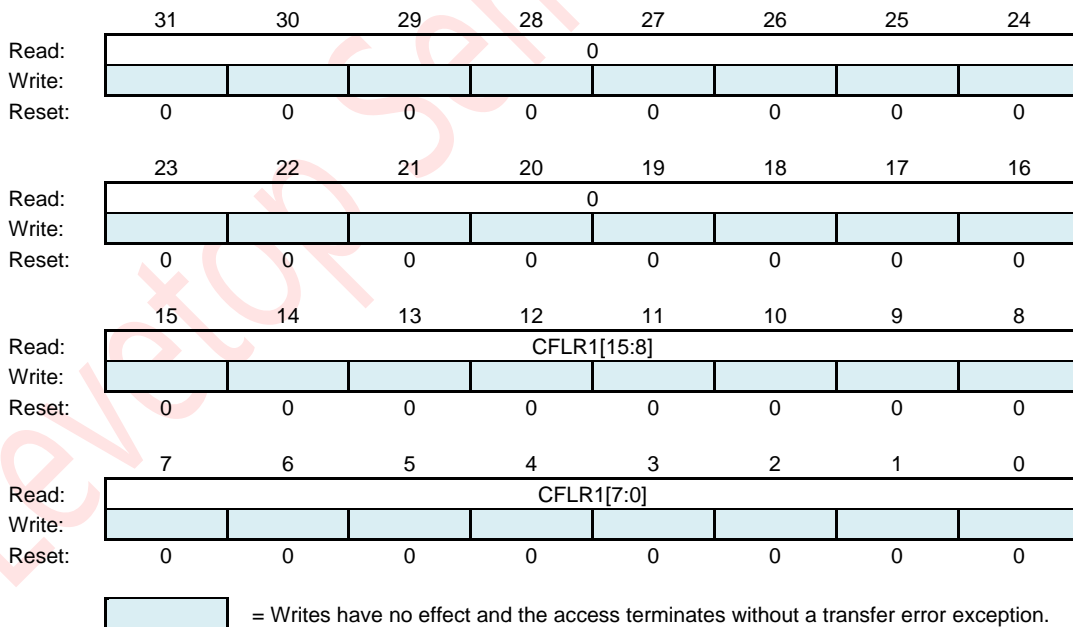
**28.5.2.11. PWM Capture Falling Latch Register (PCFLRn)**

These registers are used to latch the PWM counter when a falling transition is captured.

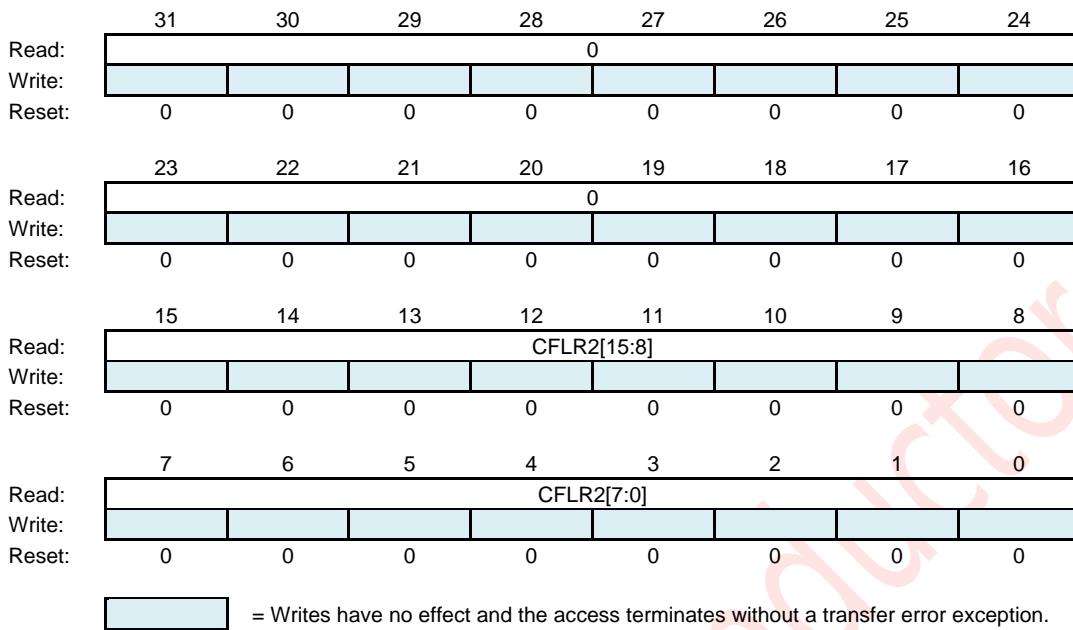
**Address: PWMn\_BASEADDR+0x0000\_0050**



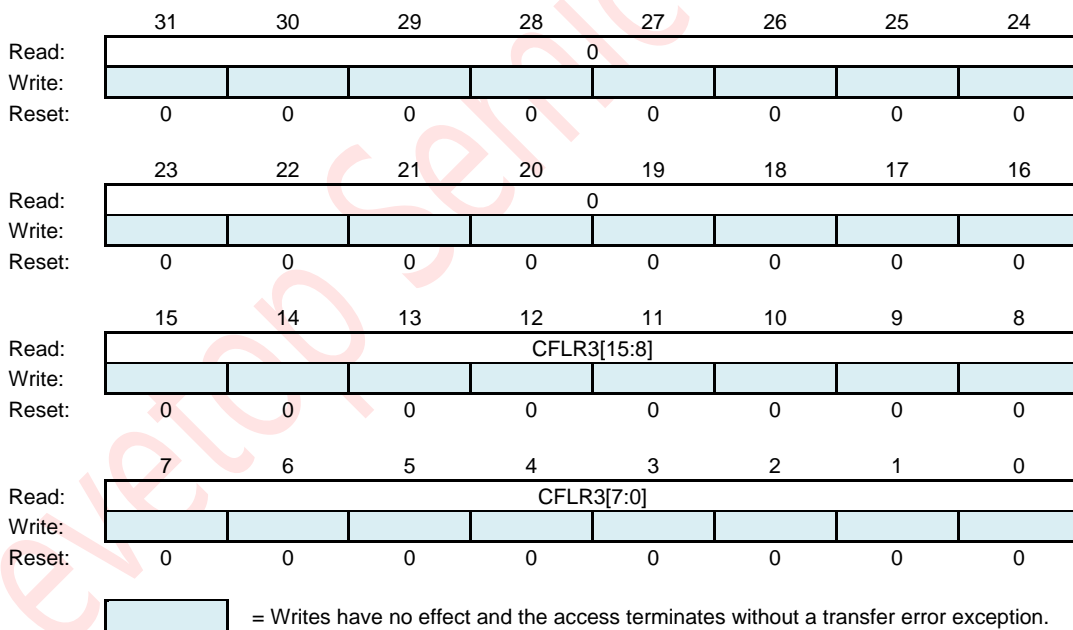
**Address: PWMn\_BASEADDR+0x0000\_0058**



**Address: PWMn\_BASEADDR+0x0000\_0060**



**Address: PWMn\_BASEADDR+0x0000\_0068**



**Figure 28-14: PWM Capture Falling Latch Register (PCFLR0/1/2/3)**

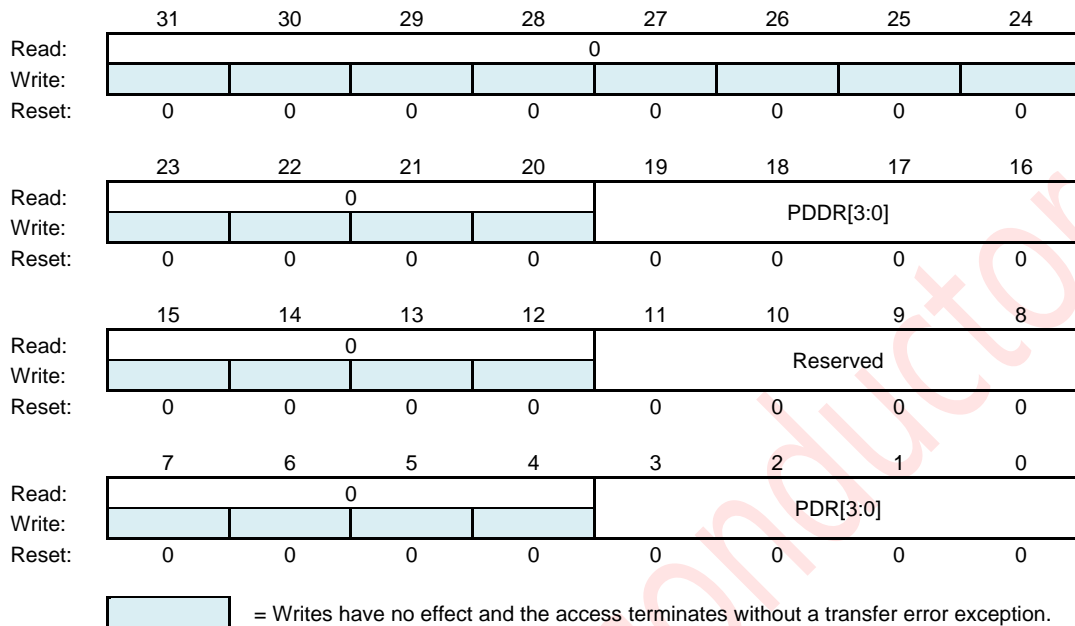
**CFLRx[15:0]** — Capture Falling Latch Registerx

Latch the PWM counter when Channel x has a falling transition.

**28.5.2.12. PWM Port Control Register (PPCR)**

The register(PPCR) is used to control PWMx pin direction and pin status.

**Address: PWMn\_BASEADDR+0x0000\_006C**



**Figure 28-15: PWM Port Control Register (PPCR)**

**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 is configured as output
- 0 = Corresponding pin is configured as input

**PDR[3:0]** — Port Data Register

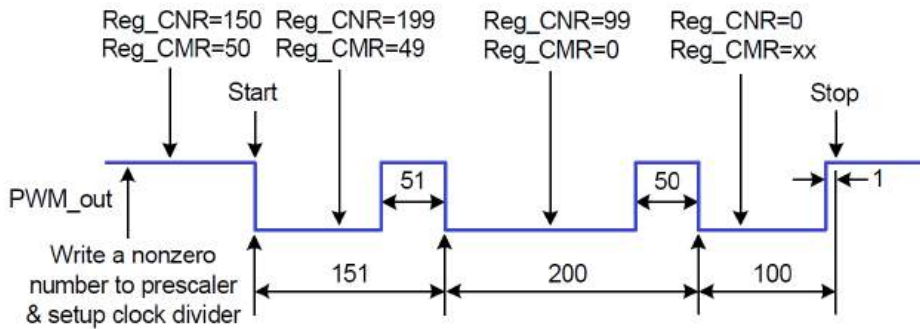
Writing data to PDR[3:0] can drive corresponding pins that are configured as general-purpose outputs. Reading an input (PDDR bit clear) returns the pin level.

## 28.6. Functional Descriptions

This subsection describes the PWM functional operation.

### 28.6.1. PWM Double Buffering And Automatic Reload

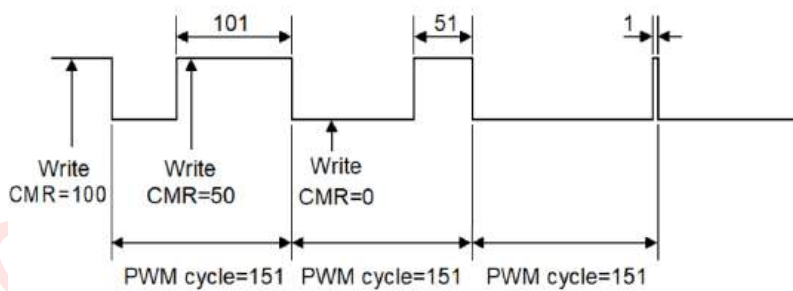
PWM-Timers have a double buffering function, enabling the reload value changed for next timer operation without stopping current timer operation. Although a new timer value is set, the current timer operation still operates successfully. The counter value can be written into CNR0~3 and the current counter value can be read from PTR0~3. The auto-reload operation will copy the data from CNR0~3 to the down-counter when the down-counter reaches zero. If CNR0~3 are set as zero, the counter will be halt when the counter count to zero. If auto-reload bit is set as zero, the counter will be stopped immediately.



**Figure 28-16: PWM Double Buffering Illustration**

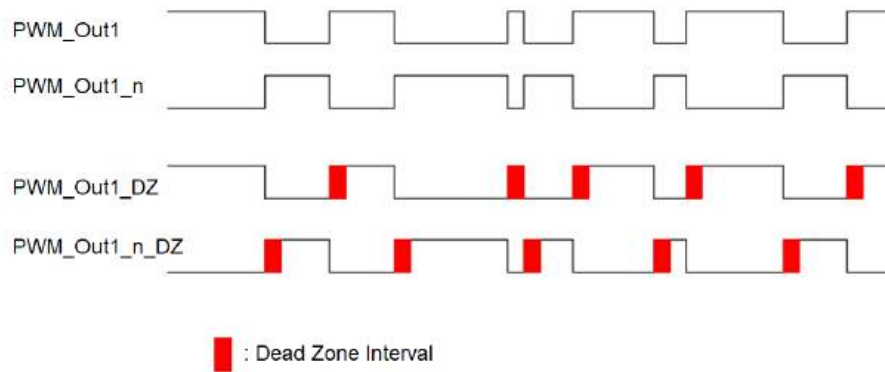
### 28.6.2. Modulate Duty Ratio

The double buffering function allows CMR to be written at any point in the current cycle. The loaded value will take effect from the next cycle.



**Figure 28-17: PWM Controller Output Duty Ratio**

**28.6.3. Dead-Zone Generator**



**Figure 28-18: Dead Zone Generation Operation**

**28.6.4. 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)

**28.6.5. PWM Timer Stop Procedure**

Method 1:

Set 16-bits down counter (CNR) as 0, and monitor PTR. When PTR reaches to 0, disable PWM timer (PCR). (Recommended)

Method 2:

Set 16-bits down counter (CNR) as 0. When an interrupt request happens, disable PWM timer (PCR). (Recommended)

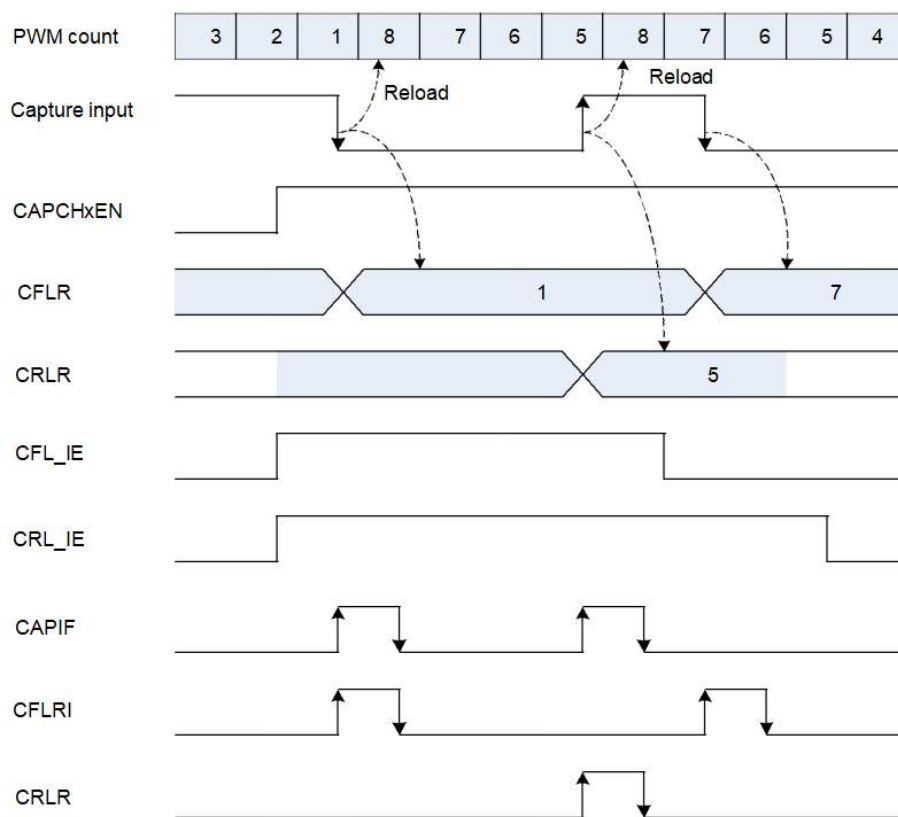
Method 3:

Disable PWM timer directly (PCR). (Not recommended)

**28.6.6. 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)

**28.6.7. Capture Basic Timer Operation**



**Figure 28-19: Capture Basic Timer Operation**

At this case, the CNR is 8:

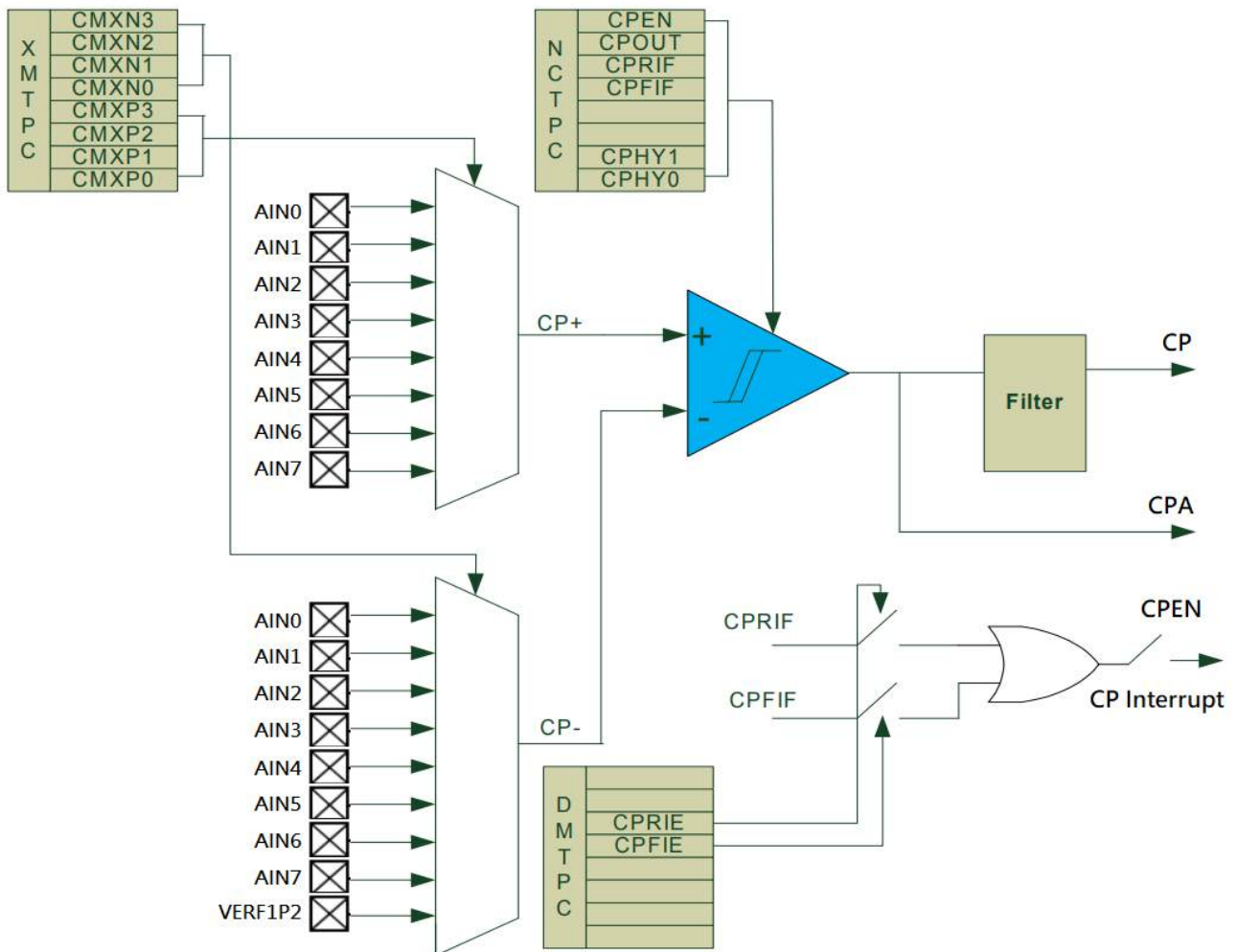
1. When CAPIF<sub>x</sub> is set to 1, the PWM counter CNR<sub>x</sub> will be reload.
2. The channel low pulse width is (CNR + 1 – CRLR).
3. The channel high pulse width is (CNR + 1 - CFLR).

## 29. Analog Comparator (COMP)

### 29.1. Introduction

The Analog 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 the system clock is not active. This allows the Comparator to operate and generate an output with the device in STOP mode.

### 29.2. Block Diagram



**Figure 29-1: Comparator Block Diagram**

### **29.3. Modes of Operation**

This subsection describes the three low-power modes.

#### **29.3.1. Wait Mode**

In wait mode, the Comparator module can continue to operate normally by setting CPEN bit and can be configured to exit the low-power mode by generating an interrupt request.

#### **29.3.2. Doze Mode**

In doze mode, the Comparator module can continue to operate normally by setting CPEN bit and can be configured to exit the low-power mode by generating an interrupt request.

#### **29.3.3. Stop Mode**

In stop mode, the system clock is absent, and the Comparator module can continue 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.



## 29.4. Memory Map and Registers

The Comparator Module memory map is shown in **Table 29-1**. The COMP0 base address is 0x400A\_0000, and COMP1 is 0x400B\_0000. This subsection describes the memory map and register structure for Comparator.

### 29.4.1. Memory Map

Refer to **Table 29-1** for a description of the memory map. This device has two Comparator modules.

**Table 29-1: Comparator Module Memory Map**

Offset Address	Bits[7:0]	Access <sup>(1)</sup>
0x3	Comparator Control Register(CPTCN)	S/U
0x2	Comparator0 Mode Selection Register(CPTMD)	S/U
0x1	Comparator MUX Selection Register(CPTMX)	S/U
0x0	Comparator Output Filter Selection Register(CPTFS)	S/U

**Note (1)** : S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. Accessing supervisor only address in user mode has no effect and result in a cycle termination transfer error.

### 29.4.2. Register Descriptions


The Comparator programming model consists of below registers:

- CPTCN: Comparator Control Register.
- CPTMD: Comparator Mode Selection Register.
- CPTMX: Comparator MUX Selection Register.
- CPTFLS: Comparator Output Filter Selection Register

#### 29.4.2.1. Comparator Control Register (CPTCN)

**Address: COMP0\_BASEADDR+0x0000\_0003, COMP1\_BASEADDR+0x0000\_0003**

	7	6	5	4	3	2	1	0
Read:	CPEN	CPOUT	CPRIF	CPFIF	0	0	CPHY[1:0]	
Write:								
Reset:	0	0	0	0	0	0	0	1

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

**Figure 29-2: 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 is enabled
- 0 = Comparator is 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 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 writing one to this bit.

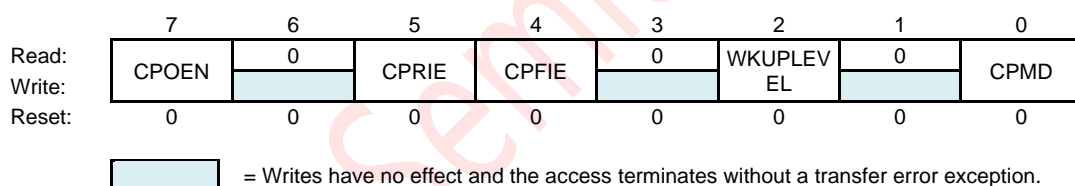
- 1 = Comparator Falling-Edge has occurred.
- 0 = No Comparator Falling-Edge has occurred since this flag was last cleared.

**CPHY[1:0]** — Comparator Hysteresis Control Bits.

- 00 = Hysteresis is disabled.
- 01 = Hysteresis = 8 mV.
- 10 = Hysteresis = 12 mV.
- 11 = Hysteresis = 15.4 mV.

**29.4.2.2. Comparator Mode Selection Register (CPTMD)**

**Address: COMP0\_BASEADDR+0x0000\_0002, COMP1\_BASEADDR+0x0000\_0002**



**Figure 29-3: Comparator Mode Selection Register(CPTMD)**

**CPOEN** — Comparator Output to pad control bit.

- 1 = Comparator output can be observed in PWM1[0] for COMP0 and PWM1[1] for COMP1.
- 0 = Comparator output is disabled.

**CPRIE** — Comparator Rising-Edge Interrupt Enable.

- 1 = Comparator Rising-edge interrupt is enabled.
- 0 = Comparator Rising-edge interrupt is disabled.

**CPFIE** — Comparator Falling-Edge Interrupt Enable.

- 1 = Comparator Falling-edge interrupt is enabled.
- 0 = Comparator Falling-edge interrupt is disabled.

**WKUPLEVEL** — Comparator WakeUp level control bit.

- 1 = Voltage on CP+ > CP– will generate an wakeup request during stop mode.
- 0 = Voltage on CP+ < CP– will generate an wakeup request during stop mode.

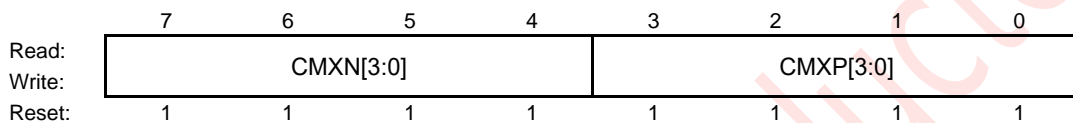
**CPMD** — Comparator Mode Select. These bits select the response time for Comparator.

**Table 29-2: Comparator Mode Selection**

Mode	CPMD Response Time Power Consumption		
Low-speed	0	500ns	3.3uA
High-speed	1	80ns	22uA

**29.4.2.3. Comparator MUX Selection Register (CPTMX)**

**Address: COMP0\_BASEADDR+0x0000\_0001, COMP1\_BASEADDR+0x0000\_0001**



**Figure 29-4: 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 29-3: Comparator Negative Input MUX Selection**

CMXN3	CMXN2	CMXN1	CMXN0	Negative Input
0	0	0	0	AIN0
0	0	0	1	AIN 1
0	0	1	0	AIN 2
0	0	1	1	AIN 3
0	1	0	0	AIN 4
0	1	0	1	AIN 5
0	1	1	0	AIN 6
0	1	1	1	AIN 7
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 29-4: Comparator Positive Input MUX Selection**

CMXP3	CMXP2	CMXP1	CMXP0	Positive Input
0	0	0	0	AIN0
0	0	0	1	AIN 1
0	0	1	0	AIN 2

CMXP3	CMXP2	CMXP1	CMXP0	Positive Input
0	0	1	1	AIN 3
0	1	0	0	AIN 4
0	1	0	1	AIN 5
0	1	1	0	AIN 6
0	1	1	1	AIN 7
1	x	x	x	None

29.4.2.4. Comparator Output Filter Selection Register (CPTFLS)

Address: COMP0\_BASEADDR+0x0000\_0000, COMP1\_BASEADDR+0x0000\_0000

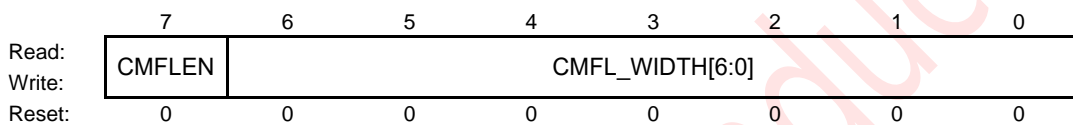


Figure 29-5: Comparator Output Filter Selection Register(CPTFLS)

**CMFLEN** — Comparator Output Digital Filter Enable

- 1 = Comparator Output digital filter is enabled, and CPMRIF and CPMFIF are generated by filter output;
- 0 = Comparator Output digital filter is disabled, and CPMRIF and CPMFIF are generated by raw output;

**CMFL\_WIDTH[6:0]** — Comparator Output Digital Filter Pulse Width Selection.

CMFL\_WIDTH[6:0] determine the width of the input pulse that will be filtered. If the input pulse width is less than (CMFL\_WIDTH[6:0]+2) \* Period of fips, the pulse will be filtered.

29.5. 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.

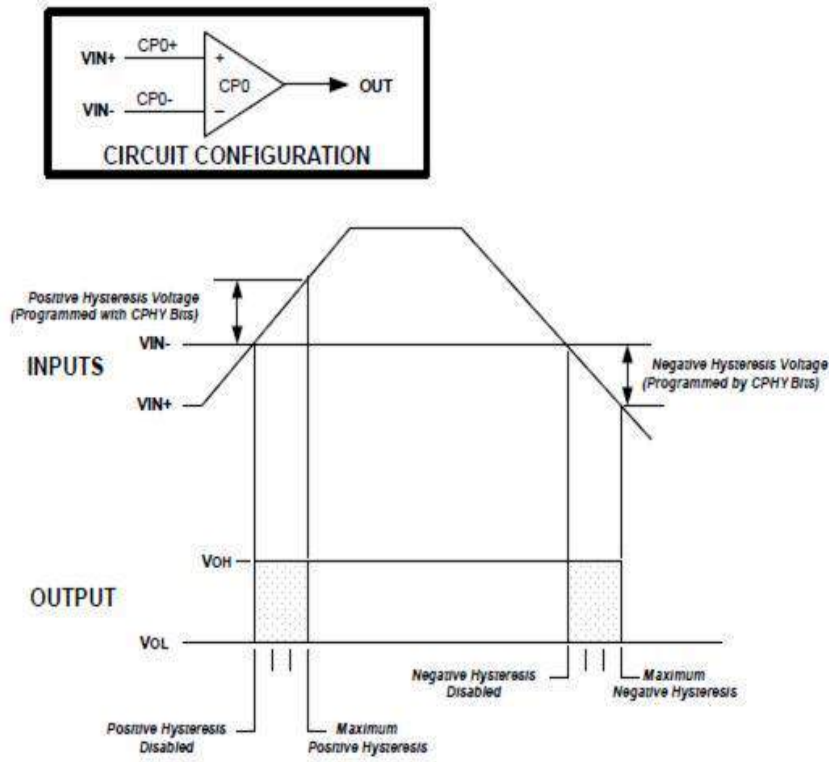
The Comparator output can be polled in software, used as an interrupt source, and/or routed to a Port pin. When routed to a Port pin, the Comparator output is available and asynchronous or synchronous to the system clock; the asynchronous output is available even in STOP mode (with no active system clock). 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. Users can program the value of the 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. The least start-up time of the comparator is less than 5 $\mu$ s.



**Figure 29-6: Comparator Hysteresis Plot**

## **30. USB2.0 Full-Speed Device Controller (USBC)**

### **30.1. Introduction**

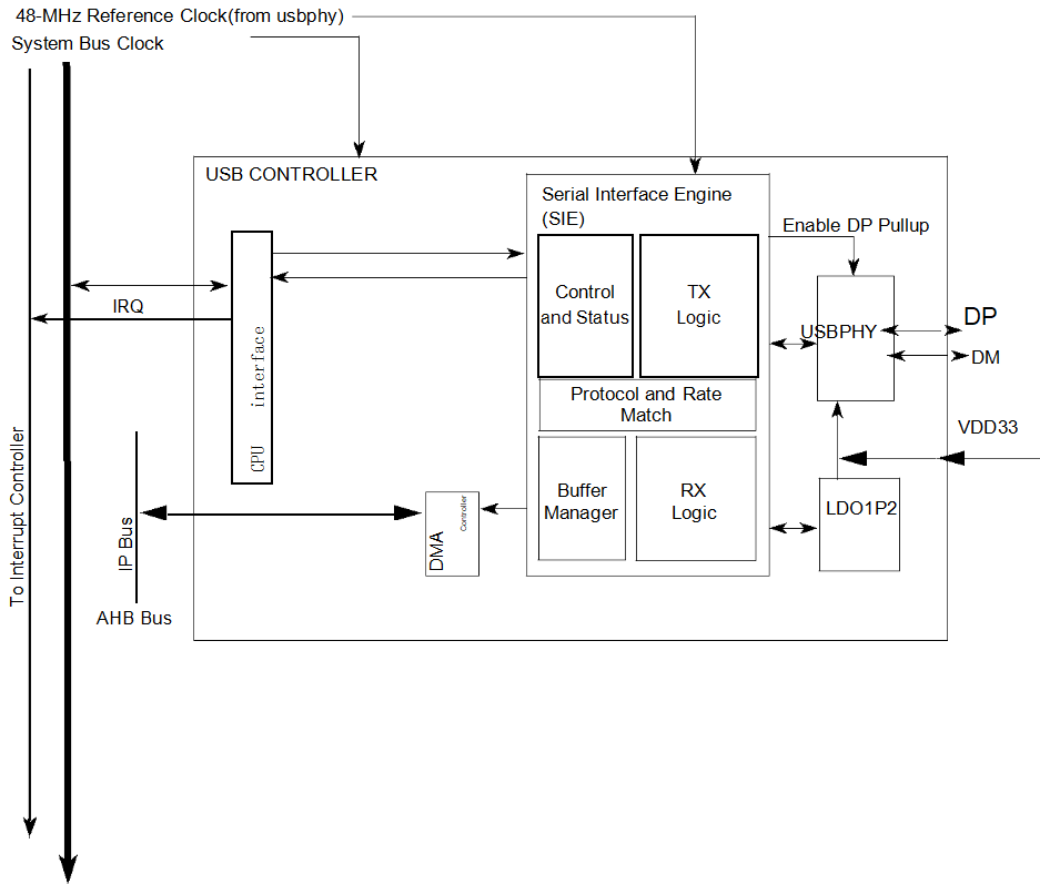
This section describes the USB2.0 Full Speed Device-Only controller. The device implementation in this module provides solutions for implementing a USB 2.0 full-speed/low-speed compliant peripheral.

### **30.2. Features**

Features of the USB module include:

- USB2.0 Device-Only Function Controller
- USB 2.0 Compliant
  - 12 Mbps Full-Speed (FS) Data Rate
  - USB Data Control Logic:
    - Packet identification and decoding/generation
    - CRC generation and checking
    - NRZI (non-return-to-zero inverted) encoding/decoding
    - Bit-stuffing
    - Sync detection
    - End-of-packet detection
- Eight USB Endpoints
- USB RAM
  - 2048 bytes of buffer RAM shared between system and USB module
  - RAM may be allocated as buffers for USB controller or extra system RAM resource
- USB Reset Options
  - USB Module reset generated by MCU
  - Bus reset generated by the host, which triggers a CPU interrupt
- Suspend and Resume Operations with Remote Wakeup Support
- Transceiver Features
  - Converts USB differential voltages to digital logic signal levels
  - On-chip USB Pullup Resistor
- On-chip 3.3 -V Regulator

### 30.3. Block Diagram



**Figure 30-1: USB Full-speed Device(USB module) Block Diagram**

### 30.4. Modes Of Operation

This subsection describes the three low-power modes.

#### 30.4.1. Wait Mode

In wait mode, the USB module can continue to operate normally and can be configured to exit the low-power mode by generating an interrupt request.

#### 30.4.2. Doze Mode

In Doze mode, the USB module can continue to operate normally and can be configured to exit the low-power mode by generating an interrupt request.

#### 30.4.3. Stop Mode

The USB Module is optionally available in stop mode. A reduced current consumption mode may be required for USB suspend mode per USB Specification Rev. 2.0, and stop mode is useful for achieving lower current consumption for the MCU and hence the overall USB device. Before entering stop via firmware, users must ensure that the device settings are configured for stop such that the USB suspend current consumption targets are achieved .

The USB module is notified about entering suspend mode when the SLEEP flag is set; this occurs after the USB bus is idle for 3ms. The USB device suspend mode current consumption level requirements are defined by the USB Specification Rev. 2.0 (500uA for low-power and 2.5mA for high-power with remote-wakeup

enabled).

If USBRESMEN is set, and a K-state (resume signaling) is detected on the USB bus, the RESUME bit will become set. This will trigger an asynchronous interrupt that will wake the MCU from stop mode and enable clocks to the USB module.

## 30.5. Memory Map and Registers

The USB Module memory map is shown in **Table 30-1**. The USB Controller base address is **0x4016\_0000**. This subsection describes the memory map and register structure for USB module.

### 30.5.1. Memory Map

Refer to **Table 30-1** for a description of the memory map.

**Table 30-1: USB Module Memory Map**

Offset Address	Bits[31:0]	Access <sup>(1)</sup>
0x0000	Reserved	S/U
0x0004	Reserved	S/U
0x0008	Reserved	S/U
0x000C	Reserved	S/U
0x001C	USBPHY Control Register 1 (USBPHY_CTRL1)	S/U
0x0080	Interrupt Status Register (INT_STAT)	S/U
0x0084	Interrupt Enable Register (INT_ENB)	S/U
0x0088	Error Interrupt Status Register (ERR_STAT)	S/U
0x008C	Error Interrupt Enable Register (ERR_ENB)	S/U
0x0090	Status Register (STAT)	S/U
0x0094	Control Register (CTL)	S/U
0x0098	Address Register (ADDR)	S/U
0x009C	EBT Page Register 1 (EBT_PAGE_01)	S/U
0x00A0	Frame Number Register (FRMNUML)	S/U
0x00A4	Frame Number Register (FRMNUMH)	S/U
0x00A8	Token Register (TOKEN)	S/U
0x00AC	SOF Threshold Register (SOF_THLD)	S/U
0x00B0	EBT Page Register 2 (EBT_PAGE_02)	S/U
0x00B4	EBT Page Register 3 (EBT_PAGE_03)	S/U
0x00C0	Endpoint Control Registers (ENDPT0)	S/U
0x00C4	Endpoint Control Registers (ENDPT1)	S/U
0x00C8	Endpoint Control Registers (ENDPT2)	S/U
0x00CC	Endpoint Control Registers (ENDPT3)	S/U
0x00D0	Endpoint Control Registers (ENDPT4)	S/U
0x00D4	Endpoint Control Registers (ENDPT5)	S/U
0x00D8	Endpoint Control Registers (ENDPT6)	S/U



Offset Address	Bits[31:0]	Access <sup>(1)</sup>
0x00DC	Endpoint Control Registers (ENDPT7)	S/U
0x0100	USBPHY Control Register 2 (USBPHY_CTRL2)	S/U
0x0104	USB PHY Observe Register	S/U
	(USB_PHY_OBSERVE)	
0x0108	USB PHY GPIO Register	S/U
0x010C	USB Resume Wakeup Enable Register	S/U
	(USB_RESMEN)	
0x0118	USBPHY Control Register 3 (USBPHY_CTRL3)	S/U
0x011C	USBPHY Control Register 4 (USBPHY_CTRL4)	S/U

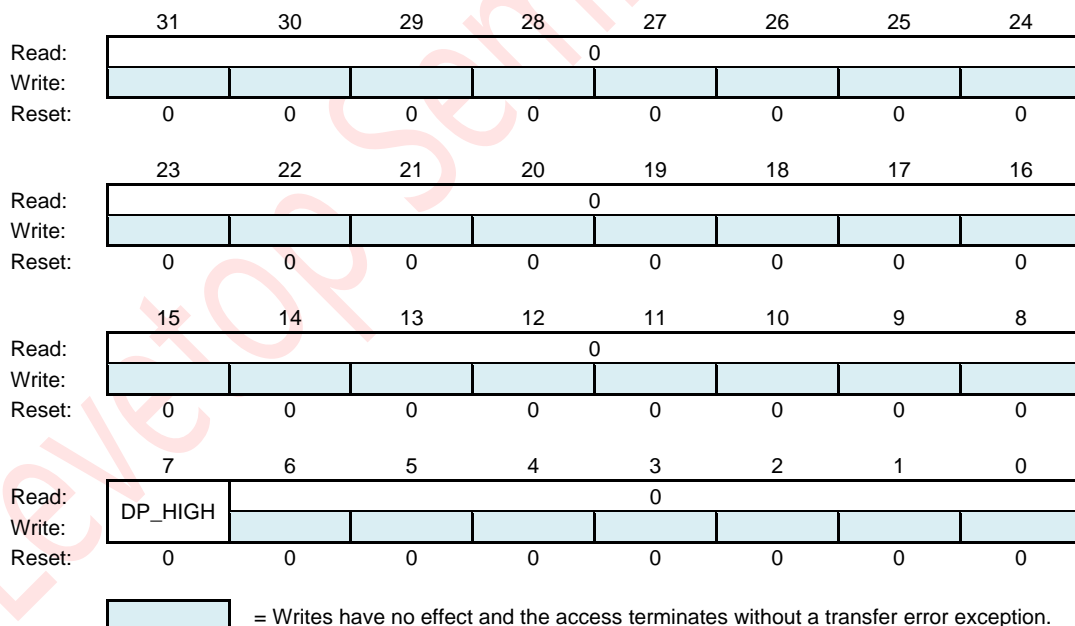
**Note (1) :** S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. Accessing supervisor only address in user mode has no effect and result in a cycle termination transfer error.

### 30.5.2. Register Descriptions

#### 30.5.2.1. USBPHY Control Register 1 (USBPHY\_CTRL1)

The USBPHY Control Register controls the operation of Data Line termination resistors.

**Address: USBC\_BASEADDR+0x0000\_001C**



**Figure 30-2: USBPHY Control Register 1 (USBPHY\_CTRL1)**

**DP\_HIGH** — D+ Data Line pullup resistor enable

0 = D+ pullup resistor is disabled

1 = D+ pullup resistor is enabled.

30.5.2.2. Interrupt Status Register (INT\_STAT)

The Interrupt Status Register contains bits for each of the interrupt sources within the USB Module. Each of these bits is qualified with their respective interrupt enable bits. All bits of this register are logically OR'd together along with the OTG Interrupt Status Register (OTG\_STAT) to form a single interrupt source for the processor's interrupt controller. After an interrupt bit has been set it may only be cleared by writing a one to the respective interrupt bit. This register contains the value of 0x00 after a reset.

Address: USBC\_BASEADDR+0x0000\_0080

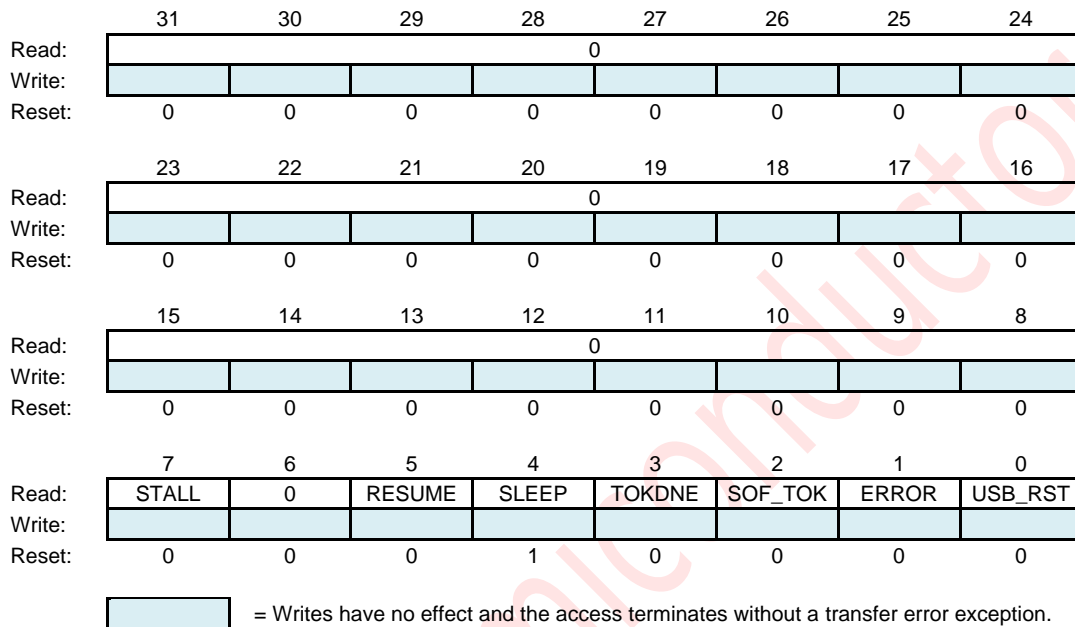


Figure 30-3: Interrupt Status Register (INT\_STAT)

**STALL** — Stall Interrupt, and cleared by writing 1 to this bit.

In Device mode this bit is asserted when a STALL handshake is sent by the USB Module. In Host mode this bit is set when the USB Module detects a STALL acknowledge during the handshake phase of a USB transaction. This interrupt can be used to determine whether the last USB transaction was completed successfully or if it stalled.

**RESUME** — Resume Interrupt, and cleared by writing 1 to this bit.

This bit is set depending on the DP/DM signals, and can be used to signal remote wake-up signaling on the USB bus. When not in suspend mode, this interrupt should be disabled.

**SLEEP** — Sleep Interrupt, and cleared by writing 1 to this bit.

This bit is set when the USB Module detects a constant idle on the USB bus for 3 milliseconds. The sleep timer is reset by activity on the USB bus.

**TOKDNE** — Token Done Interrupt, and cleared by writing 1 to this bit.

This bit is set when the current token being processed has completed. The processor should immediately read the STAT register to determine the EndPoint and EB entry used for this token. Clearing this bit (by writing a one) causes the STAT register to be cleared or the STAT holding register to be loaded into the STAT register.

**SOF\_TOK** — SOF Token Interrupt, and cleared by writing 1 to this bit.

This bit is set when the USB Module receives a Start Of Frame (SOF) token. In Host mode this bit is set when the SOF threshold is reached, so that software can prepare for the next SOF.

**ERROR** — Error Interrupt, and cleared by writing 1 to this bit.

This bit is set when any of the error conditions within the ERR\_STAT register occurs. The processor must then read the ERR\_STAT register to determine the source of the error.

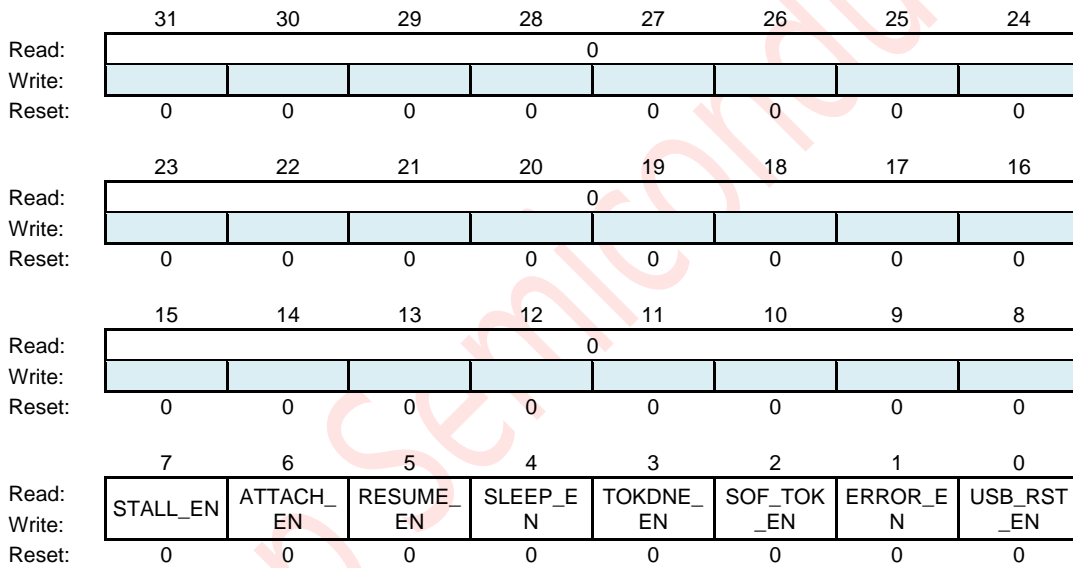
**USB\_RST** — USB RST Interrupt, and cleared by writing 1 to this bit.

This bit is set when the USB Module has decoded a valid USB reset. This informs the Microprocessor that it should write 0x00 into the address register and enable endpoint 0. USB\_RST is set after a USB reset has been detected for 2.5 microseconds. It is not asserted again until the USB reset condition has been removed and then reasserted.

**30.5.2.3. Interrupt Enable Register (INT\_ENB)**

The Interrupt Enable Register contains enable bits for each of the interrupt sources within the USB Module. Setting any of these bits enables the respective interrupt source in the INT\_STAT register. This register contains the value of 0x00 after a reset.

**Address: USBC\_BASEADDR+0x0000\_0084**



[shaded box] = Writes have no effect and the access terminates without a transfer error exception.

**Figure 30-4: Interrupt Enable Register (INT\_ENB)**

**STALL\_EN** — Stall Interrupt Enable

- 0 = Stall Interrupt is disabled
- 1 = Stall Interrupt is enabled.

**ATTACH\_EN** — Attach Interrupt Enable.

- 0 = Attach Interrupt is disabled.
- 1 = Attach Interrupt is enabled.

**RESUME\_EN** — Resume Interrupt Enable.

- 0 = Resume Interrupt is disabled.
- 1 = Resume Interrupt is enabled.

**SLEEP\_EN** — Sleep Interrupt Enable.

0 = Sleep Interrupt is disabled.

1 = Sleep Interrupt is enabled.

**TOKDNE\_EN** — Token Done Interrupt Enable.

0 = Token Done Interrupt is disabled.

1 = Token Done Interrupt is enabled.

**SOF\_TOK\_EN** — SOF Token Interrupt Enable.

0 = SOF Token Interrupt is disabled.

1 = SOF Token Interrupt is enabled.

**ERROR\_EN** — Error Interrupt Enable.

0 = Error Interrupt is disabled.

1 = Error Interrupt is enabled.

**USB\_RST\_EN** — USB Reset Interrupt Enable.

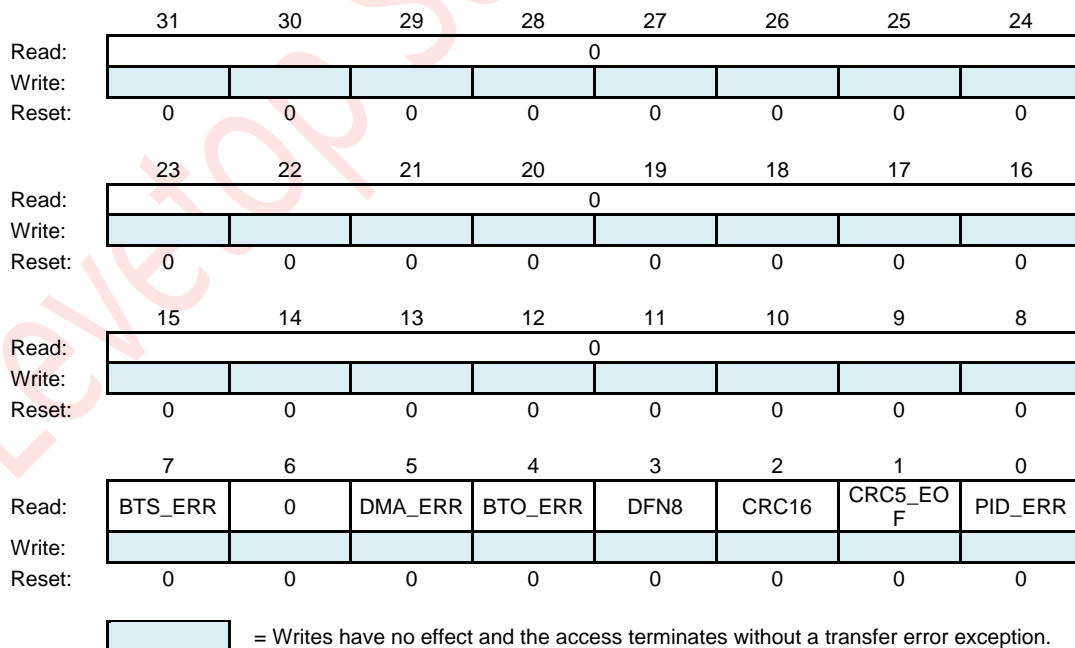
0 = USB Reset Interrupt is disabled.

1 = USB Reset Interrupt is enabled.

### 30.5.2.4. Error Interrupt Status Register (ERR\_STAT)

The Error Interrupt Status Register contains enable bits for each of the error sources within the USB Module. Each of these bits is qualified with their respective error enable bits. All bits of this Register are logically OR'd together and the result is placed in the ERROR bit of the INT\_STAT register. After an interrupt bit has been set it may only be cleared by writing a one to the respective interrupt bit. Each bit is set as soon as the error conditions is detected. Therefore, the interrupt does not typically correspond with the end of a token being processed. This register contains the value of 0x00 after a reset.

**Address: USBC\_BASEADDR+0x0000\_0088**



**Figure 30-5: Error Interrupt Status Register (ERR\_STAT)**

**BTS\_ERR** — This bit is set when a bit stuff error is detected. If set, the corresponding packet is rejected due to the error.

**DMA\_ERR** — This bit is set if the USB Module has requested a DMA access to read a new EBT but has not been given the bus before it needs to receive or transmit data. If processing a TX transfer, this would cause a transmit data underflow condition. If processing a RX transfer, this would cause a receive data overflow condition. This interrupt is useful when developing device arbitration hardware for the microprocessor and the USB Module to minimize bus request and bus grant latency. This bit is also set if a data packet to or from the host is larger than the buffer size allocated in the EBT. In this case the data packet is truncated as it is put into buffer memory.

**BTO\_ERR** — This bit is set when a bus turnaround timeout error occurs. The USB Module contains a bus turnaround timer that keeps track of the amount of time elapsed between the token and data phases of a SETUP or OUT TOKEN or the data and handshake phases of a IN TOKEN. If more than 16-bits times are counted from the previous EOP before a transition from IDLE, a bus turnaround timeout error occurs.

**DFN8** — This bit is set if the data field received was not 8-bits in length. USB Specification 1.0 requires that data fields be an integral number of bytes. If the data field was not an integral number of bytes, this bit is set.

**CRC16** — This bit is set when a data packet is rejected due to a CRC16 error.

**CRC5\_EOF** — This error interrupt has two functions. When the USB Module is operating in device mode (HOST\_MODE\_EN = 0), this interrupt detects CRC5 errors in the token packets generated by the host. If set, the token packet was rejected due to a CRC5 error. When the USB Module is operating in host mode (HOST\_MODE\_EN = 1), this interrupt detects End Of Frame (EOF) error conditions. This occurs when the USB Module is transmitting or receiving data and the SOF counter reaches zero. This interrupt is useful when developing USB packet scheduling software to ensure that no USB transactions cross the start of the next frame.

**PID\_ERR** — This bit is set when the PID check field fails.

30.5.2.5. Error Interrupt Enable Register (ERR\_ENB)

The Error Interrupt Enable Register contains enable bits for each of the error interrupt sources within the USB Module. Setting any of these bits enables the respective interrupt source in the ERR\_STAT register. Each bit is set as soon as the error conditions is detected. Therefore, the interrupt does not typically correspond with the end of a token being processed. This register contains the value of 0x00 after a reset.

Address: USBC\_BASEADDR+0x0000\_008C

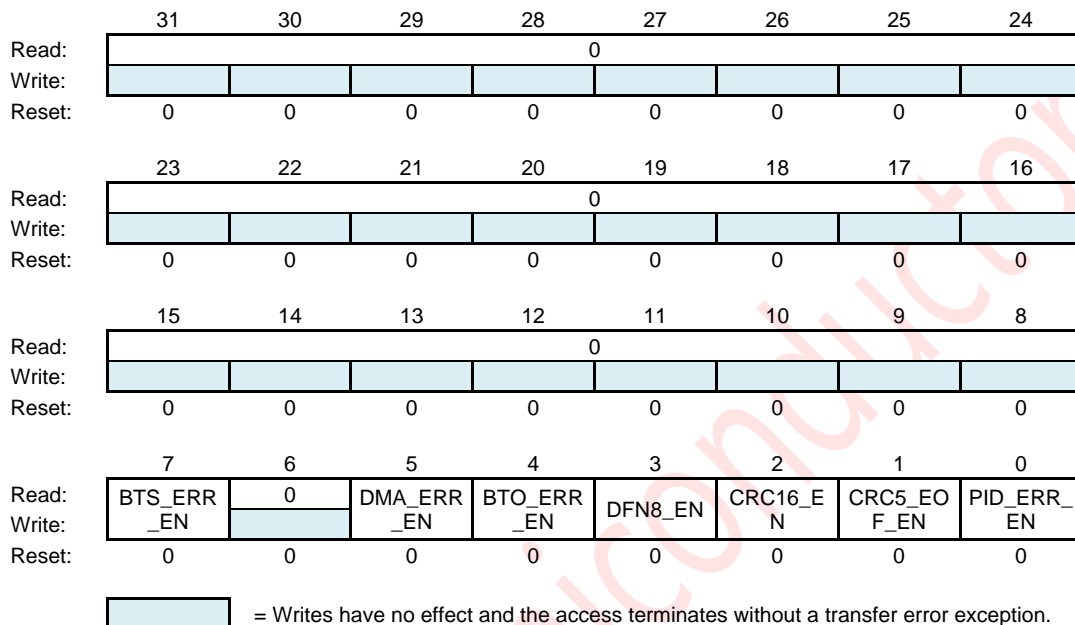


Figure 30-6: Error Interrupt Enable Register (ERR\_ENB)

**BTS\_ERR\_EN** — BTS\_ERR Interrupt Enable

0 = BTS\_ERR Interrupt is disabled.  
1 = BTS\_ERR Interrupt is enabled.

**DMA\_ERR\_EN** — DMA\_ERR Interrupt Enable.

0 = DMA\_ERR Interrupt is disabled.  
1 = DMA\_ERR Interrupt is enabled.

**BTO\_ERR\_EN** — BTO\_ERR Interrupt Enable.

0 = BTO\_ERR Interrupt is disabled.  
1 = BTO\_ERR Interrupt is enabled.

**DFN8\_EN** — DFN8 Interrupt Enable.

0 = DFN8 Interrupt is disabled.  
1 = DFN8 Interrupt is enabled.

**CRC16\_EN** — CRC16 Interrupt Enable.

0 = CRC16 Interrupt is disabled.  
1 = CRC16 Interrupt is enabled.

**CRC5\_EOF\_EN** — CRC5\_EOF Interrupt Enable.

- 0 = CRC5\_EOF Interrupt is disabled.
- 1 = CRC5\_EOF Interrupt is enabled.

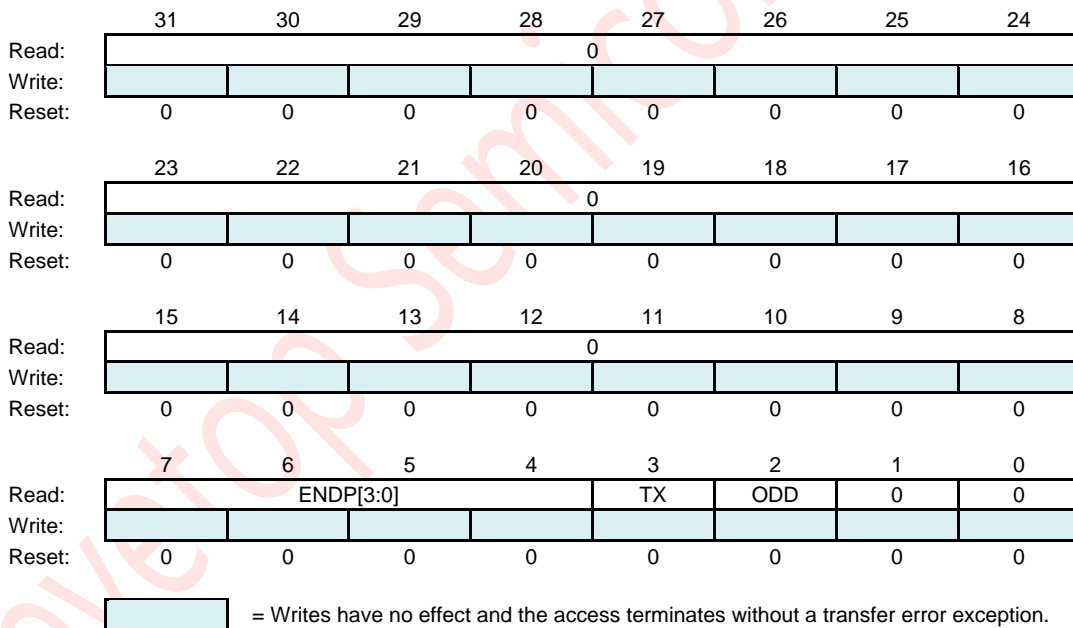
**PID\_ERR\_EN** — PID\_ERR Interrupt Enable.

- 0 = PID\_ERR Interrupt is disabled.
- 1 = PID\_ERR Interrupt is enabled.

**30.5.2.6. Status Register (STAT)**

The Status Register reports the transaction status within the USB Module. When the processor's interrupt controller has received a TOK\_DNE interrupt, the Status Register should be read to determine the status of the previous endpoint communication. The data in the status register is valid when the TOK\_DNE interrupt bit is asserted. The STAT register is actually a read window into a status FIFO maintained by the USB Module. When the USB Module uses a EB entry, it updates the Status Register. If another USB transaction is performed before the TOK\_DNE interrupt is serviced, the USB Module stores the status of the next transaction in the STAT FIFO. Thus the STAT register is actually a four byte FIFO that allows the processor core to process one transaction while the SIE is processing the next transaction. Clearing the TOK\_DNE bit in the INT\_STAT register causes the SIE to update the STAT register with the contents of the next STAT value. If the data in the STAT holding register is valid, the SIE immediately reasserts to TOK\_DNE interrupt.

**Address: USBC\_BASEADDR+0x0000\_0090**



**Figure 30-7: Status Register (STAT)**

**ENDP[3:0]** — Endpoint Number.

These four bits encode the endpoint address that received or transmitted the previous token. This allows the microcontroller to determine which EBT entry was updated by the last USB transaction.

- 0000 Endpoint 0
- 0001 Endpoint 1
- 0010 Endpoint 2
- 0011 Endpoint 3
- 0100 Endpoint 4

- 0101 Endpoint 5
- 0110 Endpoint 6
- 0111 Endpoint 7

**TX** — Transmit Indicator.

- 0 = The most recent transaction was a Receive operation.
- 1 = The most recent transaction was a Transmit operation.

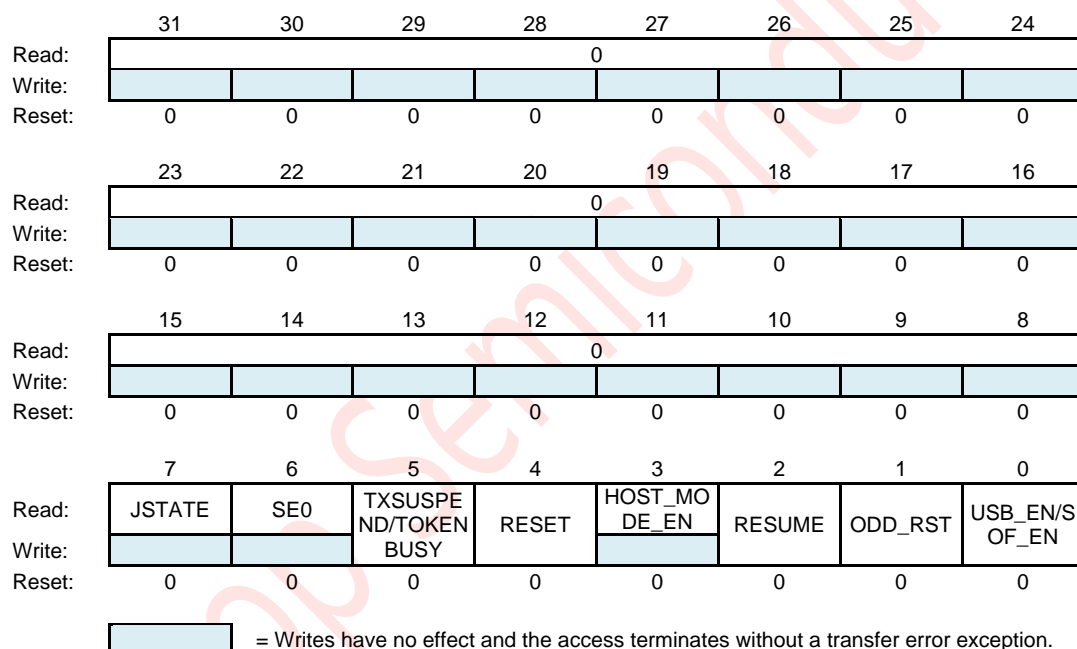
**ODD** — Odd/Even Transaction.

This bit is set if the last Endpoint Buffer Table updated was in the odd bank of the EBT.

### 30.5.2.7. Control Register (CTL)

The Control Register provides various control and configuration information for the USB Module.

**Address: USBC\_BASEADDR+0x0000\_0094**



**Figure 30-8: Control Register (CTL)**

**JSTATE** — Live USB differential receiver JSTATE signal

The polarity of this signal is affected by the current state of LS\_EN .

**SE0** — Live USB Single Ended Zero signal

The polarity of this signal is affected by the current state of LS\_EN .

#### **TXSUSPEND / TOKENBUSY**

When the USB Module is in Host mode, TOKEN\_BUSY is set when the USB Module is busy executing a USB token and no more token commands should be written to the Token Register. Software should check this bit before writing any tokens to the Token Register to ensure that token commands are not lost. In Device mode TXSUSPEND is set when the SIE has disabled packet transmission and reception. Clearing this bit allows the SIE to continue token processing. This bit is set by the SIE when a Setup Token is received, allowing software to dequeue any pending packet transactions in the EBT before resuming token processing.



**RESET** — This bit is invalid for current device-role-only function.

Setting this bit enables the USB Module to generate USB reset signaling. This allows the USB Module to reset USB peripherals. This control signal is only valid in Host mode (HOST\_MODE\_EN = 1). Software must set RESET to 1 for the required amount of time and then clear it to 0 to end reset signaling.

**HOST\_MODE\_EN** — This bit is read-only for this device-role-only function.

When set to 1, this bit enables the USB Module to operate in Host mode. In host mode, the USB module performs USB transactions under the programmed control of the host processor.

#### **RESUME**

When set to 1, this bit enables the USB Module to execute resume signaling.

This allows the USB Module to perform remote wake-up. Software must set RESUME to 1 for the required amount of time and then clear it to 0. If the HOST\_MODE\_EN bit is set, the USB module appends a Low Speed End of Packet to the Resume signaling when the RESUME bit is cleared.

#### **ODD\_RST**

Setting this bit to 1 resets all the EBT ODD ping/pong bits to 0, which then specifies the EVEN EBT bank.

**USB\_EN/SOF\_EN** — USB Enable.

Setting this bit causes the SIE to reset all of its ODD bits to the EBTs. Therefore, setting this bit resets much of the logic in the SIE. When host mode is enabled, clearing this bit causes the SIE to stop sending SOF tokens.

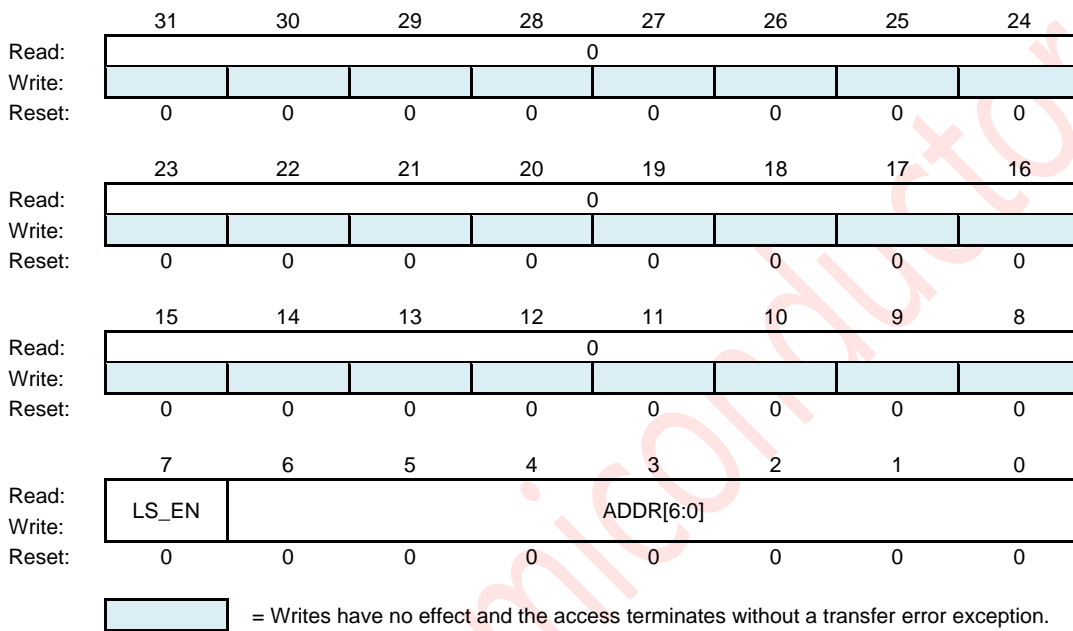
0 = The USB Module is disabled.

1 = The USB Module is enabled.

**30.5.2.8. Address Register (ADDR)**

The Address Register holds the unique USB address that the USB Module decodes when in Peripheral mode (HOST\_MODE\_EN = 0). When operating in Host mode (HOST\_MODE\_EN = 1) the USB Module transmits this address with a TOKEN packet. This enables the USB Module to uniquely address an USB peripheral. In either mode, the USB\_EN bit within the control register must be set. The Address Register is reset to 0x00 after the reset input becomes active or the USB Module decodes a USB reset signal. This action initializes the Address Register to decode address 0x00 as required by the USB specification.

**Address: USBC\_BASEADDR+0x0000\_0098**



**Figure 30-9: Address Register (ADDR)**

**LS\_EN** — Low Speed Enable bit.

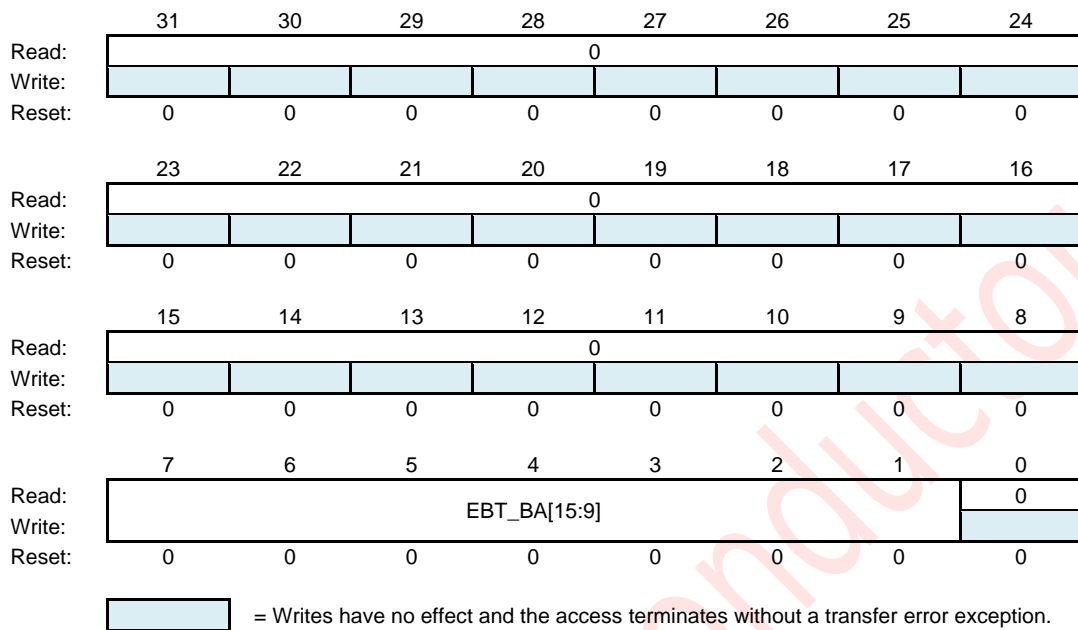
This bit informs the USB Module that the next token command written to the token register must be performed at low speed. This enables the USB Module to perform the necessary preamble required for low-speed data transmissions.

**ADDR[6:0]** — USB address.

This 7-bits value defines the USB address that the USB Module decodes in peripheral mode, or transmits when in host mode.

**30.5.2.9. EBT Page Register 1 (EBT\_PAGE\_01)**

**Address: USBC\_BASEADDR+0x0000\_009C**



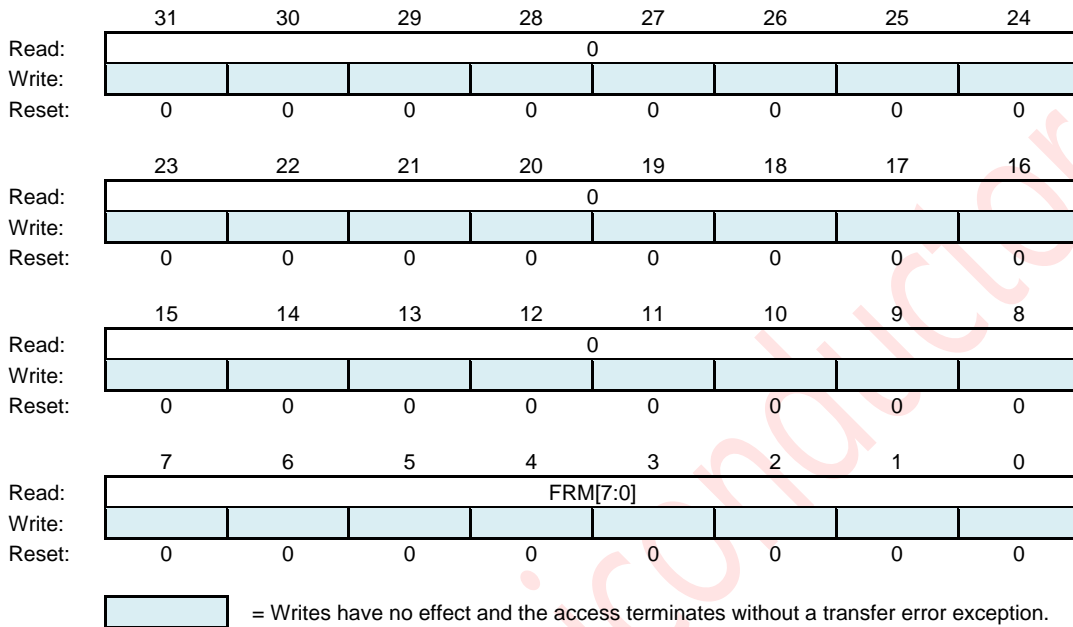
**Figure 30-10: EBT Page Register 1 (EBT\_PAGE\_01)**

**EBT\_BA[15:9]** — This 7-bits field provides address bits 15 through 9 of the EBT base address, which defines where the Buffer Descriptor Table resides in system memory.

**30.5.2.10. Frame Number Register (FRMNUML)**

The Frame Number Registers contains the 11-bits frame number. The Frame Number Register requires two 8-bits registers to implement. The low order byte is contained in FRMNUML, and the high order byte is contained in FRMNUMH. These registers are updated with the current frame number whenever a SOF TOKEN is received.

**Address: USBC\_BASEADDR+0x0000\_00A0**



**Figure 30-11: Frame Number Register (FRMNUML)**

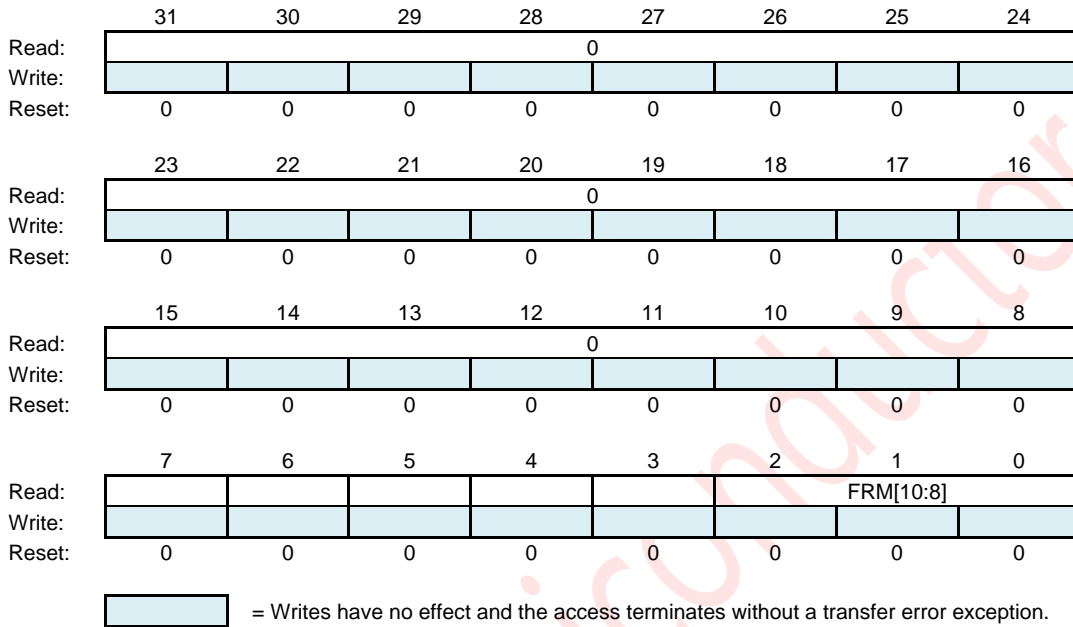
**FRM[7:0]** — Frame Number.

These bits represent the low order bits of the 11 bit Frame Number.

**30.5.2.11. Frame Number Register (FRMNUMH)**

The Frame Number Registers contains the 11-bits frame number. The Frame Number Register requires two 8-bits registers to implement. The low order byte is contained in FRMNUML, and the high order byte is contained in FRMNUMH. These registers are updated with the current frame number whenever a SOF TOKEN is received.

**Address: USBC\_BASEADDR+0x0000\_00A4**



**Figure 30-12: Frame Number Register (FRMNUMH)**

**FRM[10:8]** — Frame Number.

These bits represent the high order bits of the 11-bits Frame Number.

30.5.2.12. Token Register (TOKEN)

The Token Register is used to perform USB transactions when in host mode (HOST\_MODE\_EN = 1). When the processor core wishes to execute a USB transaction to a peripheral, it writes the TOKEN type and endpoint to this register. After this register has been written, the USB module begins the specified USB transaction to the address contained in the address register. The processor core should always check that the TOKEN\_BUSY bit in the control register is not set before performing a write to the Token Register. This ensures token commands are not overwritten before they can be executed. The address register and endpoint control register 0 are also used when performing a token command and therefore must also be written before the Token Register. The address register is used to correctly select the USB peripheral address transmitted by the token command. The endpoint control register determines the handshake and retry policies used during the transfer.

Address: USBC\_BASEADDR+0x0000\_00A8

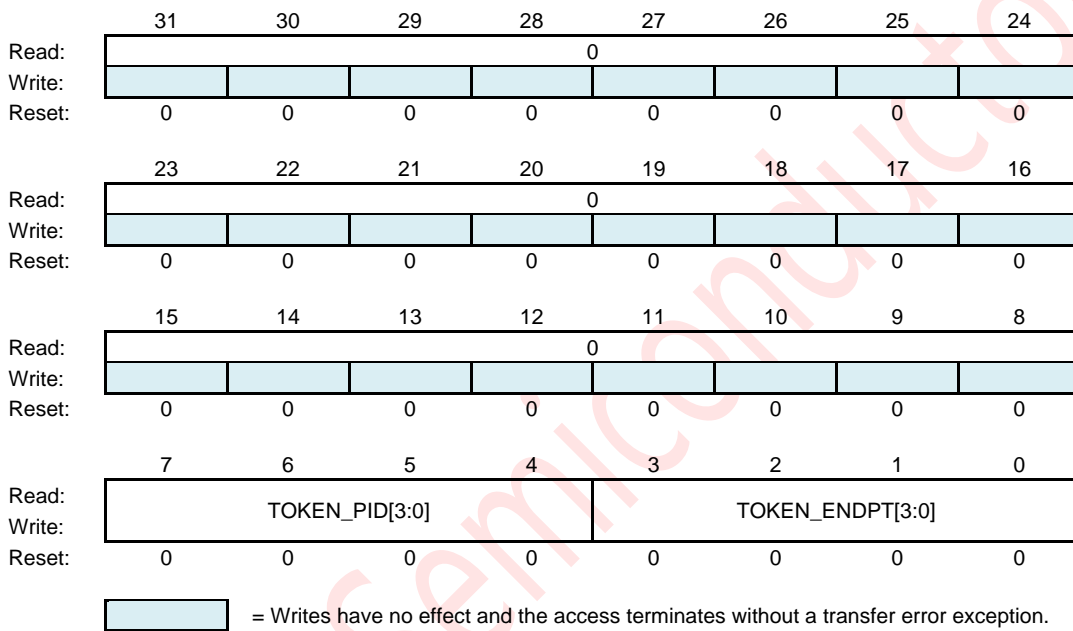


Figure 30-13: Token Register (TOKEN)

**TOKEN\_PID[3:0]** — This 4-bits field contains the token type executed by the USB Module.

0001 OUT Token. USB Module performs an OUT (TX) transaction.

1001 IN Token. USB Module performs an In (RX) transaction.

1101 SETUP Token. USB Module performs a SETUP (TX) transaction

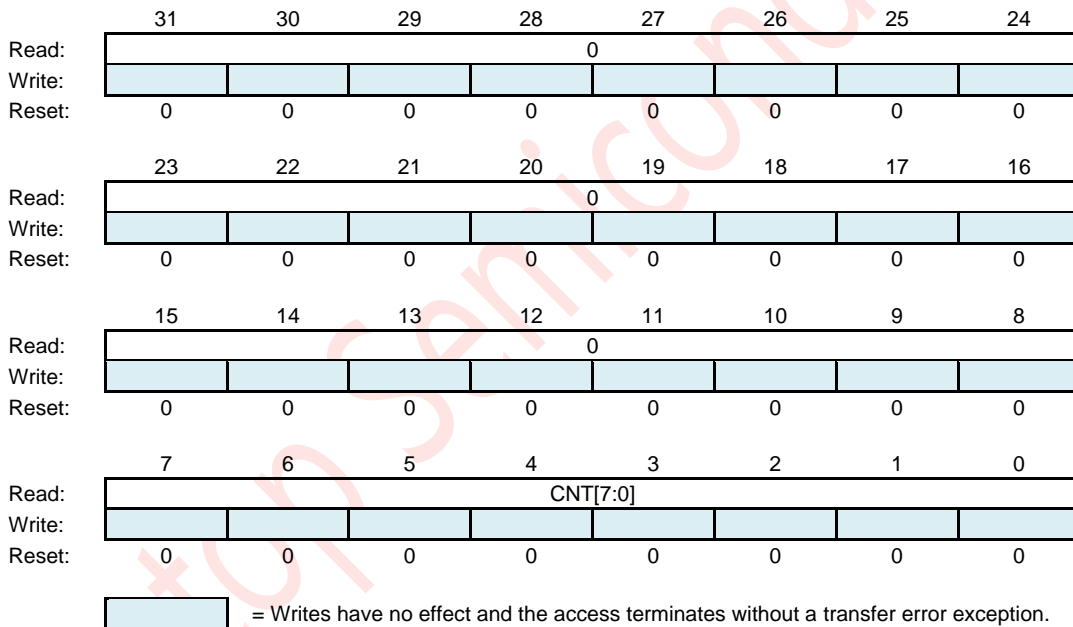
**TOKEN\_ENDPT[3:0]** — This 4-bits field holds the Endpoint address for the token command. The 4-bits value written must be a valid endpoint.

**30.5.2.13. SOF Threshold Register (SOF\_THLD)**

The SOF Threshold Register is used only in Host mode (HOST\_MODE\_EN = 1). When in Host mode, the 14-bits SOF counter counts the interval between SOF frames. The SOF must be transmitted every 1msec so the SOF counter is loaded with a value of 12000. When the SOF counter reaches zero, a Start Of Frame (SOF) token is transmitted. The SOF threshold register is used to program the number of USB byte times before the SOF to stop initiating token packet transactions. This register must be set to a value that ensures that other packets are not actively being transmitted when the SOF time counts to zero. When the SOF counter reaches the threshold value, no more tokens are transmitted until after the SOF has been transmitted. The value programmed into the threshold register must reserve enough time to ensure the worst case transaction completes. In general, the worst case transaction is a IN token followed by a data packet from the target followed by the response from the host. The actual time required is a function of the maximum packet size on the bus. Typical values for the SOF threshold are:

- 64-byte packets = 74;
- 32-byte packets = 42;
- 16-byte packets = 26;
- 8-byte packets = 18.

**Address: USBC\_BASEADDR+0x0000\_00AC**



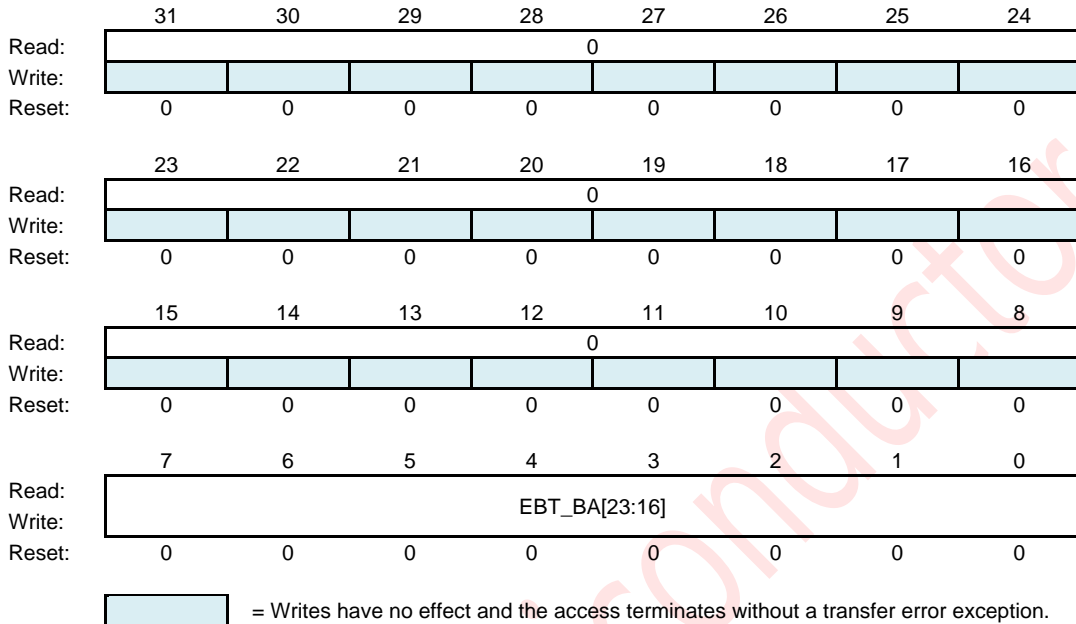
**Figure 30-14: SOF Threshold Register (SOF\_THLD)**

**CNT[7:0]** — This 8-bits field represents the SOF count threshold in byte times. This register is read only in this device.

**30.5.2.14. EBT Page Register 2 (EBT\_PAGE\_02)**

The Endpoint Buffer Table Page Register 2 contains an 8-bits value used to compute the address where the current Endpoint Buffer Table (EBT) resides in system memory.

**Address: USBC\_BASEADDR+0x0000\_00B0**



**Figure 30-15: EBT Page Register 2 (EBT\_PAGE\_02)**

**EBT\_BA[23:16]**

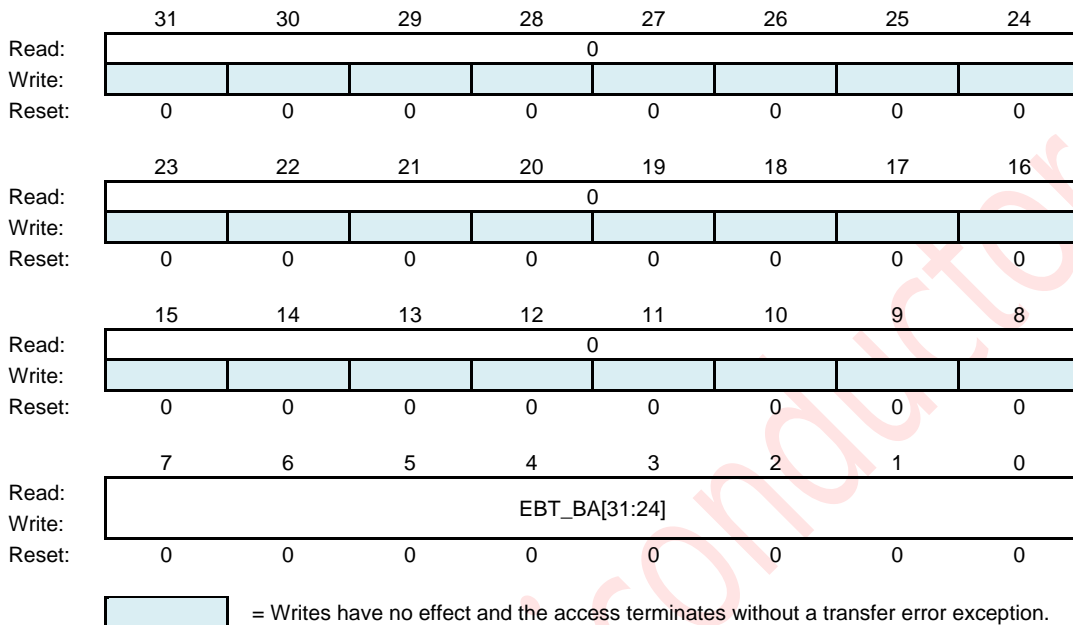
This 8-bits field provides address bits 23 through 16 of the EBT base address, which defines where the Buffer Descriptor Table resides in system memory.



**30.5.2.15. EBT Page Register 3 (EBT\_PAGE\_03)**

The Endpoint Buffer Table Page Register 3 contains an 8-bits value used to compute the address where the current Endpoint Buffer Table (EBT) resides in system memory.

**Address: USBC\_BASEADDR+0x0000\_00B4**



**Figure 30-16: EBT Page Register 3 (EBT\_PAGE\_03)**

**EBT\_BA[31:24]**

This 8-bits field provides address bits 31 through 24 of the EBT base address, which defines where the Buffer Descriptor Table resides in system memory.

**30.5.2.16. Endpoint Control Registers (ENDPTn)**

The Endpoint Control Registers contain the endpoint control bits for each of the 8 endpoints available within the USB Module for a decoded address. The format for these registers is shown in the following figure. Endpoint 0 (ENDPT0) is associated with control pipe 0, which is required for all USB functions. Therefore, after a USB\_RST interrupt occurs, the processor core should set the ENDPT0 register to contain 0x0D.

In Host mode, ENDPT0 is used to determine the handshake, retry and low speed characteristics of the host transfer. For Host mode control, bulk, and interrupt transfers the EP\_HSHK bit should be set to 1. For Isochronous transfers, it should be set to 0. Common values to use for ENDPT0 in host mode are 0x4D for Control, Bulk, and Interrupt transfers, and 0x4C for Isochronous transfers.

Registers Offset address:

- ENDPT0 Address: USBC\_BASEADDR+0x0000\_00C0**
- ENDPT1 Address: USBC\_BASEADDR+0x0000\_00C4**
- ENDPT2 Address: USBC\_BASEADDR+0x0000\_00C8**
- ENDPT3 Address: USBC\_BASEADDR+0x0000\_00CC**
- ENDPT4 Address: USBC\_BASEADDR+0x0000\_00D0**
- ENDPT5 Address: USBC\_BASEADDR+0x0000\_00D4**
- ENDPT6 Address: USBC\_BASEADDR+0x0000\_00D8**
- ENDPT7 Address: USBC\_BASEADDR+0x0000\_00DC**

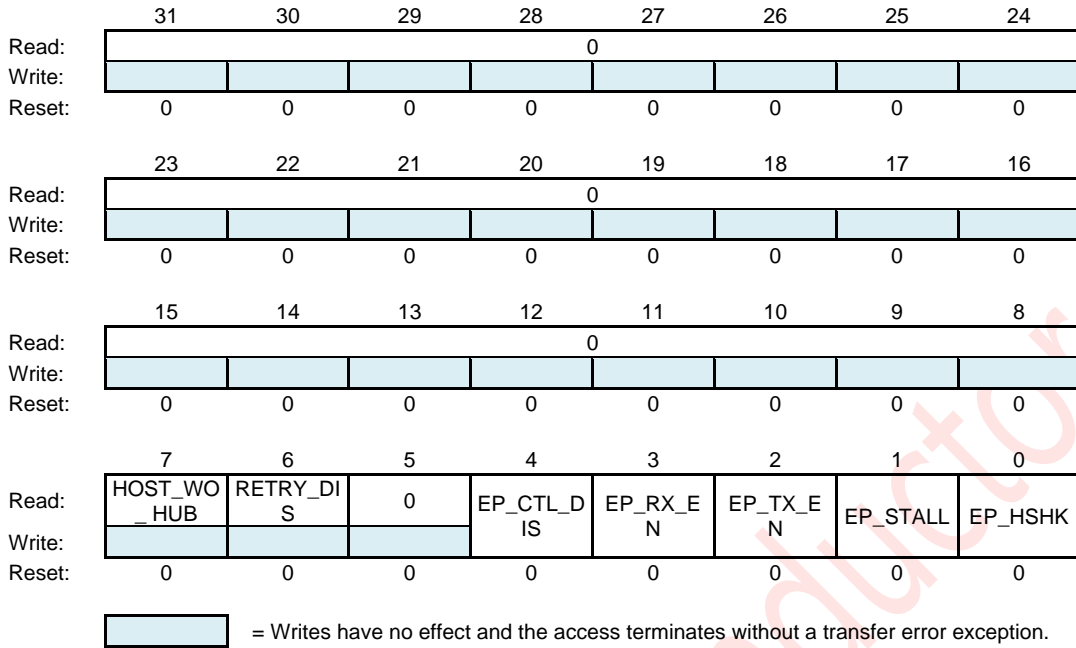


Figure 30-17: Endpoint Control Registers(ENDPTn, n = 0-7)

**HOST\_WO\_HUB** — This is a Host mode only bit and is only present in the control register for endpoint 0 (ENDPT0). This bit is read only in this device.

When set, this bit allows the host to communicate to a directly connected low speed device. When cleared, the host produces the PRE\_PID, then switches to low speed signaling, when sending a token to a low speed device as required to communicate with a low speed device through a hub.

**RETRY\_DIS** — This is a Host mode only bit and is only present in the control register for endpoint 0 (ENDPT0). This bit is read only in this device.

When set, this bit causes the host to not retry NAK'ed (Negative Acknowledgement) transactions. When a transaction is NAKed, the EBT PID field is updated with the NAK PID, and the TOKEN\_DNE interrupt is set. When this bit is cleared, NAKed transactions is retried in hardware. This bit must be set when the host is attempting to poll an interrupt endpoint.

**EP\_CTL\_DIS** — This bit, when set, disables control (SETUP) transfers. When this bit is cleared, control transfers are enabled. This applies if and only if the EP\_RX\_EN and EP\_TX\_EN bits are also set.

**EP\_RX\_EN** — This bit, when set, enables the endpoint for RX transfers.

**EP\_TX\_EN** — This bit, when set, enables the endpoint for TX transfers.

**EP\_STALL** — When set, this bit indicates that the endpoint is stalled. This bit has priority over all other control bits in the EndPoint Enable Register, but it is only valid if EP\_TX\_EN = 1 or EP\_RX\_EN = 1. Any access to this endpoint causes the USB Module to return a STALL handshake. After an endpoint is stalled, it requires intervention from the Host Controller.

**EP\_HSHK** — When set, this bit enables an endpoint to perform handshaking during a transaction to this endpoint. This bit is generally set unless the endpoint is Isochronous.

Table 30-2: Endpoint Enable/Direction Control

Bit Name			Endpoint Enable/Direction Control
EP_CTL_DIS	EP_RX_EN	EP_TX_EN	
X	0	0	Disable endpoint
X	0	1	Enable endpoint for IN(TX) transfers only
X	1	0	Enable endpoint for OUT(RX) transfers only
0	1	1	Enable endpoint for IN, OUT and SETUP transfers.
1	1	1	RESERVE

30.5.2.17. USBPHY Control Register 2 (USBPHY\_CTRL2)

Address: USBC\_BASEADDR+0x0000\_0100

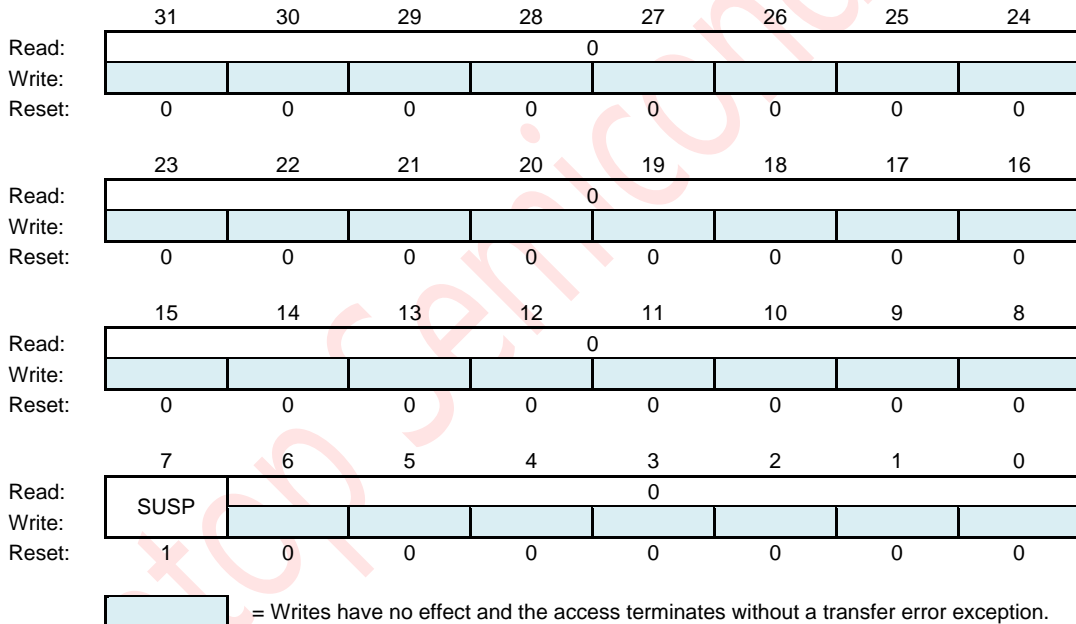
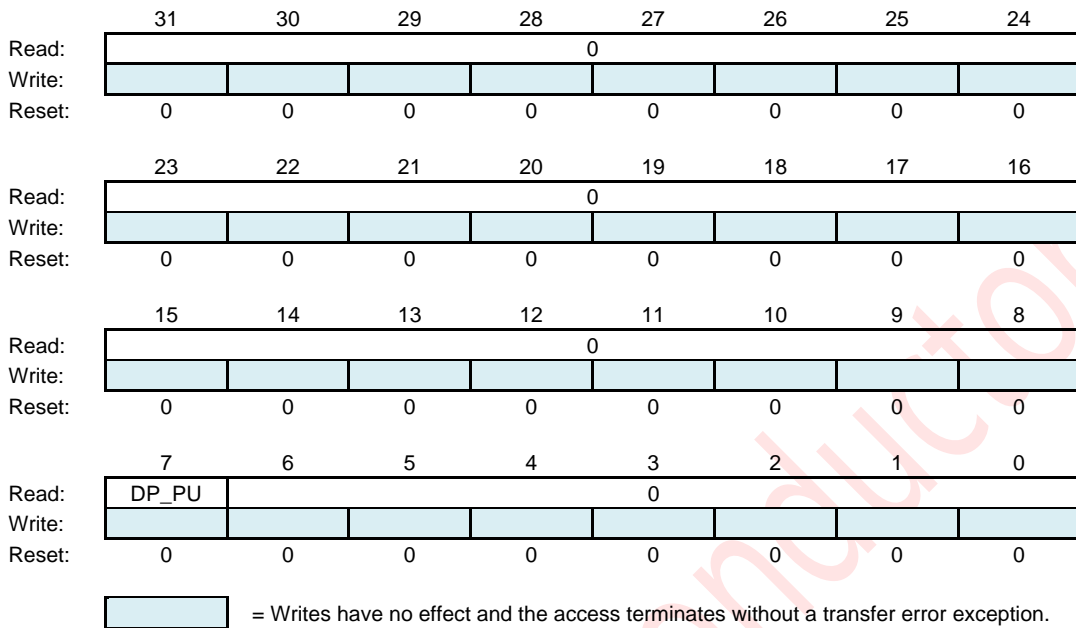


Figure 30-18: USBPHY Control Register 2 (USBPHY\_CTRL2)

- SUSP** — Places the USB transceiver into the suspend state.
- 0 = USB transceiver is not in suspend state.
- 1 = USB transceiver is in suspend state.

**30.5.2.18. USB PHY Observe Register (USB\_PHY\_OBSERVE)**

**Address: USBC\_BASEADDR+0x0000\_0104**



**Figure 30-19: USB PHY Observe Register (USB\_PHY\_OBSERVE)**

**DP\_PU** — Provides observability of the D+ Pull Up signal output from the USB OTG module. This bit is useful when interfacing to an external OTG control module via a serial interface.

0 = D+ pullup is disabled.

1 = D+ pullup is enabled.

30.5.2.19. USBPHY GPIO Register (USB\_PHY\_GPIO)

Address: USBC\_BASEADDR+0x0000\_0108

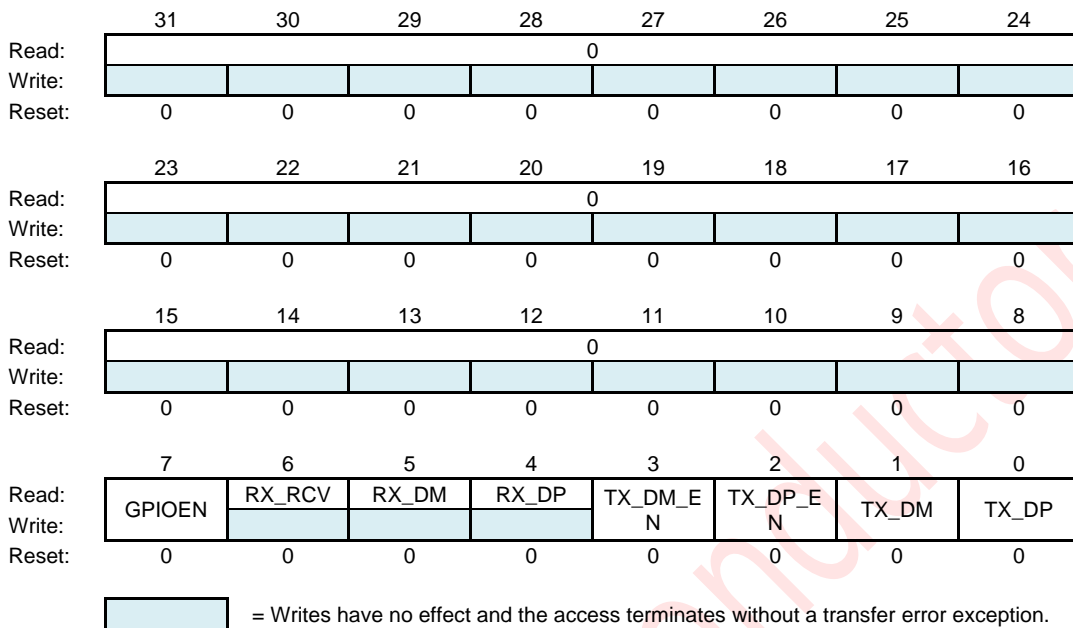


Figure 30-20: USB PHY GPIO Register (USB\_PHY\_GPIO)

**GPIOEN** — USB PHY GPIO Mode control.

- 0 = USB PHY is not in GPIO Mode
- 1 = USB PHY is in GPIO Mode

**RX\_RCV** — USB PHY Differential input data during USBPHY GPIO mode.

**RX\_DM** — USB PHY DM input data during USBPHY GPIO mode.

**RX\_DP** — USB PHY DP input data during USBPHY GPIO mode.

**TX\_DM\_EN** — USB PHY DM output direction control during USBPHY GPIO mode.

- 0 = DM is input direction
- 1 = DM is output direction

**TX\_DP\_EN** — USB PHY DP output direction control during USBPHY GPIO mode.

- 0 = DP is input direction
- 1 = DP is output direction

**TX\_DM** — USB PHY DM output data during USBPHY GPIO mode.

**TX\_DP** — USB PHY DP output data during USBPHY GPIO mode.

30.5.2.20. USB Resume Enable Register (USB\_RESMEN)

Address: USBC\_BASEADDR+0x0000\_010C

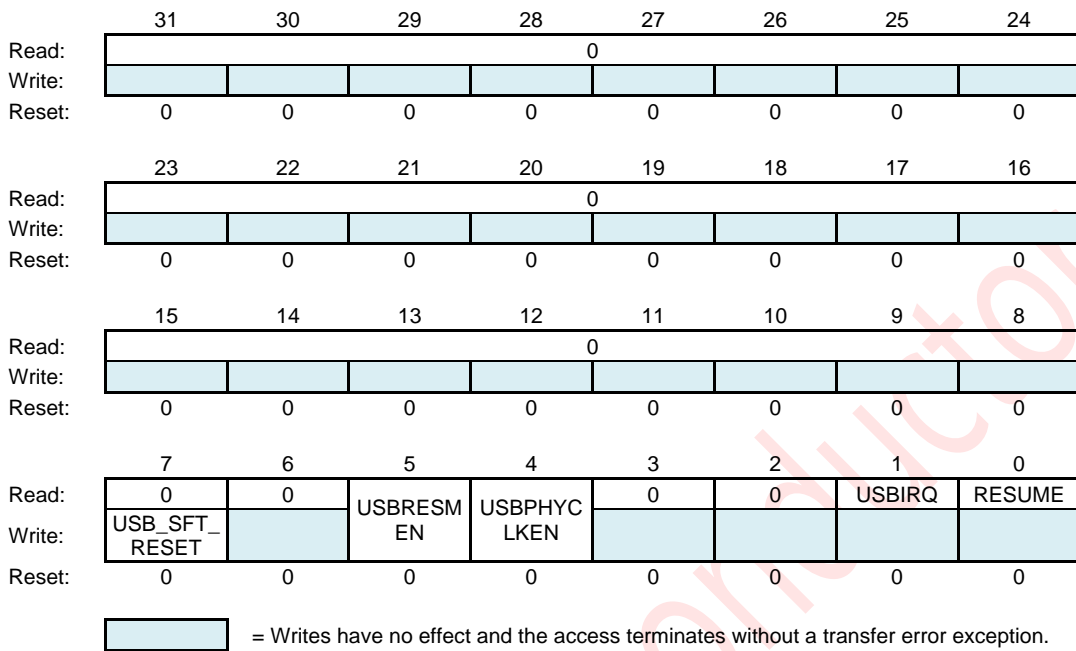


Figure 30-21: USB Resume Enable Register (USB\_RESMEN)

**USB\_SFT\_RESET** — USB Soft Reset control bit.

The USB Module will be reset by writing a one to this bit.

**USBRESMEN** — USB resume wakeup enable control bit

If USBRESMEN is set, and a K-state (resume signaling) is detected on the USB bus, the RESUME bit will become set. This will trigger an asynchronous interrupt that will wake the MCU from stop mode and enable clocks to the USB module.

**USBPHYCLKEN** — USB PHY clock enable control bit.

**USBIRQ** — All enabled interrupt in INT\_STAT register.

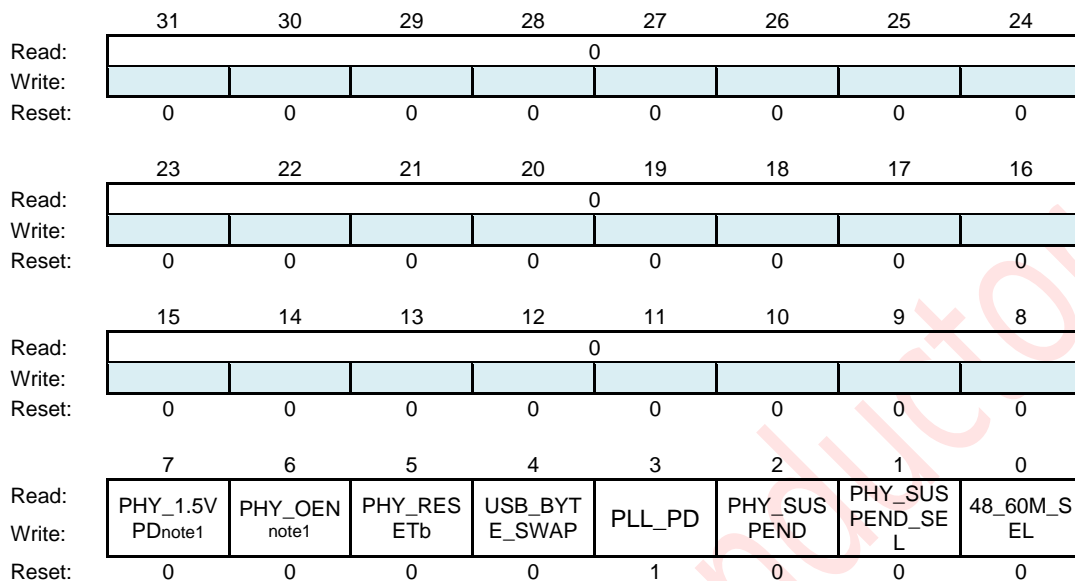
This bit will be set if any one of INT\_STAT is set and the corresponding INT\_ENB is set.

**RESUME** — USB DP/DM resume status bit.

If USBRESMEN is set, and a K-state (resume signaling) is detected on the USB bus, this bit will be set when CPU is waked-up from stop mode and cleared by clearing USBRESMEN bit.

30.5.2.21. USB PHY Control Register3 (USBPHY\_CTRL3)

Address: USBC\_BASEADDR+0x0000\_0118



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

note1 These bits are meaning-less in this chip

Figure 30-22: USB PHY Control Register 3 (USBPHY\_CTRL3)

**PHY\_1.5VPD** — USB PHY core power control.

- 0 = USB PHY core power supply will not be lost.
- 1 = USB PHY core power supply will be lost.

**PHY\_OEN** — USB PHY output isolate control.

- 0 = USB PHY output will not be isolated.
- 1 = USB PHY output will be isolated.

**PHY\_RESE**T**<sub>b</sub>** — USB PHY RESET control.

- 0 = USB PHY will be in RESET state.
- 1 = USB PHY will be NORMAL state.

**USB\_BYTE\_SWAP** — When this bit is set, the data received/transmitted by USB will be swapped when reading from or writing to the system memory.

- 0 = USB Received or Transmit data will not be swapped.
- 1 = USB Received or Transmit data will be swapped.

**PLL\_PD** — USB PHY PLL power down mode control.

- 0 = USB PHY PLL is not in power down state.
- 1 = USB PHY PLL is in power down state.

**PHY\_SUSPEND** — USB PHY Suspend control

- 0 = USB PHY is not in Suspend state.
- 1 = USB PHY is in Suspend state.

**PHY\_SUSPEND\_SEL** — USB PHY Suspend control selection.

0 = USB PHY Suspend is controlled by USBPHY\_CTRL2[SUSP].

1 = USB PHY Suspend is controlled by USBPHY\_CTRL3[PHY\_SUSPEND].

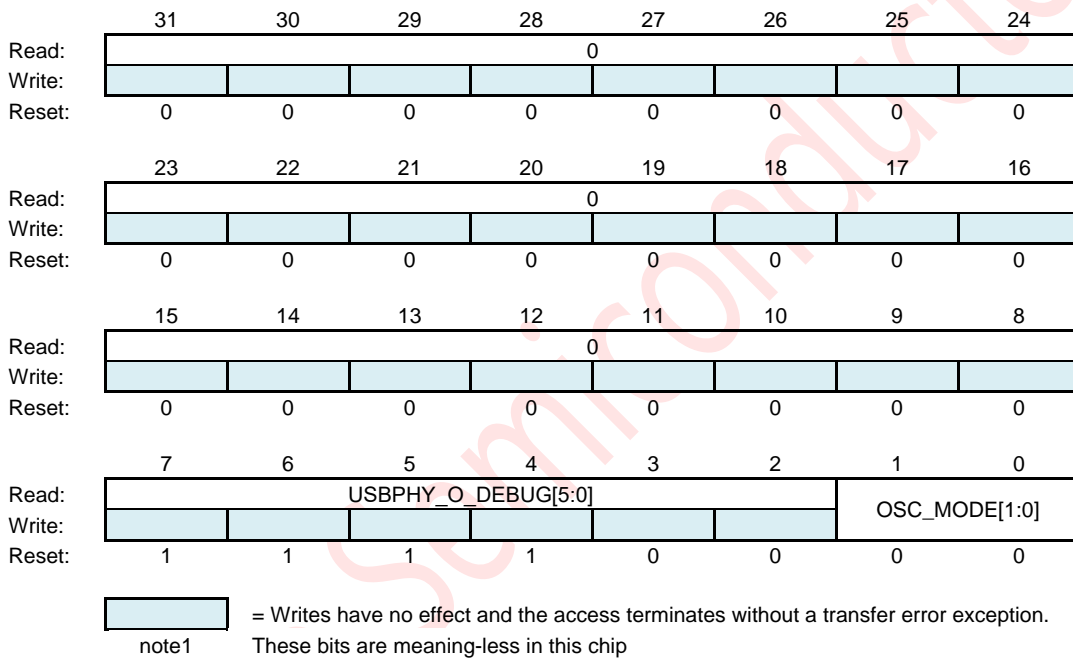
**48\_60M\_SEL** — USB PHY 48/60Mhz clock selection.

0 = 48Mhz is used as USB SIE decode clock.

1 = 60Mhz is used as USB SIE decode clock.

**30.5.2.22. USB PHY Control Register4 (USBPHY\_CTRL4)**

**Address: USBC\_BASEADDR+0x0000\_011C**



**Figure 30-23: USB PHY Control Register 4 (USBPHY\_CTRL4)**

**USBPHY\_O\_DEBUG[5:0]** — USBPHY Output signals are for debug purpose and are meaningless during normal work.

**OSC\_MODE[1:0]** — USB PHY clock mode selection.

**Table 30-3: USB PHY Oscillator Mode Selection**

OSC_MODE[1:0]	Oscillator Mode Selection
00	Auto detection oscillator
01	Auto detection oscillator (fast simulation mode)
10	Selection internal oscillator
11	Selection external oscillator



### 30.6. Function Description

The USB-FS 2.0 full-speed/low-speed module communicates with the processor core through status registers, control registers, and data structures in memory.

#### 30.6.1. Data Structure

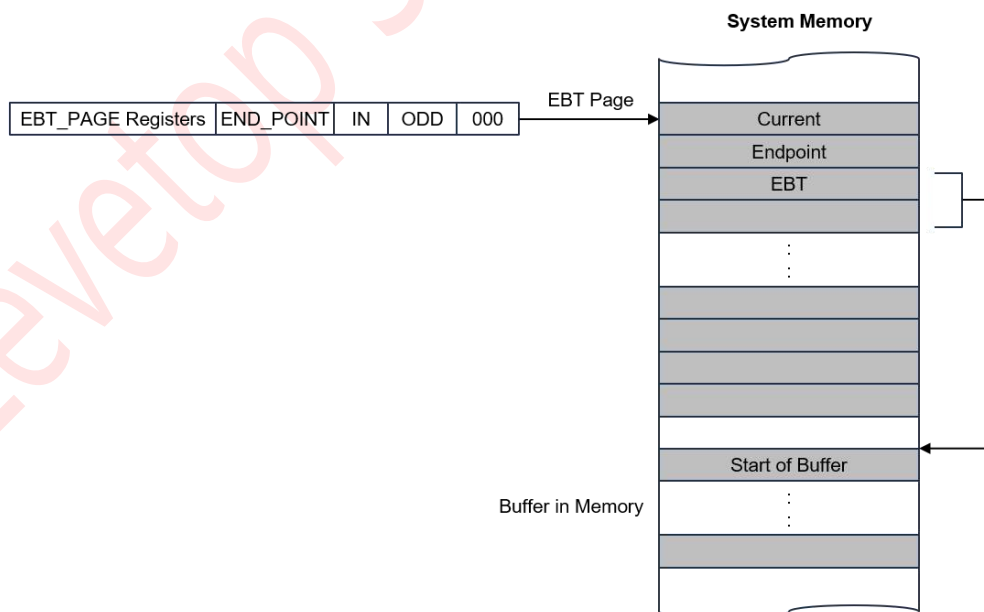
The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. To efficiently manage USB endpoint communications, the USB module implements an Endpoint Buffer Table (EBT) in system memory. See **Figure 30-24**.

#### 30.6.2. Endpoint Buffer Table

To efficiently manage USB endpoint communications the USB module implements a Buffer Descriptor Table (EBT) in system memory. The EBT resides on a 512 byte boundary in system memory and is pointed to by the EBT Page Registers. Every endpoint direction requires two eight-byte Endpoint Buffer Table entries.

Therefore, a system with 16 fully bidirectional endpoints would require 512 bytes of system memory to implement the EBT. The two Endpoint Buffer Table (EBT) entries allows for an EVEN EB entry and ODD EB entry for each endpoint direction. This allows the microprocessor to process one EB entry while the USB module is processing the other EB entry. Double buffering EB entries in this way allows the USB SIE to easily transfer data at the maximum throughput provided by USB.

The software API intelligently manages buffers for the USB SIE by updating the EBT when needed. This allows the USB SIE to efficiently manage data transmission and reception, while the microprocessor performs communication overhead processing and other function dependent applications. Because the buffers are shared between the microprocessor and the USB module, a simple semaphore mechanism is used to distinguish who is allowed to update the EBT and buffers in system memory. A semaphore bit, the OWN bit, is cleared to 0 when the EB entry is owned by the microprocessor. The microprocessor is allowed read and write access to the EB entry and the buffer in system memory when the OWN bit is 0. When the OWN bit is set to 1, the EB entry and the buffer in system memory are owned by the USB module. The USB module now has full read and write access and the microprocessor should not modify the EB entry or its corresponding data buffer. The EB entry also contains indirect address pointers to where the actual buffer resides in system memory. This indirect address mechanism is shown in the following diagram.



**Figure 30-24: Endpoint Buffer Table**

**30.6.3. Rx vs. Tx As A USB Target Device**

The centric nomenclature is used to describe the direction of the data transfer between the USB SIE core and the USB Host:

Rx (or receive): describes transfers that move data from the USB to memory.

Tx (or transmit): describes transfers that move data from memory to the USB.

The following table shows how the data direction corresponds to the USB token type in host and target device applications.

**Table 30-4: Data Direction for USB Target Device**

	RX	TX
Device	OUT or Setup	IN

**30.6.4. Addressing Endpoint Buffer Table Entries**

An understanding of the addressing mechanism of the Endpoint Buffer Table is useful when accessing endpoint data via the USB module or microprocessor. Some points of interest are:

- The Endpoint Buffer Table occupies up to 512 bytes of system memory.
- 16 bidirectional endpoints can be supported with a full EBT of 512 bytes.
- 16 bytes are needed for each USB endpoint direction.
- Applications with less than 16 endpoints require less RAM to implement the EBT.
- The EBT Page Registers point to the starting location of the EBT.
- The EBT must be located on a 512-byte boundary in system memory.
- All enabled TX and RX endpoint EB entries are indexed into the EBT to allow easy access via the USB module or CPU.

When a USB token on an enabled endpoint is received, the USB module uses its integrated DMA controller to interrogate the EBT. The USB SIE reads the corresponding endpoint EB entry to determine if it owns the EB entry and corresponding buffer in system memory.

To compute the entry point in the EBT, the EBT\_PAGE registers is concatenated with the current endpoint and the TX and ODD fields to form a 32-bits address. This address mechanism is shown in the following diagrams:

**Table 30-5: EBT Address Calculation Fields**

Field	Description
EBT_PAGE	EBT_PAGE registers in the Control Register Block
END_POINT	END POINT field from the USB TOKEN
TX	1 for an TX transmit transfers and 0 for an RX receive transfers
ODD	This bit is maintained within the USB SIE. It corresponds to the buffer currently in use.
	The buffers are used in a ping-pong fashion.

**30.6.5. Endpoint Buffer Table Formats**

The Endpoint Buffer Table (EBT) provides endpoint control information for the USB module and microprocessor. The Endpoint Buffer Tables have different meaning based on whether it is the USB module or microprocessor reading the EB entry in memory.

The USB SIE Controller uses the data stored in the EB entries to determine:

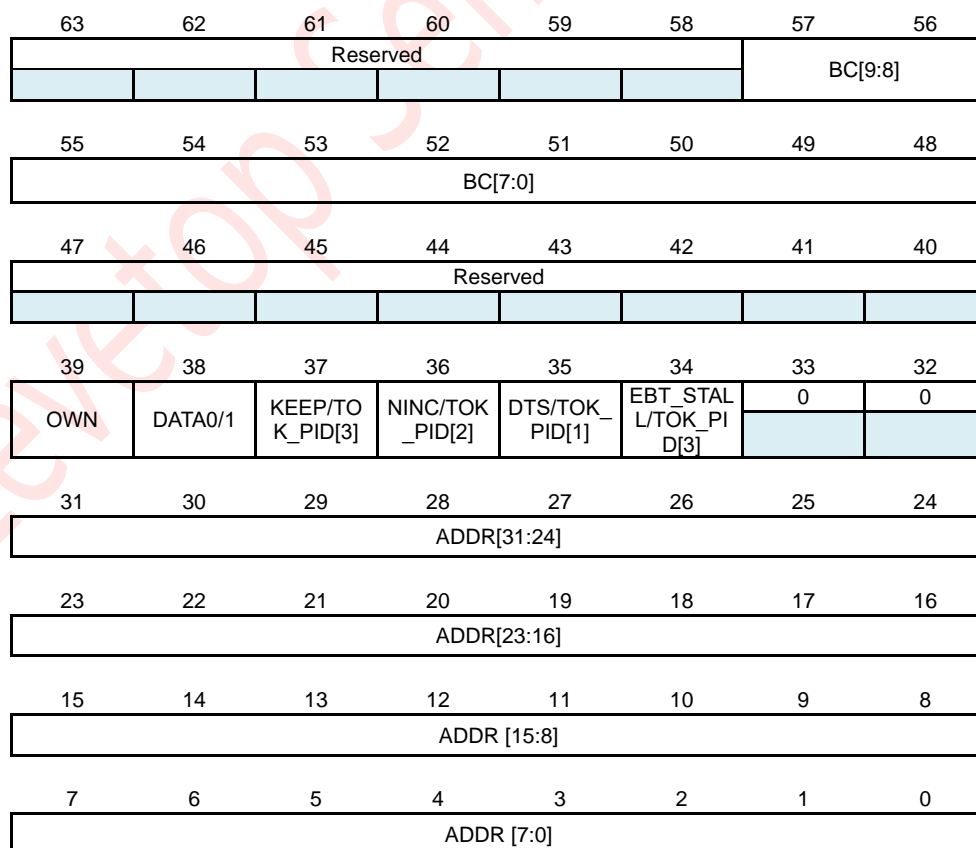
- Who owns the buffer in system memory
- Data0 or Data1 PID
- Release Own upon packet completion
- No address increment (FIFO Mode)
- Data toggle synchronization enable
- How much data is to be transmitted or received
- Where the buffer resides in system memory

While the microprocessor uses the data stored in the EB entries to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- The received TOKEN PID
- How much data was transmitted or received
- Where the buffer resides in system memory

The format for the EB entry is shown in the following figure.

**Table 30-6: Endpoint Buffer Table Byte Format**



**Table 30-7: Endpoint Buffer Table Byte Fields**

Field	Description
OWN	<p><b>OWN</b> — This OWN bit determines who currently owns the buffer. Except when KEEP = 1, the USB SIE writes a 0 to this bit when it has completed a token. The USB module ignores all other fields in the EB entry when OWN = 0. Once the EB entry has been assigned to the USB module (OWN = 1), the MCU should not change it in any way. This byte of the EB entry should always be the last byte the MCU (firmware) updates when it initializes a EB entry. Although the hardware will not block the MCU from accessing the EB entry while owned by the USB SIE, doing so may cause undefined behavior and is generally not recommended.</p> <p>0: The MCU has exclusive access to the entire EB entry 1: The USB module has exclusive access to the EB entry</p>
DATA0/1	<p><b>Data Toggle</b> — This bit defines if a DATA0 field (DATA0/1 = 0) or a DATA1 (DATA0/1 = 1) field was transmitted or received. It is unchanged by the USB module.</p> <p>0: Data 0 packet 1: Data 1 packet</p>
KEEP/ TOK_PID[3]	<p><b>KEEP / EB entry Token PID [3]</b> — Typically this bit is set (that is, 1) with ISO endpoints feeding a FIFO. The microprocessor is not informed that a token has been processed, the data is simply transferred to or from the FIFO. If KEEP is set, normally the NINC bit is also set to prevent address increment.</p> <p>0: Bit 3 of the current token PID is written back in to the EB entry by the USB SIE. Allows the USB SIE to release the EB entry when a token has been processed. 1: This bit is unchanged by the USB SIE. If the OWN bit also is set, the EB entry remains owned by the USB SIE forever.</p>
NINC/ TOK_PID[2]	<p><b>No Increment / EB entry Token PID [2]</b> — The No Increment (NINC) bit disables the DMA engine address increment. This forces the DMA engine to read or write from the same address. This is useful for endpoints when data needs to be read from or written to a single location such as a FIFO. Typically this bit is set with the KEEP bit for ISO endpoints that are interfacing to a FIFO.</p> <p>0: The USB SIE writes bit 2 of the current token PID to the EB entry. 1: This bit is unchanged by the USB SIE.</p>
DTS/ TOK_PID[1]	<p><b>Data Toggle Synchronization / EB entry Token PID [1]</b> — This bit enables data toggle synchronization.</p> <p>Setting this bit enables the USB SIE to perform Data Toggle Synchronization.</p> <ul style="list-style-type: none"> <li>• If KEEP = 0, bit 1 of the current token PID is written back to the EB entry.</li> <li>• If KEEP = 1, this bit is unchanged by the USB SIE.</li> </ul> <p>0: No data toggle synchronization is performed. 1: Data toggle synchronization is performed.</p>
EBT_STALL/ TOK_PID[0]	<p><b>EBT Stall / EB entry Token PID [0]</b> — Setting this bit will cause the USB module to issue a STALL handshake if a token is received by the SIE that would use the EBT in this location. The EBT is not consumed by the SIE (the OWN bit remains and the rest of the EB entry is unchanged) when the EBT_STALL bit is set.</p> <ul style="list-style-type: none"> <li>• If KEEP = 0, bit 0 of the current token PID is written back to the EB entry.</li> <li>• If KEEP = 1, this bit is unchanged by the USB SIE.</li> </ul> <p>0: EBT stall is disabled 1: USB will issue a STALL handshake if a token is received by the SIE that would use the EBT in this location</p>

Field	Description
TOK_PID[n]	Bits [37:34] can also represent the current token PID. The current token PID is written back in to the EB entry by the USB SIE when a transfer completes. The values written back are the token PID values from the USB specification: <ul style="list-style-type: none"> <li>• 0x1 for an OUT token.</li> <li>• 0x9 for an IN token.</li> <li>• 0xD for a SETUP token.</li> </ul>
BC[9:0]	<b>Byte Count</b> — The Byte Count bits represent the 10-bits byte count. The USB module serial interface engine (SIE) will change this field upon the completion of a RX transfer with the byte count of the data received.
ADDR[31:0]	The Address bits represent the 32-bits buffer address in system memory. These bits are unchanged by the USB SIE.

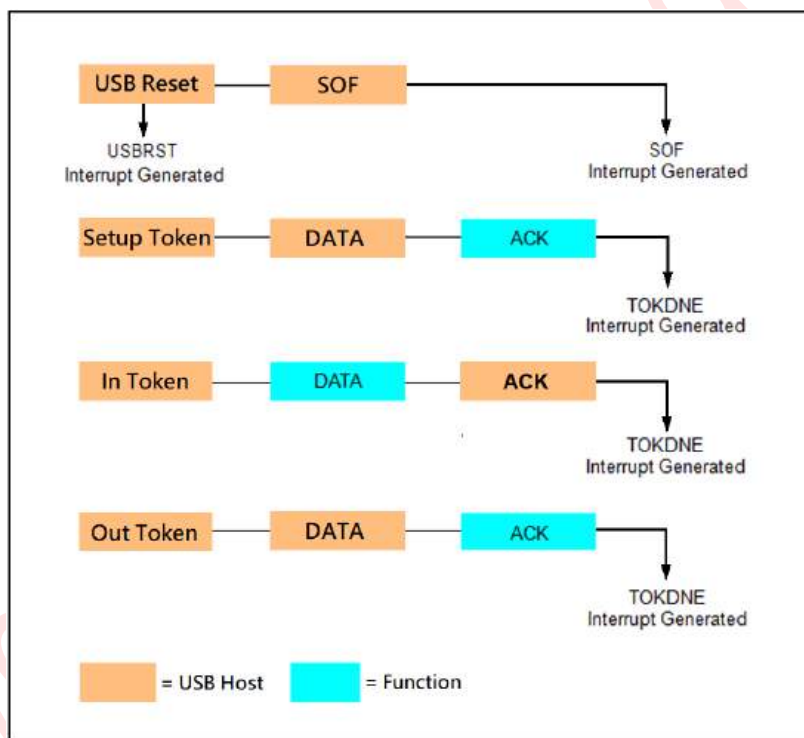
**30.6.6. USB Transaction**

When the USB SIE transmits or receives data, it computes the EBT address using the address generation shown in "Addressing Endpoint Buffer Table Entries" table.

If OWN = 1, the following process occurs:

1. The USB SIE reads the EBT.
2. The SIE transfers the data via the DMA to or from the buffer pointed to by the ADDR field of the EB entry.
3. When the TOKEN is complete, the USB SIE updates the EBT and, if KEEP = 0, changes the OWN bit to 0.
4. The STAT register is updated and the TOK\_DNE interrupt is set.
5. When the microprocessor processes the TOK\_DNE interrupt, it reads from the status register all the information needed to process the endpoint.
6. At this point, the microprocessor allocates a new EB entry so additional USB data can be transmitted or received for that endpoint, and then processes the last EB entry.

The following figure shows a timeline of how a typical USB token is processed after the EBT is read and OWN = 1.



**Figure 30-25: USB Token Transaction**

The USB has two sources for the DMA overrun error:

**Memory Latency**

The memory latency on the DMA interface may be too high and cause the receive FIFO to overflow. This is predominantly a hardware performance issue, usually caused by transient memory access issues.

**Oversized Packets**

The packet received may be larger than the negotiated MaxPacket size. Typically, this is caused by a software bug. For DMA overrun errors due to oversized data packets, the USB specification is ambiguous. It assumes correct software drivers on both sides. NAKing the packet can result in retransmission of the already oversized packet data. Therefore, in response to oversized packets, the USB core continues ACKing the packet for non-isochronous transfers.

**Table 30-8: USB Responses to DMA Overrun Errors**

Errors due to Memory Latency	Errors due to Oversized Packets
Non-Acknowledgment (NAK) or Bus Timeout (BTO) — See bit 4 in "Error Interrupt Status Register (ERR_STAT)" as appropriate for the class of transaction.	Continues acknowledging (ACKing) the packet for Nonisochronous transfers.
—	The data written to memory is clipped to the MaxPacket size so as not to corrupt system memory.
The DMA_ERR bit is set in the ERR_STAT register. Depending on the values of the INT_ENB and ERR_ENB register, the core may assert an interrupt to notify the processor of the DMA error.	Asserts the DMA_ERR bit of the ERR_STAT register (which could trigger an interrupt) and a TOK_DNE interrupt fires.  <b>Note:</b> The TOK_PID field of the EBT is not 1111 because the DMA_ERR is not due to latency.
The EBT is not written back nor is the TOK_DNE interrupt triggered because it is assumed that a second attempt is queued and will succeed in the future.	The packet length field written back to the EBT is the MaxPacket value that represents the length of the clipped data actually written to memory.
From here, the software can decide an appropriate course of action for future transactions such as stalling the endpoint, canceling the transfer, disabling the endpoint, etc.	

## 31. Analog-to-Digital Converter (ADC)

### 31.1. Introduction

The 12-bits ADC is a successive approximation analog-to-digital converter. It has up to 9 channels allowing it to measure signals from 8 external and 2 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 12-bits x 8 depth 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.

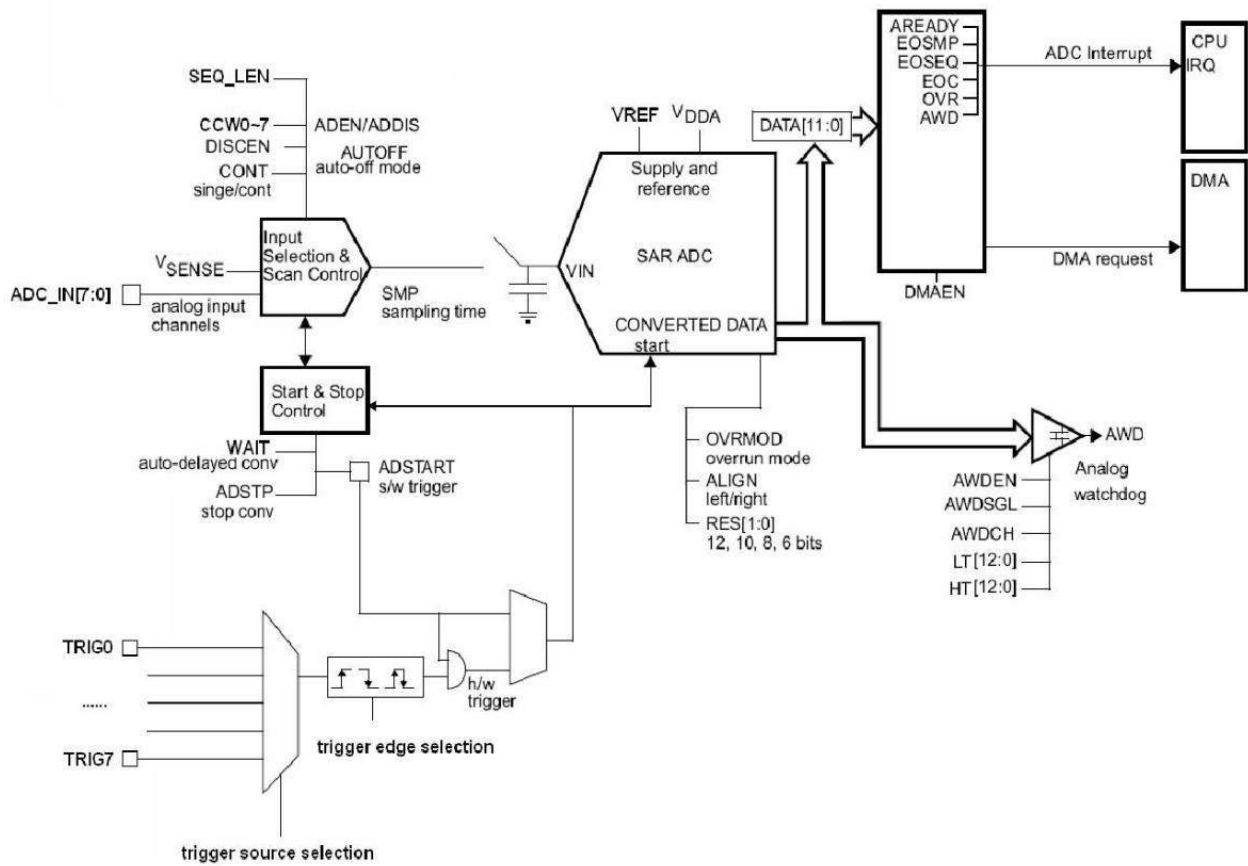
An efficient low power mode is implemented to allow very low consumption at low frequency.

### 31.2. ADC Main Features

- High Performance
  - 12-bits, 10-bits, 8-bits or 6-bits configurable resolution
  - ADC conversion time: 1.0  $\mu$ s for 12-bits resolution (1 MHz), 0.88  $\mu$ s conversion time for 10-bits 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  $\mu$ s 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



**31.3. ADC Functional Description**



**Figure 31-1: ADC Block Diagram**

**31.3.1. 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 31-2**, the ADC needs a stabilization time of  $t_{STAB}$ (~2.0  $\mu$ s) 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.

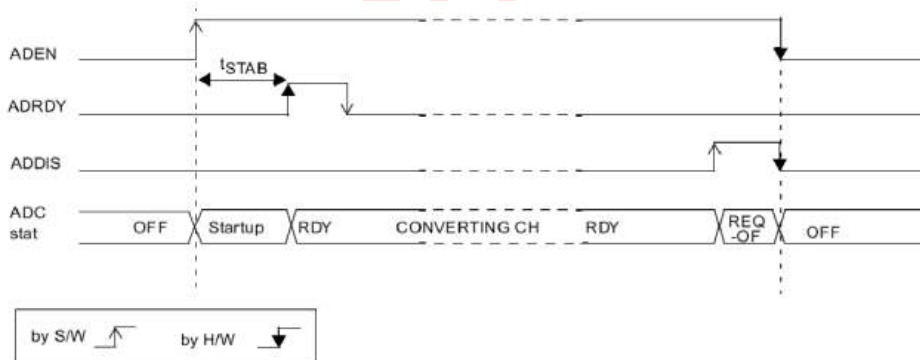
Conversion can then start either by setting ADSTART = 1 or when an external trigger event occurs if triggers are enabled.

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 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).

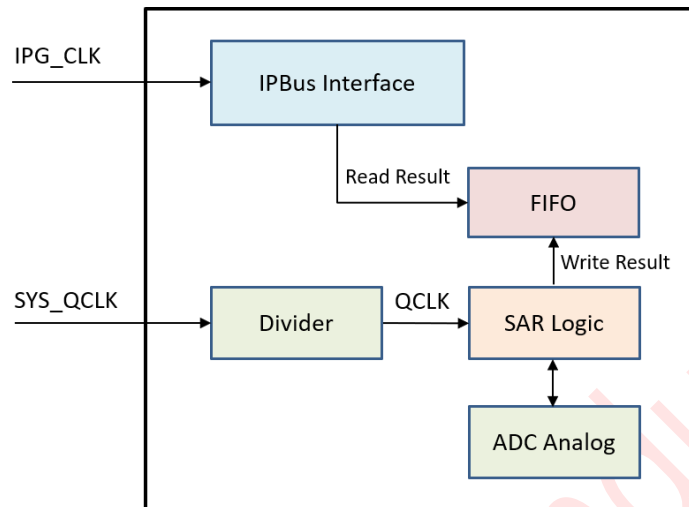


**Figure 31-2: Enabling/Disabling the ADC**

**Note:** In auto-off mode (AUTOFF = 1) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.

### 31.3.2. ADC Clock

The ADC has a dual clock-domain architecture, as show in **Figure 31-3**, this has the advantage of reaching the maximum ADC clock frequency whatever the IPG clock scheme selected.



**Figure 31-3: ADC Clock Scheme**

### 31.3.3. 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 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).

### 31.3.4. 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 31-1**.

**Table 31-1: 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

**31.3.5. 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.

**31.3.6. 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.

### 31.3.7. 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.

### 31.3.8. 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

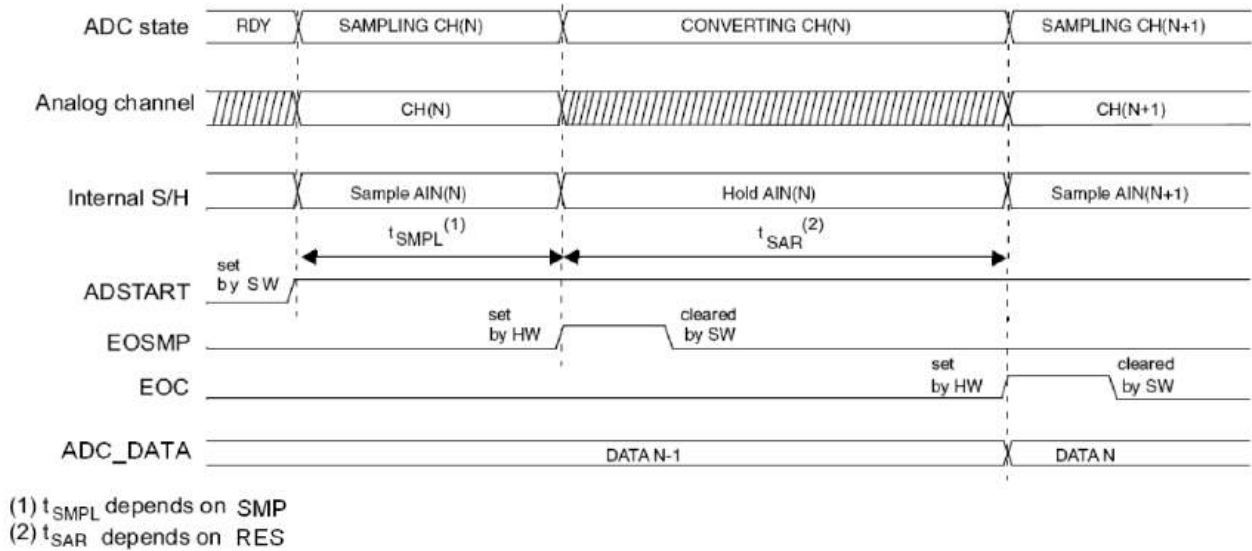
**Note:**

- 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.

**31.3.9. 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:

$$t_{ADC} = t_{SMPL} + t_{SAR} = [ 4|_{min} + 12|_{12\text{-bits}} ] \times t_{QCLK} = 1\mu\text{s} |_{min} \text{ (for } f_{QCLK} = 16 \text{ MHz)}$$



**Figure 31-4: Analog to Digital Conversion Time**

### 31.3.10. 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 31-5: Stopping an Ongoing Conversion**

## 31.4. Conversion On External Trigger And Trigger Polarity

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 31-2:** Provides the correspondence between the TRIGMODE values and the trigger polarity.

**Table 31-2: Configuring the Trigger Polarity.**

TRIGMODE[2:0]	Source
3'b000	Trigger detection is 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 31-3** 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 31-3: Configuring the Trigger Polarity**

TRIGSCR[2:0]	Name	Source
3'b000	TRG0	
3'b001	TRG1	
3'b010	TRG2	
3'b011	TRG3	
3'b100	TRG4	
3'b101	TRG5	
3'b110	TRG6	
3'b111	TRG7	

**Note:** The trigger selection can be changed only when the ADC is not converting (ADSTART = 0)



### 31.4.1. 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  
Conversion on external trigger and trigger polarity (TRIGMODE, TRIGSCR)
  - 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.

### 31.4.2. Programmable Resolution (RES) - Fast Conversion Mode

It is possible to obtain faster conversion times (t<sub>SAR</sub>) 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.

### 31.4.3. 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 writing 1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC\_IER register.

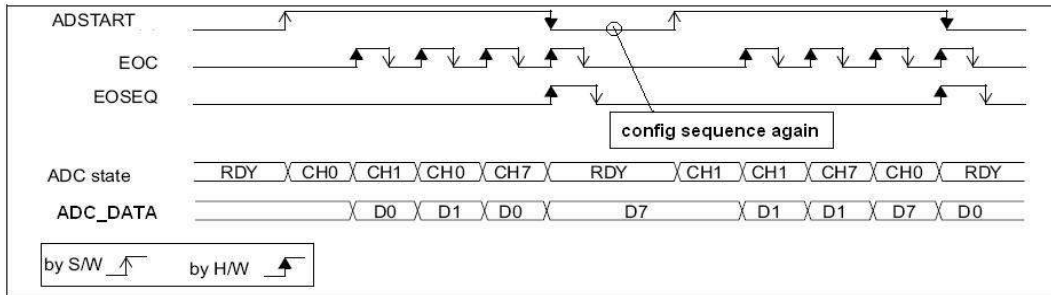
### 31.4.4. End Of Conversion Sequence (Eoseq Flag)

The ADC notifies the application of each end of sequence (EOSEQ) event.

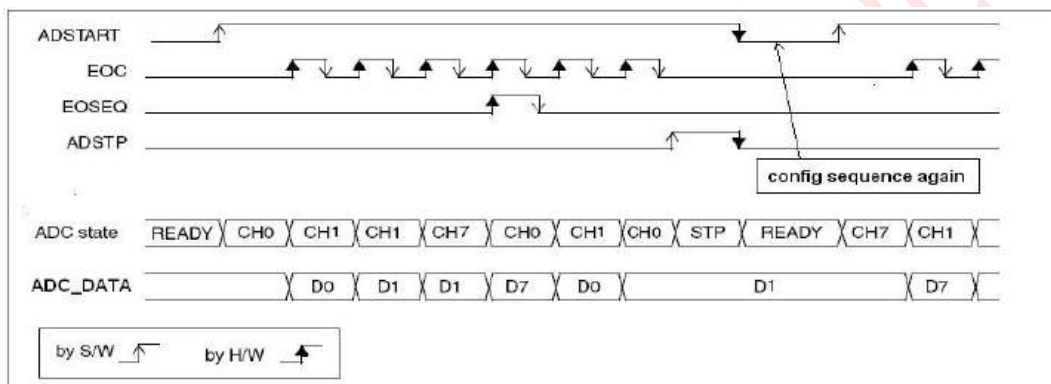
The ADC sets the EOSEQ flag in the ADC\_ISR register as soon as the last data result of a conversion sequence is available in the FIFO. An interrupt can be generated if the EOSEQIE bit is set in the ADC\_IER register. The EOSEQ flag is cleared by software by writing 1 to it.

**31.4.5. Example Timing Diagrams**

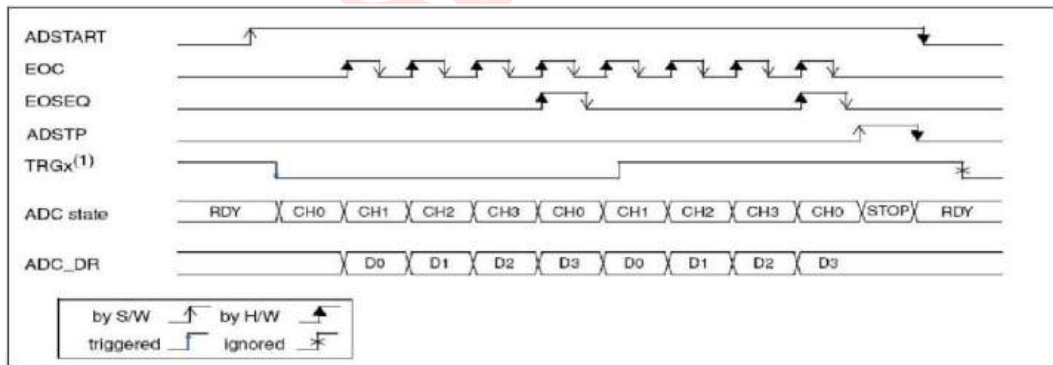
This section shows the single and continuous modes in hardware or software triggers condition.



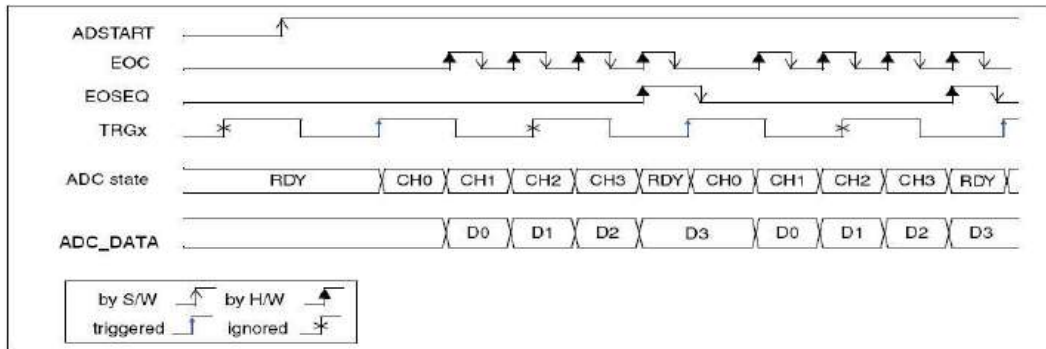
**Figure 31-6: Single conversions of a sequence, software trigger**



**Figure 31-7: Continuous Conversion of a Sequence, Software Trigger**



**Figure 31-8: Single Conversions of a Sequence, Hardware Trigger**



**Figure 31-9: Continuous Conversions of a Sequence, Hardware Trigger**

Levetop Semiconductor

### 31.5. Data Management

#### 31.5.1. 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 13-bits 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 **Table 31-4**.

**Table 31-4: Data Alignment and Resolution**

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			....																				
	0x3			....																				
1	0x0			....	Data[11:0]																			
	0x1			....	Data[9:0]																			
	0x2			....											Data[7:0]									
	0x3			....											Data[5:0]									

The FIFO supports byte, half-word and word read, but the address offset should be always 0x4C. For different data alignments and resolutions, users should note:

- If read by word, then the data format is as data[31:0] as **Table 31-4** shows
- if read by half word, then the data format is as data[15:0] as **Table 31-4** shows
- If read by byte when the data is longer than 8-bits, then the high byte is first read out, and the low byte should read again.
- If read by byte when the data is not longer than 8-bits, then the data format is as data[7:0] as **Table 31-4** shows.

#### 31.5.2. 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.

### **31.5.3. 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.

### **31.5.4. 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.

### **31.5.5. 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. This 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.

## 31.6. Low Power Features

### 31.6.1. 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.

### 31.6.2. 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

### 31.7. Analog Window Watchdog (AWD)

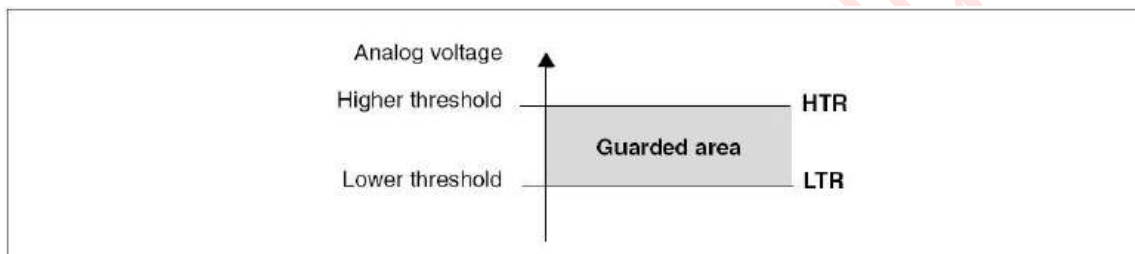
The AWD analog watchdog feature is enabled by setting the AWDEN bit in the ADC\_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels remain within a configured voltage range (window) as shown in **Figure 31-10**.

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 12-bits (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 12-bits raw converted data (left aligned).

**Table 31-5** 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 31-10: Analog Watchdog Guarded Area**

**Table 31-5: Analog Watchdog Channel Selection**

Channels Guarded by the Analog Watchdog	AWDSGL Bit	AWDEN Bit
None	X	0
All Channel	0	1
Single <sup>(1)</sup> Channel	1	1

**Note (1)** : Selected by the AWDCH

### 31.8. 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.

### 31.9. 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 31-6: ADC Interrupts**

Interrupt Event	Event Flag	Enable Control Bit
ADC Ready	ADRDY	ADRDYIE
End of Conversion	EOC	1EOCIE
End of Sequence of Conversions	EOSEQ	1EOSEQIE
Analog Watchdog Status bit is Set	AWD	AWDIE
End of Sampling Phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE



### 31.10. Memory Map and Registers

The ADC Module memory map is shown in **Table 31-7**. The ADC base address is **0x4011\_0000**. This subsection describes the memory map and register structure of ADC.

#### 31.10.1. Memory Map

Refer to **Table 31-7** for a description of the QADC memory map.

**Table 31-7: ADC Module Memory Map**

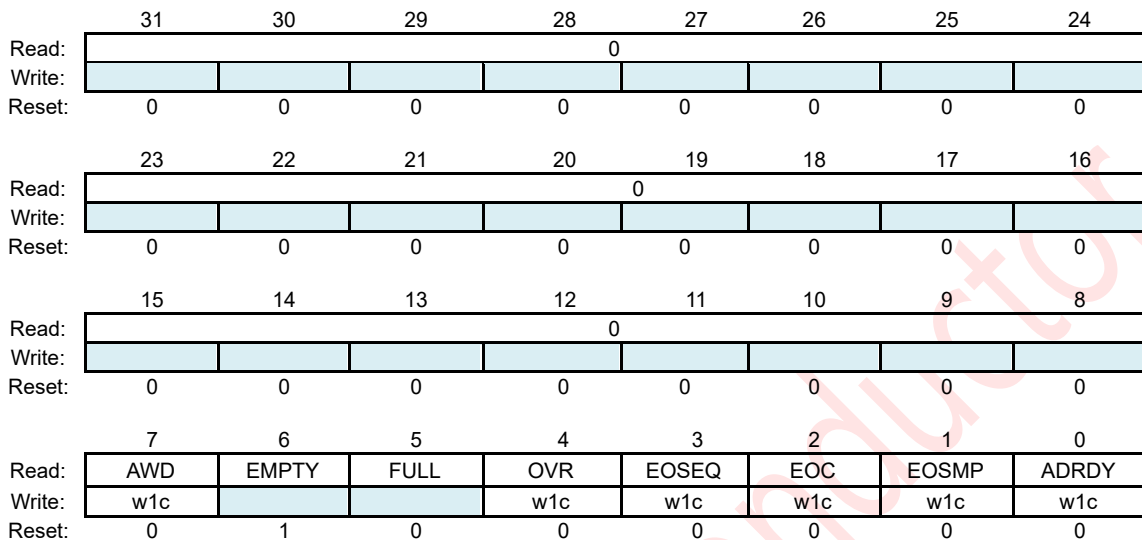
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_ISR																								AWD	Empty	FULL	OVR	ESEQ	EOC	EOSMP	ADRDY	
0x04	ADC_IER																								AWDIE			OVRIE	ESEQIE	EOCIE	EOSMPIE	ADRDYIE	
0x08	ADC_CR																												ADSTP	ADSTART	ADDIS	ADEN	
0x0C	ADC_CFG R1	DIFF	OVRMOD				SEQLEN[2:0]			DISCEN	AUTOFF	WAIT	CONT		TRIGSCR[2:0]					TRIGMODE[2:0]		ALIGN	RES[1:0]									DMAEN	
0x10	ADC_CFG R2																					QPR[3:0]									STCNT[7:0]		
0x14	ADC_SMP R																														SMP[7:0]		
0x18	ADC_WD G																								AWDEN	AWDSGL					AWDCH[3:0]		
0x1C	ADC_TR											HT[11:0]																			LT[11:0]		
0x2C	ADC_CHS ELR1						CCW3[3:0]								CCW2[3:0]									CCW1[3:0]							CCW0[3:0]		
0x30	ADC_CHS ELR2						CCW7[3:0]								CCW6[3:0]									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.

**31.10.2. Register Descriptions**

**31.10.2.1. ADC Interrupt And Status Register (ADC\_ISR)**

**Address: ADC\_BASEADDR+0x0000\_0000**



 = Writes have no effect and the access terminates without a transfer error exception.  
w1c = .Write 1 to the bit will clear it

**Figure 31-11: 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)

**ADDRDY** — ADC ready

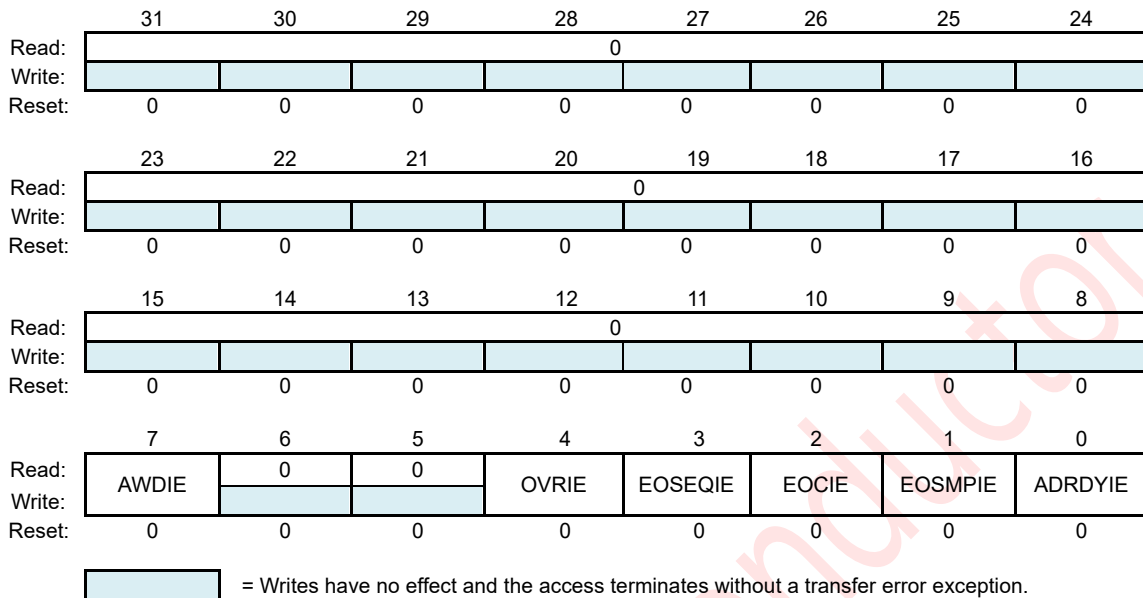
This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests. It is cleared by software writing 1 to it.

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)

**31.10.2.2. ADC Interrupt Enable Register (ADC\_IER)**

**Address: ADC\_BASEADDR+0x0000\_0004**



**Figure 31-12: 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 is enabled
- 0 = Analog watchdog interrupt is disabled

**OVRIE** — Overrun interrupt enable

This bit is set and cleared by software to enable/disable the overrun interrupt.

- 1 = Overrun interrupt is enabled. An interrupt is generated when the OVR bit is set.
- 0 = Overrun interrupt is 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 is enabled. An interrupt is generated when the EOSEQ bit is set.
- 0 = EOSEQ interrupt is 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 is enabled. An interrupt is generated when the EOC bit is set.
- 0 = EOC interrupt is 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 is enabled. An interrupt is generated when the EOSMP bit is set.
- 0 = EOSMP interrupt is disabled.

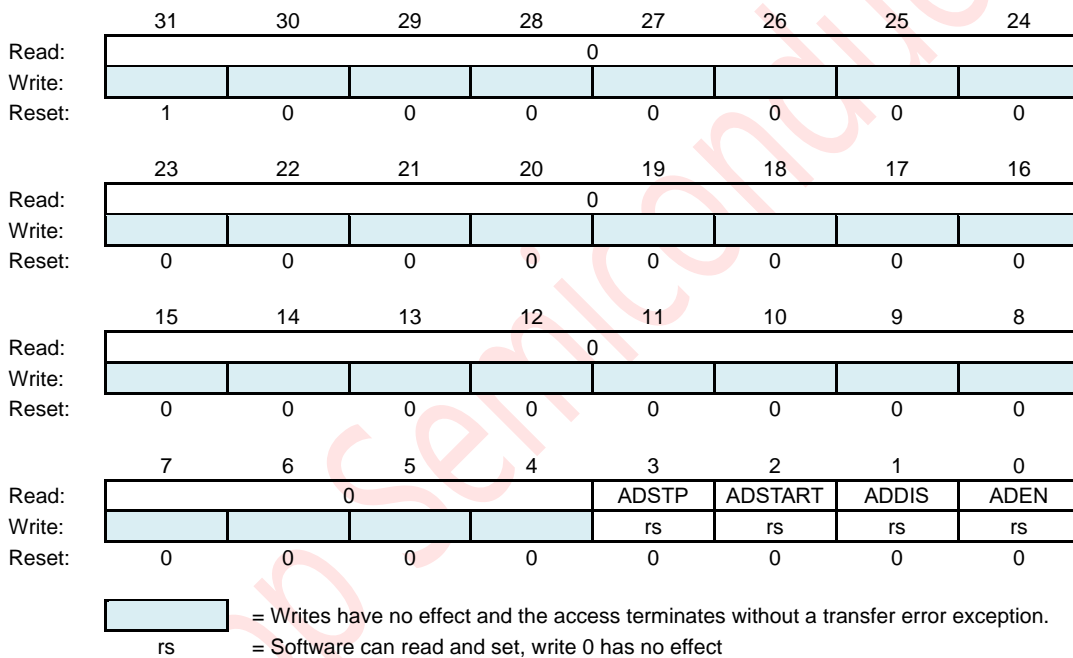
**ADRDYIE** — ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

- 1 = ADRDY interrupt is enabled. An interrupt is generated when the ADRDY bit is set.
- 0 = ADRDY interrupt is disabled.

**31.10.2.3. ADC Control Register (ADC\_CR)**

Address: ADC\_BASEADDR+0x0000\_0008



**Figure 31-13: 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.

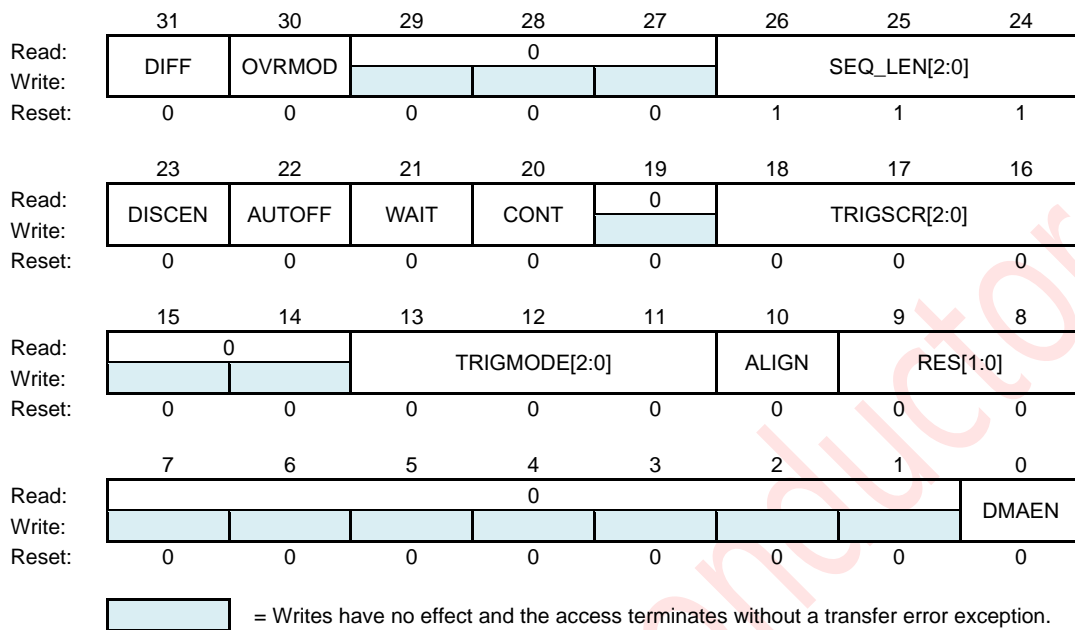
1 = Write 1 to enable the ADC.

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)

**31.10.2.4. ADC Configuration Register 1 (ADC\_CFGR1)**

**Address: ADC\_BASEADDR+0x0000\_000C**



**Figure 31-14: 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 configures 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 is enabled
- 0 = Discontinuous mode is disabled

**AUTOFF** — Auto-off mode

This bit is set and cleared by software to enable/disable auto-off mode.

- 1 = Auto-off mode is enabled
- 0 = Auto-off mode is 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 31-3**.

**TRIGMODE[2:0]** — Trigger mode select

These bits are used to select software trigger mode or external trigger polarity, see **Table 31-2**.

**ALIGN** — Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to **Table 31-4**.

- 1 = Right alignment
- 0 = Left alignment

**RES[1:0]** — Data resolution

These bits are written by software to select the resolution of the conversion. Refer to **Table 31-4**.

**DMAEN** — Direct memory access enable

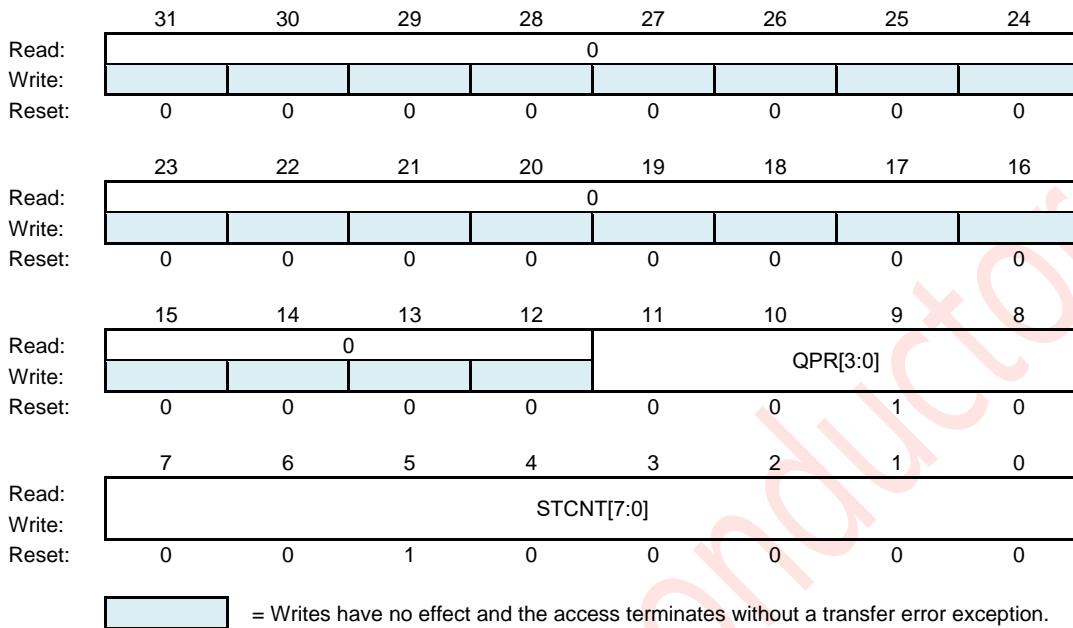
This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA controller to manage automatically the converted data.

- 1 = DMA is enabled
- 0 = DMA is disabled



**31.10.2.5. ADC Configuration Register 2 (ADC\_CFGR2)**

**Address: ADC\_BASEADDR+0x0000\_0010**



**Figure 31-15: 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:

$$FQCLK = F_{sys\_QCLK} / (QPR[3:0] + 1)$$

where:

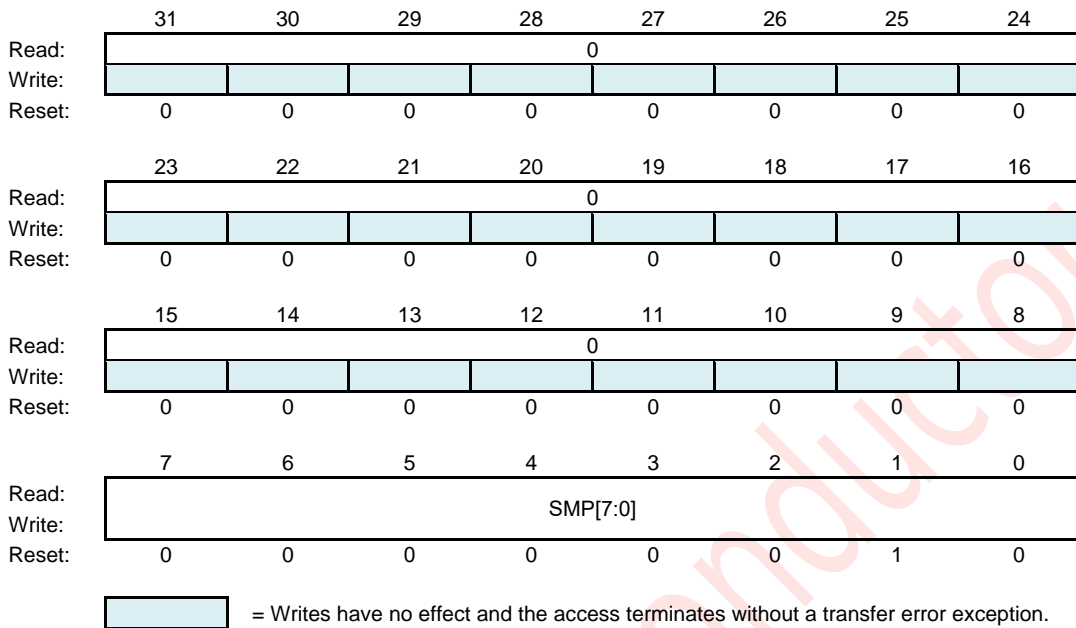
$$0 < QPR[3:0] \leq 15 \text{ and the value } 1 \text{ is not allowed.}$$

**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 users should set the STCNT[7:0] = 2000/(1000/16) = 32.

**31.10.2.6. ADC Sampling Time Register (ADC\_SMPR)**

**Address: ADC\_BASEADDR+0x0000\_0014**



**Figure 31-16: 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] = 0x2 means the sample time is 4 QCLKs

31.10.2.7. ADC Watch Dog Register (ADC\_WDG)

Address: ADC\_BASEADDR+0x0000\_0018

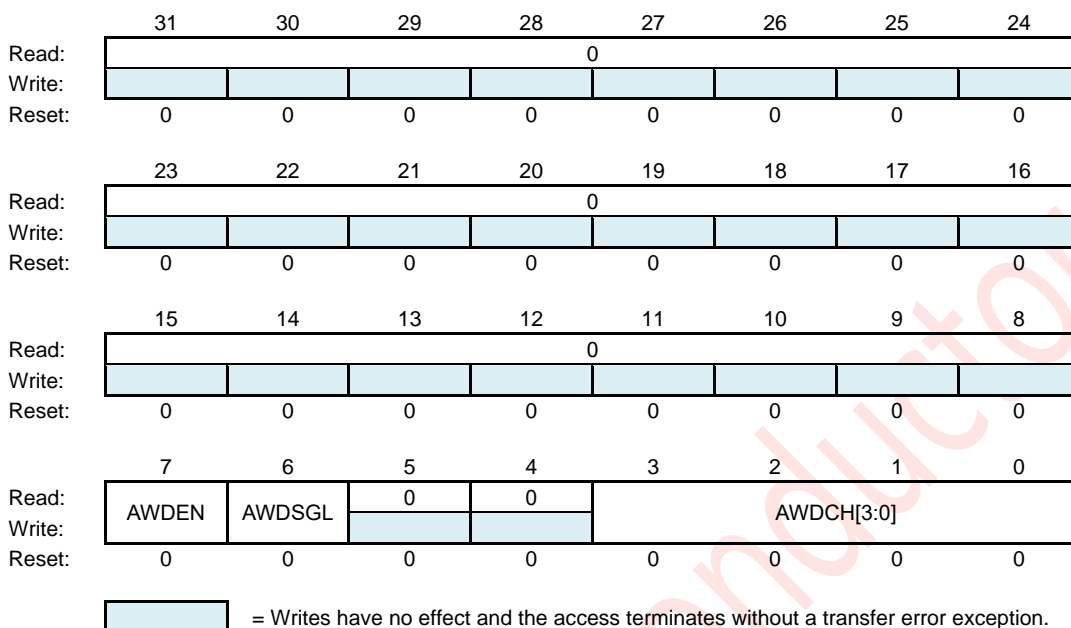


Figure 31-17: ADC Watchdog Register (ADC\_WDG)

Read: Anytime

Write: Before ADC start

**AWDEN** — Analog watchdog enable

This bit is set and cleared by software.

1 = Analog watchdog is enabled

0 = Analog watchdog is disabled

**AWDSDL** — 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 is enabled on a single channel

0 = Analog watchdog is enabled on all channels

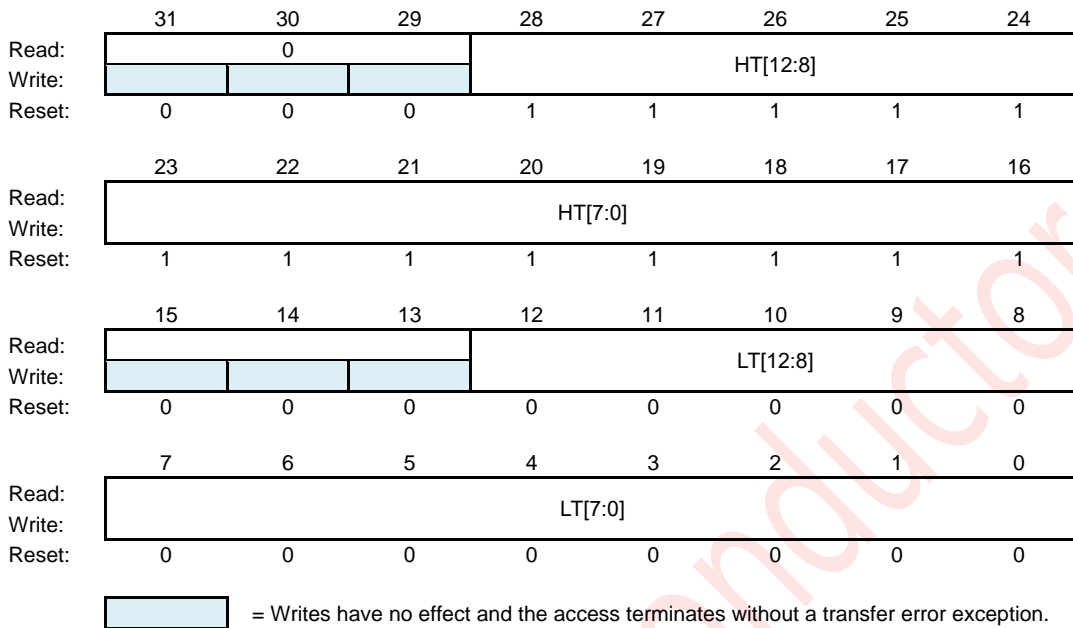
**AWDCH[3:0]** — Analog watchdog channel selection

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

**31.10.2.8. ADC Watchdog Threshold Register (ADC\_TR)**

**Address: ADC\_BASEADDR+0x0000\_001C**



**Figure 31-18: 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.

31.10.2.9. ADC Channel Selection Register (ADC\_CHSELR)

Address: ADC\_BASEADDR+0x0000\_002C

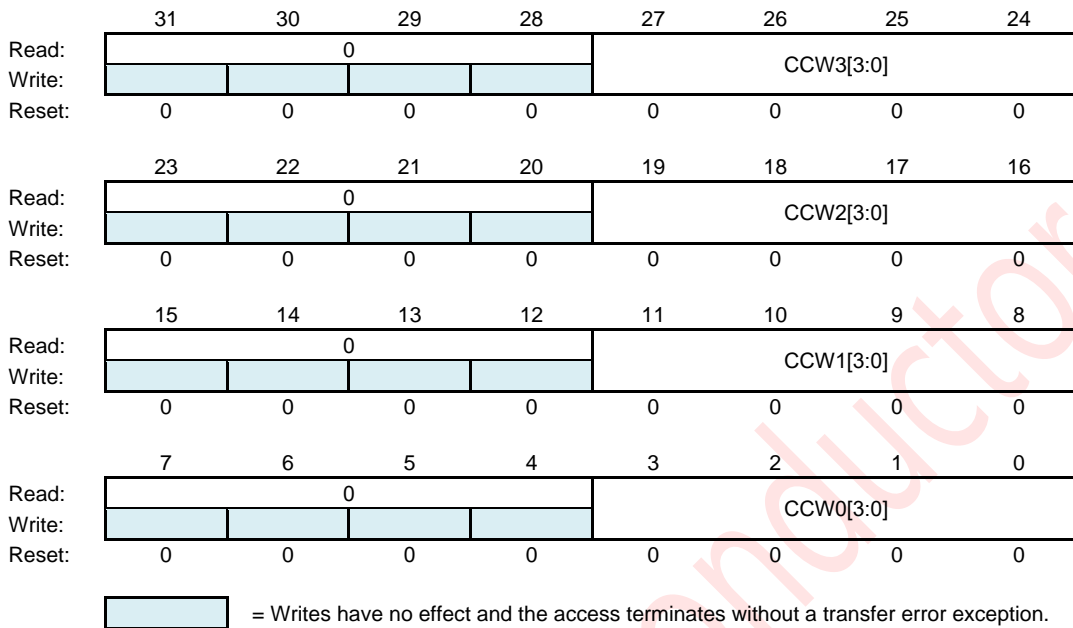


Figure 31-19: ADC Channel Selection Register 1(ADC\_CHSELR1)

Address: ADC\_BASEADDR+0x0000\_0030

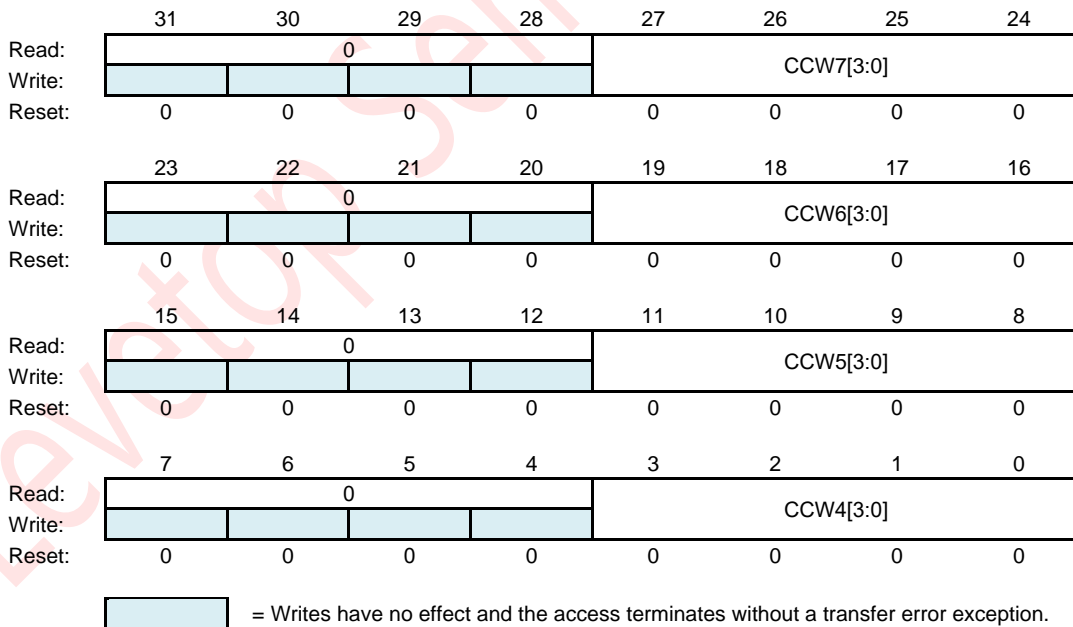


Figure 31-20: ADC Channel Selection Register 2(ADC\_CHSELR2)

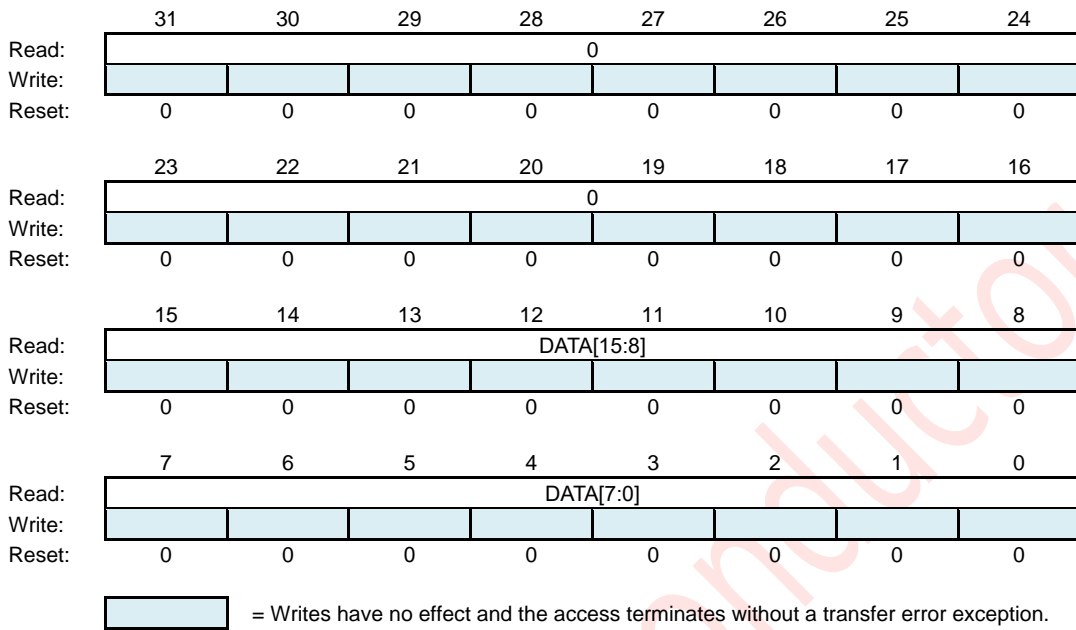
Read: Anytime

Write: Before ADC start

**CCWx[3:0]** — The number x conversion select channel, refer to **Table 31-1**.

**31.10.2.10. ADC FIFO Access Register (ADC\_FIFO)**

**Address: ADC\_BASEADDR+0x0000\_004C**



**Figure 31-21: ADC FIFO Access Register (ADC\_FIFO)**

Read: Anytime

Write: Never

**DATA[15:0]** — Converted data

Refer to **Table 31-4**.

## 32. Electrical Characteristic

This chapter provides the electrical characteristic parameters and limits for LT168.

### 32.1. Absolute Maximum Ratings

**Table 32-1: Absolute Maximum Rating**

Symbol	Item	Range	Unit
V <sub>DD33</sub>	Power Supply	-0.5 ~ 4.6	V
V <sub>IN</sub>	Input Voltage Range	-0.5 ~ V <sub>DD33</sub> +0.5	V
V <sub>OUT</sub>	Output Voltage Range	-0.5 ~ V <sub>DD33</sub> +0.5	V
P <sub>D</sub>	Power Dissipation	≤ 300	mW
T <sub>OPR</sub>	Operation Temperature	-40 ~ 105	°C
T <sub>ST</sub>	Storage Temperature	-55 ~ 150	°C
T <sub>SOL</sub>	Soldering Temperature	260	°C

If the loading to the chip exceeds the absolute maximum rating listed in **Table 32-1**, 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.

### 32.2. Electrostatic Discharge (ESD) Protection

**Table 32-2: 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°C

### 32.3. DC Electrical Specification

**Table 32-3: 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 <sub>IH</sub>	2.0	-	VDD33+0.3	V
Input Low Voltage	V <sub>IL</sub>	-0.3	-	0.8	V
Output High Voltage	V <sub>OH</sub>	2.4	-	VDD33	V
Output Low Voltage	V <sub>OL</sub>	0	-	0.4	V
Input Leakage Current	I <sub>IN</sub>	-	-	1	uA
Pull-Up Resistor	RPU	33	41	62	KΩ
Pull-Down Resistor	RPD	33	42	68	KΩ

**Table 32-4: 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

**Table 32-5: Power Consumption**

Item	Condition	Min	Typical	Max	Unit
Operation Mode @200MHz	LDO: Normal, Clock Source: On, System Clocks: On, ADC Clock: On	-	48.0	-	mA
Standby Mode 00	LDO: Normal, Clock Source: On, System Clocks: Off, ADC Clock: On	-	35	-	mA
Standby Mode 01	LDO: Normal, Clock Source: On, System Clocks: Off, ADC Clock: Off	-	34	-	mA
Standby Mode 10	LDO: Normal, Clock Source: Off, System Clocks: Off, ADC Clock: Off	-	230	-	uA
Standby Mode 11	LDO: STB, Clock Source: Off, System Clocks: Off, ADC Clock: Off	-	150	-	uA
RTC Operation Mode	V <sub>BAT</sub> = 3.3V, I <sub>RTC</sub>	-	3.0	-	uA

**Note:** Refer to **Table 9-5** "Sleep Operation Control Bit in Sleep Mode".



**Table 32-6: Wake up Timing Characteristic**

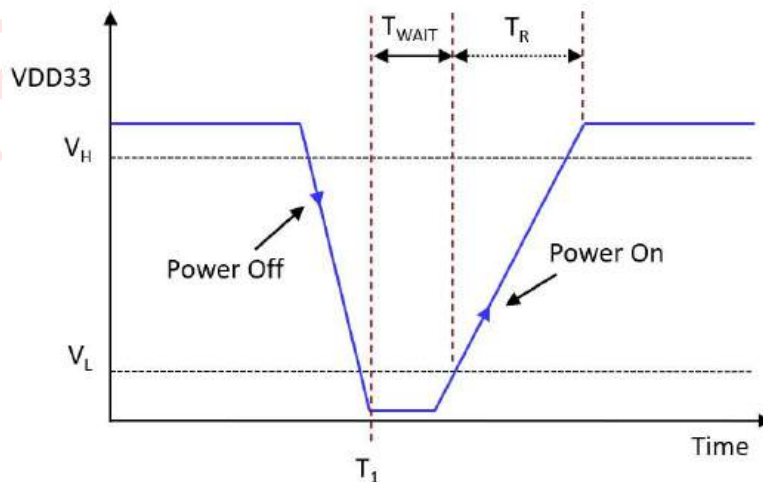
Item	Symbol	Min	Typical	Max	Unit
Power on Reset Time	T <sub>POR</sub>	-		-	uS
Lower Power Mode Wake up Time	T <sub>LP</sub>	-		-	uS
Stop Mode Wake up Time	T <sub>RET</sub>	-		-	uS
Deep Sleep Wake up Time	T <sub>DS</sub>	-		-	uS
Sleep Mode Wake up Time	T <sub>HIBER</sub>	-		-	uS

### 32.4. VDD Power Up Timing

When using the LT168x, 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<sub>L</sub>) at the time of power-up (T<sub>WAIT</sub>). At the same time, the rise time (T<sub>R</sub>) of VDD33 from V<sub>L</sub> to normal operating voltage should not be too long. The normal operating voltage range must be reached within 500ms. Otherwise, the MCU inside the LT168 will not boot properly.

**Table 32-7: VDD Power Up Characteristic**

Item	Symbol	Description	Min.	Nom.	Max.	Unit
Rise Time	T <sub>R</sub>	The rise time of input voltage from V <sub>L</sub> to the normal operating voltage	-	-	500	ms
Wait Time	T <sub>WAIT</sub>	The retention time of the V <sub>L</sub> before Power On	400	-	-	ms
VDD Input Voltage	V <sub>L</sub>	at T=T <sub>1</sub> on pin VDD33 (The input voltage before Power Up)	-	-	200	mV
VDD Input Voltage	V <sub>H</sub>	Normal Operation Voltage	2.97	3.3	3.63	V



**Figure 32-1: VDD Power up Timing Requirement**

### 33. Application Circuit

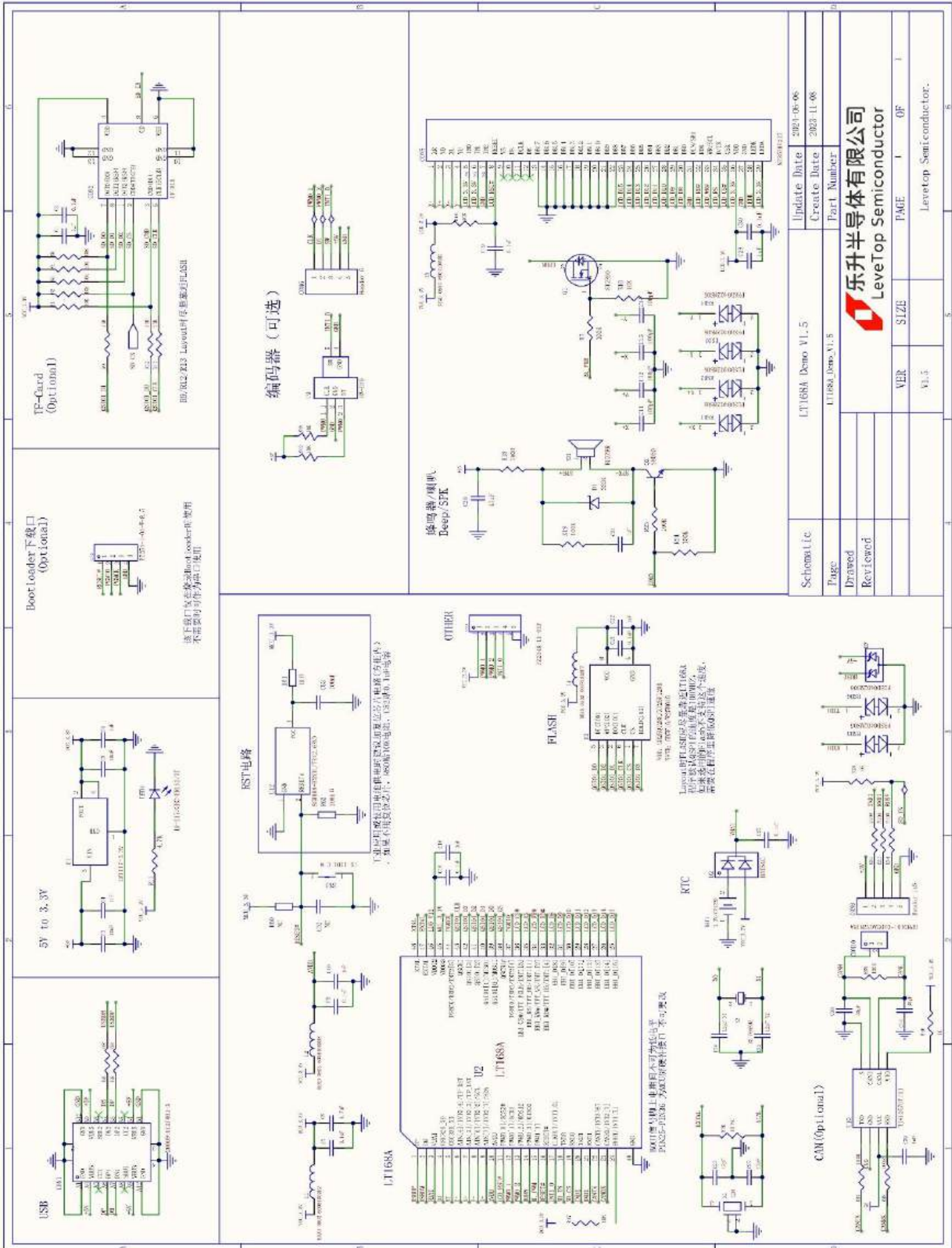


Figure 33-1: Application Circuit – LT168A for 8bit MCU Panel

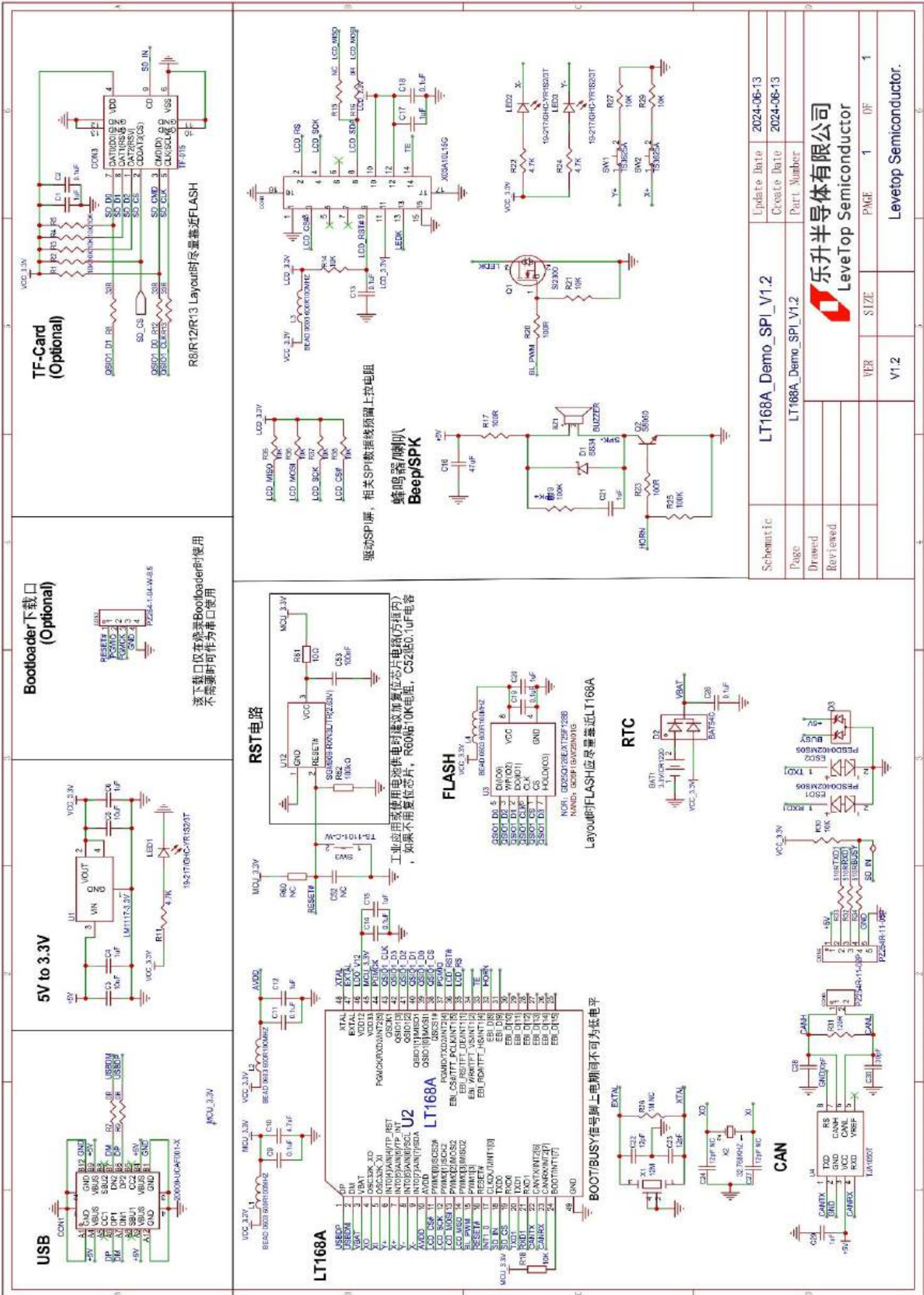


Figure 33-2: Application Circuit – LT168A for SPI Panel

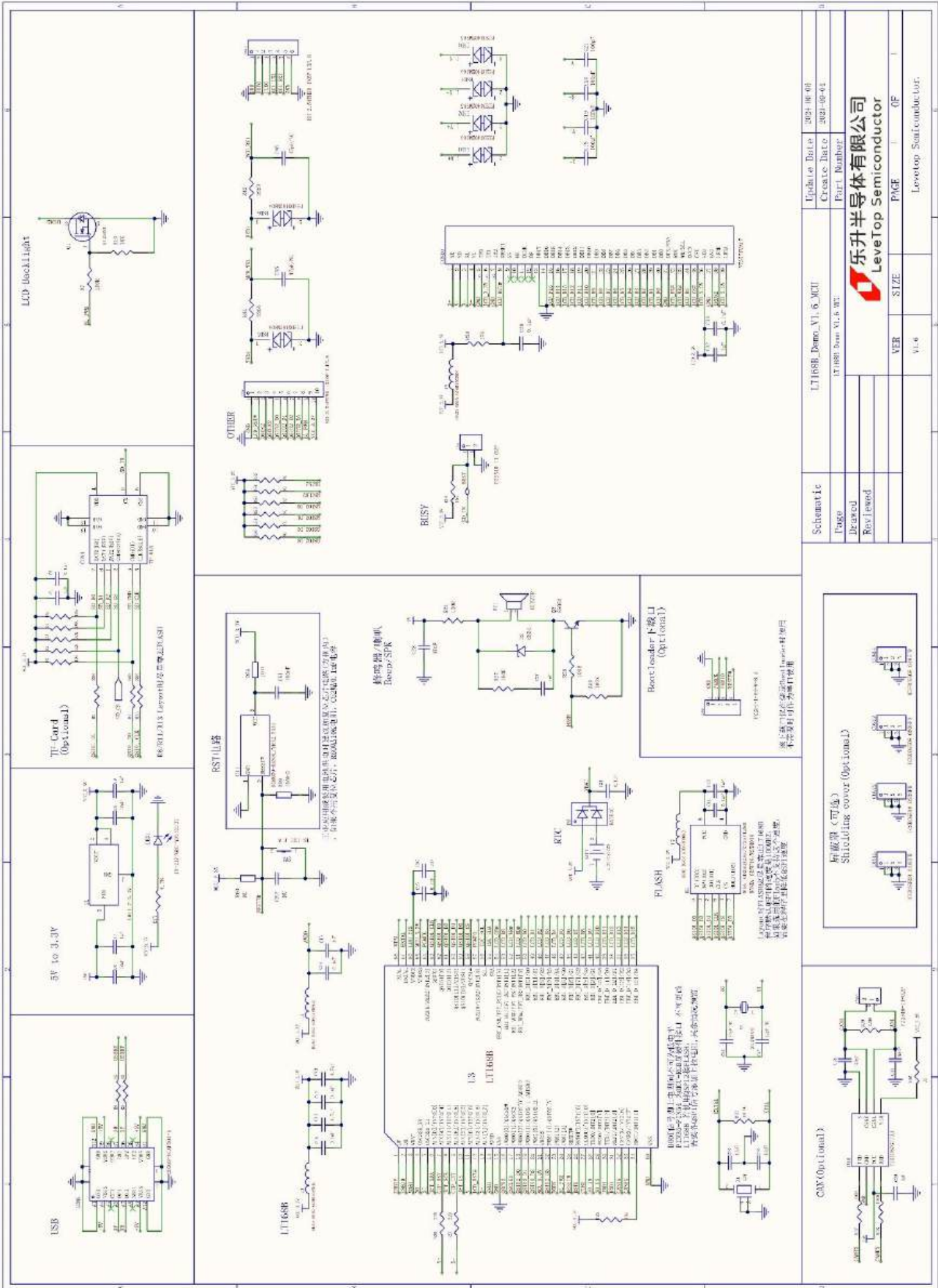


Figure 33-3: Application Circuit – LT168B for 16bit MCU Panel

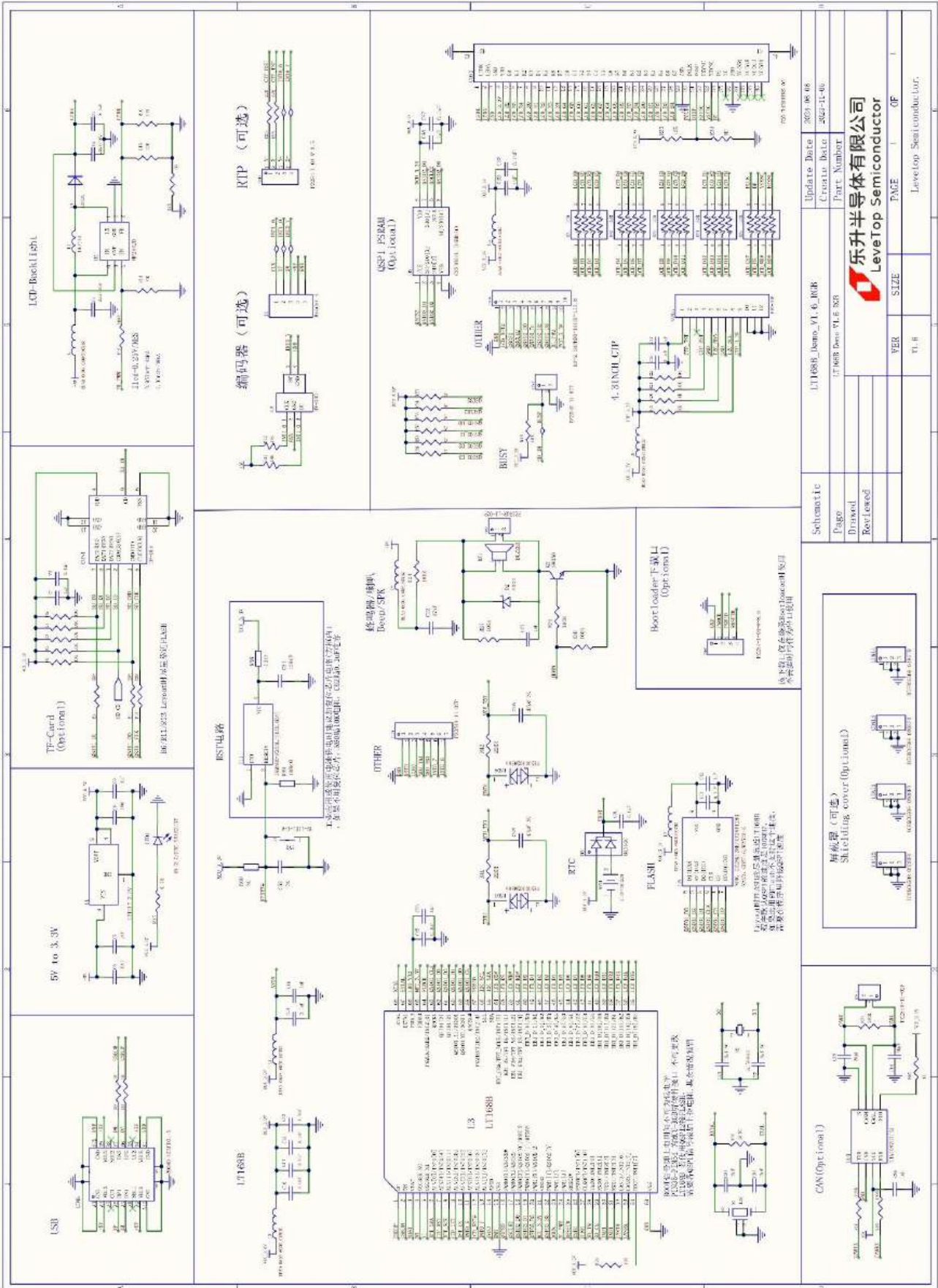


Figure 33-4: Application Circuit – LT168B for RGB TFT Panel

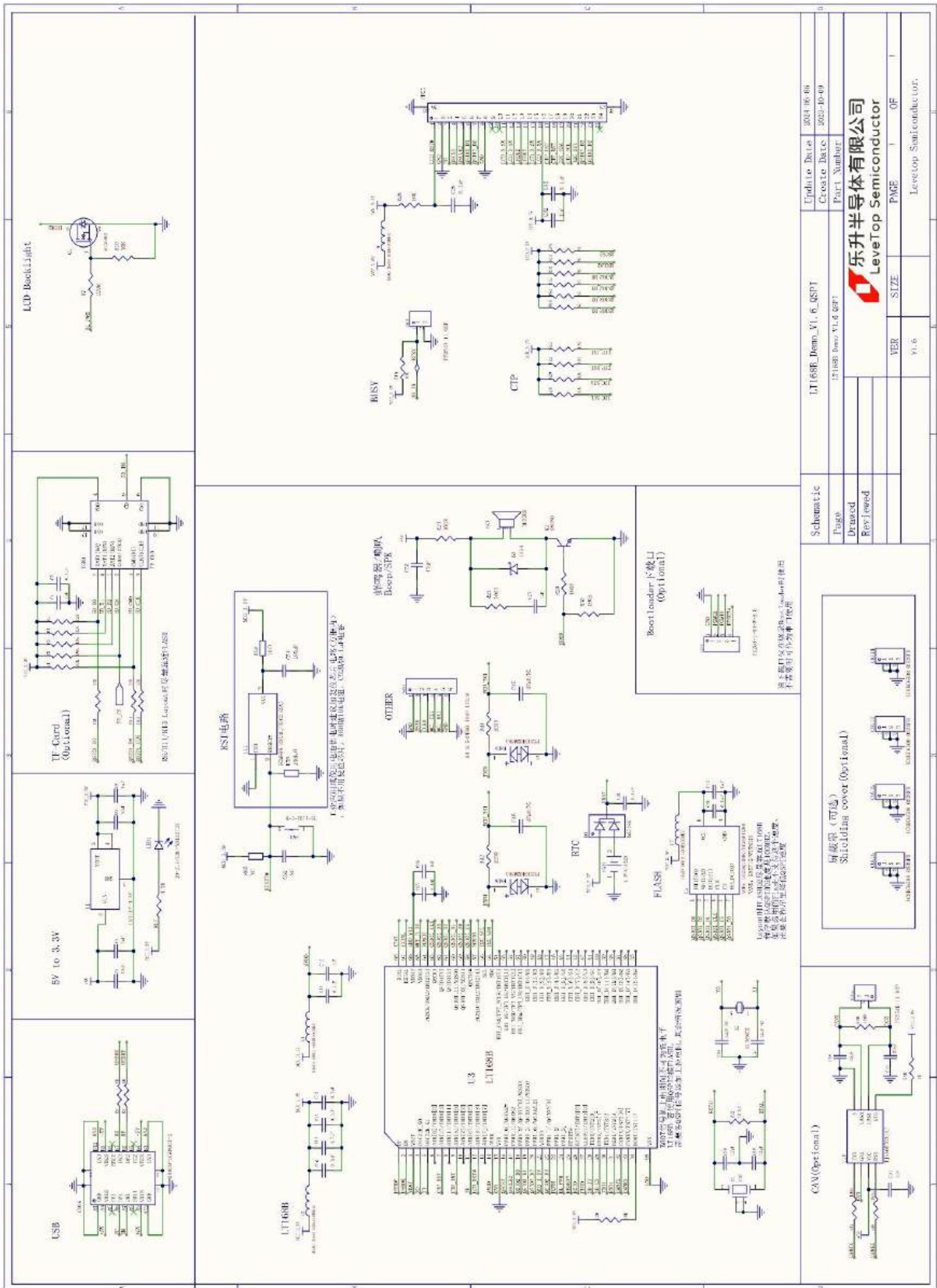
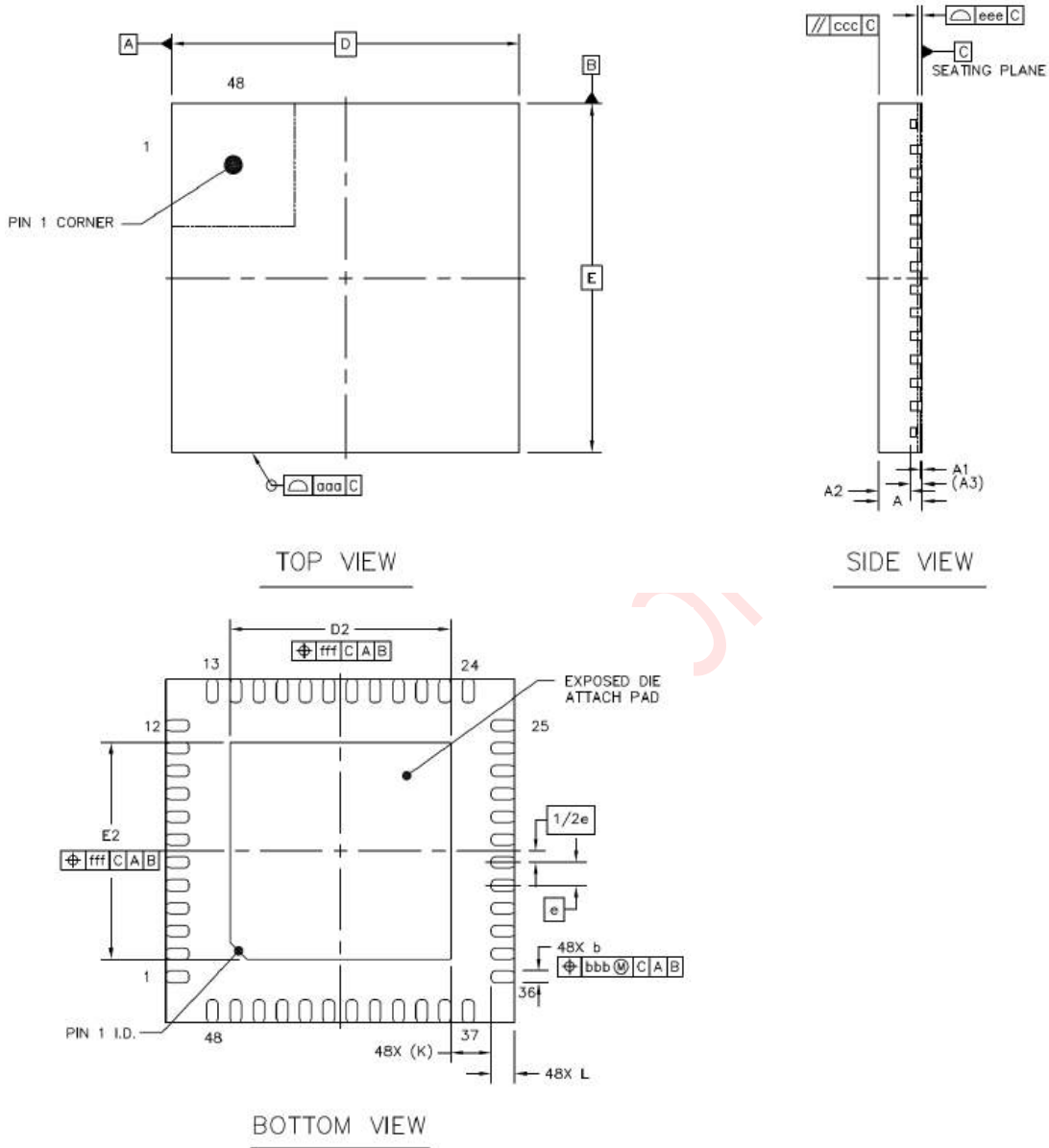


Figure 33-5: Application Circuit – LT168B for QSPI Panel

### 34. Package Information

#### 34.1. LT168A (QFN-48pin)



**Figure 34-1: LT168A Package Overview**

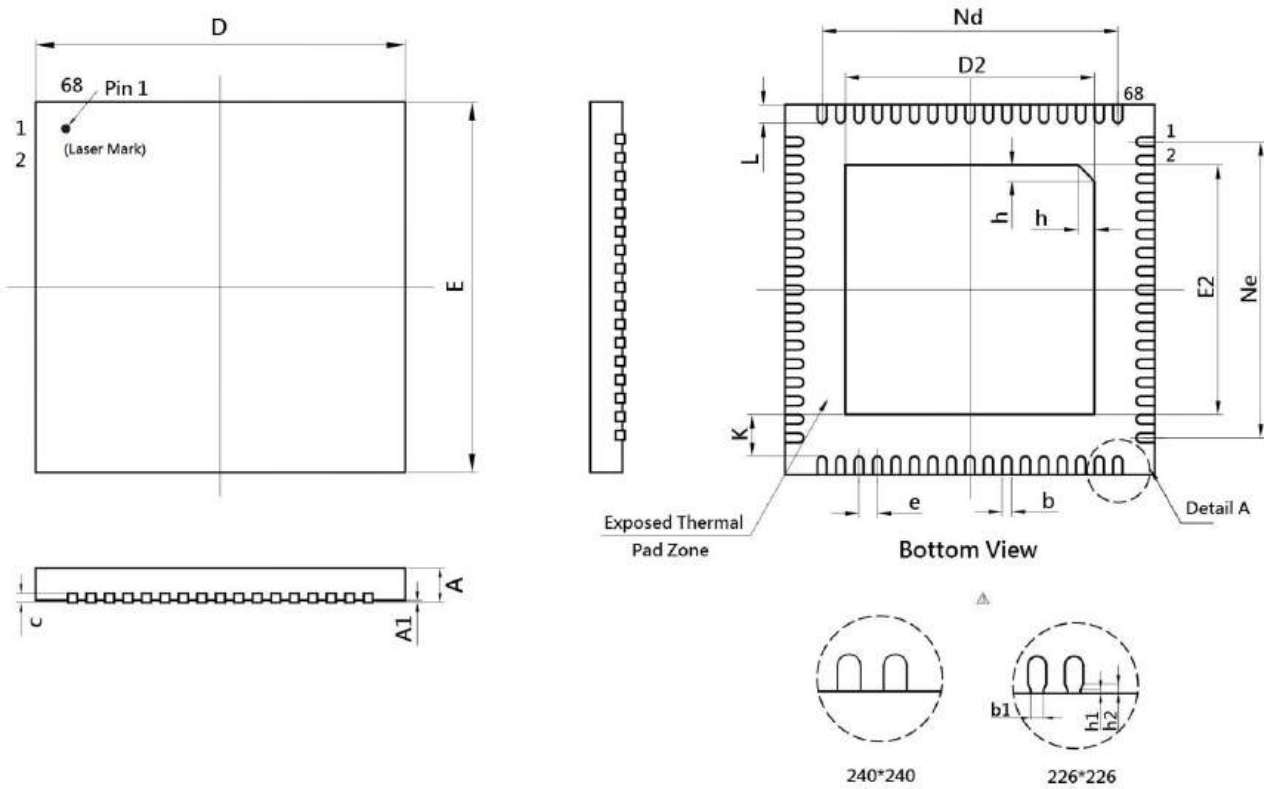
**Table 34-1: LT168A Package Parameter**

		Symbol	Min.	Nom.	Max.
Total Thickness		A	0.80	0.85	0.90
Stand Off		A1	0	0.02	0.05
Mold Thickness		A2	--	0.55	--
L/F Thickness		A3	0.203 Ref		
Lead Width		b	0.15	0.2	0.25
Body Size	X	D	6 BSC		
	Y	E	6 BSC		
Lead Pitch		e	0.4 BSC		
EP Size	X	D2	3.7	3.8	3.9
	Y	E2	3.7	3.8	3.9
Lead Length		L	0.3	0.4	0.5
Lead Tip to Exposed Pad Edge		K	0.7 Ref		
Package Edge Tolerance		aaa	0.1		
Mold Flatness		ccc	0.1		
Coplanarity		eee	0.08		
Lead Offset		bbb	0.07		
Exposed Pad Offset		fff	0.1		

**Note:** When layout PCB, LT168A Thermal Pad Zone must be connected to ground.



**34.2. LT168B (QFN-68pin)**



**Figure 34-2: LT168B Package Overview**

**Note:** When layout PCB, LT168B Thermal Pad Zone must be connected to ground.

**Table 34-2: LT168B Package Parameter**

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
<b>A</b>	0.80	0.85	0.90	<b>Ne</b>	6.40BSC		
<b>A1</b>	-	0.02	0.05	<b>L</b>	0.35	0.40	0.45
<b>b</b>	0.15	0.20	0.25	<b>K</b>	0.20	-	-
<b>b1</b>	0.14REF			<b>h</b>	0.30	0.35	0.40
<b>c</b>	0.18	0.20	0.25	<b>h1</b>	0.04REF		
<b>D</b>	7.90	8.00	8.10	<b>h2</b>	0.10REF		
<b>e</b>	0.40BSC			<b>D2</b>	5.39	5.49	5.59
<b>Nd</b>	6.40BSC			<b>E2</b>	5.39	5.49	5.59
<b>E</b>	7.9	8.0	8.10				

## 35. Copyright

The copyright of this document belongs to Levetop Semiconductor. If you need to copy, please obtain the license of Levetop Semiconductor in advance. Although the information contained in this document has been proofread, Levetop Semiconductor does not assume any responsibility for the specifications of the document. The application mentioned in the document is for reference only. Levetop Semiconductor does not guarantee that such applications do not require further modify. Levetop Semiconductor reserves the right to change its product specifications or documents without prior notice. For the latest product information, please visit our website: [www.levetop.cn](http://www.levetop.cn)

Levetop Semiconductor